3D-Modeling with respect to boundary representations within geo-DBMS

Edward Verbree
Sisi Zlatanova

Onderzoeksinstituut OTB

Technische Universiteit Delft

3D-Modeling with respect to boundary representations within geo-DBMS

GISt Report No.29

This research had been carried out at request of:
Topografische Dienst Kadaster - Afdeling Onderzoek en Ontwikkeling

Authors:
Edward Verbree
Sisi Zlatanova

OTB Research Institute for Housing,
Urban and Mobility Studies
Delft University of Technology
Jaffalaan 9, 2628 BX Delft, The Netherlands
Tel. +31 (0)15 278 30 05
Fax +31 (0)15 278 44 22
E-mail mailbox@otb.tudelft.nl
http://www.otb.tudelft.nl

Contents

Samenvatting

De Topografische Dienst Kadaster verzamelt, verwerkt en verstrekt topografische informatie van het gehele Nederlandse grondgebied. Binnen dit proces wordt de derde dimensie (hoogte) steeds belangrijker, gezien het multi-functionele grondgebruik, waarin steeds meer op, onder en boven elkaar wordt gebouwd.

De TU Delft, sectie GIS-technologie van het onderzoeksinstituut OTB, is gevraagd om binnen een viertal rapporten een beschrijving te geven van de huidige stand van zaken, de nieuwe mogelijkheden te schetsen en met aanbevelingen te komen voor vervolgonderzoek. Het eerste rapport (GISt22) geeft derhalve een overzicht van de beschikbaarheid van topografische hoogte-informatie binnen de Top10Vector, het tweede rapport (GISt28) gaat in op de ontwikkeling van Top10Vector naar Top10NL met specifiek aandacht voor de hoogtecomponent, het vierde rapport (GISt30) biedt een specifieke onderzoeksagenda voor de hoogtemodellering en hiervan afgeleide cartografische producten. Voor veel toepassingen is een hoogtemodellering van het grondgebied ruim voldoende; voor het opnemen van complete 3D-objecten binnen dit hoogtemodel is het echter noodzakelijk een goede aansluiting hiermee te realiseren. Vandaar dat dit rapport (GISt28) meer theoretisch van aard is met een overzicht van 3D-data modellen en 3D-datastructuren. Gezien het karakter van dit rapport is gekozen voor het Engels; deze samenvatting biedt een leidraad voor de lezer die snel de meest relevante informatie, conclusies en aanbevelingen tot zich wil nemen.

Hoofdstuk 1 beschrijft het topografische modelleer proces, waarbij objecten uit de werkelijkheid gerepresenteerd worden door een afbeelding in de database. Hierbij is de begrenzing van het object vaak leidend; zo wordt een perceel als een polygoon gerepresenteerd. Door de definitie van de grens is – impliciet – ook het inwendige van het object beschreven. Het is echter ook mogelijk het inwendige van het object expliciet te maken door deze op te delen in een verzameling aangrenzende, elkaar niet doorsnijdende driehoeken. Zo'n opdeling wordt een 2D-TIN (Triangular Irregular Network) genoemd. Vanuit deze TIN-representatie kan dan weer de begrenzing van het object worden afgeleid door de grensranden van het TIN te selecteren.

Deze aanpak werkt ook in binnen 3D. Als een object is beschreven door een verzameling vlakken die het object omhullen, zonder gaten of doorsnijdingen, dan is daarmee tevens het inwendige van het object – zij het impliciet – beschreven. Impliciet wil hier zeggen dat het inwendige weliswaar gedefinieerd is, maar dat er niet direct een rekenkundige bewerking mee kan worden uitgevoerd. Dat kan weer wel als op basis van de grensvlakken het inwendige wordt opgedeeld door een verzameling aanliggende, niet doorsnijdende, tetraëders. Zo'n beschrijving wordt een TEN (Tetrahedronized Irregular Network) genoemd. En net zo als binnen de 2D variant kan nu weer de begrenzing van het object worden afgeleid door de grensdriehoeken van het TEN te selecteren en deze begrenzing te representeren als een 3D-TIN.

Er bestaan diverse classificaties van het modelleren van 3D-object-begrenzingen. *Hoofdstuk 2* geeft hiervan een overzicht, met achtereenvolgens een onderscheid ten aanzien van: 2.1 topologie/geometrie; 2.2 vlakbeschrijving op basis van driehoeken

of op basis van 'simple features' vlakken; 2.3 al dan niet veronderstelde valide (waterdichte) definitie van de begrenzing; 2.4 al dan niet de mogelijkheid tot het expliciet modelleren van het inwendige van het object. Een pure geometrische beschrijving leunt er sterk op de applicatie-laag voor het afdwingen van een onder andere het valideren van de 3D-object begrenzing, terwijl bij het topologische model deze valide status zelf kan afdwingen en onderhouden. Bovendien is vanuit het topologische model eenvoudig een geometrische modellering kan afleiden.

Hoofdstuk 3 biedt een overzicht van diverse realisaties van 3D topologische datamodellen. Deze worden beoordeeld aan de hand van de criteria 2.2, 2.3 en 2.4 uit hoofdstuk 2. Gezien het belang van een valide begrenzing met de daarbij behorende mogelijkheid tot een expliciete vastlegging van het inwendige van het object, komt uit deze vergelijking naar voren dat de 3D-TIN/TEN en het Urban Data Model (UDM) het meest geschikt zijn.

Hoofdstuk 4 behandelt de structurering en opslag van 3D Data in Geo-Database Management Systemen (geo-DBMS). Ook hier vindt een onderscheid plaats op topologische en op geometrische representaties en de implementatie van deze modellen binnen een geo-DBMS omgeving. Het grote voordeel van geometrische modellen is dat deze vrijwel direct vanuit het geo-DBMS door front-end GIS-omgevingen geraadpleegd en gevisualiseerd kunnen worden. Maar met een tussenstap is de geometrie ook vanuit de topologische tabellen af te leiden.

Als een directe verbinding tussen geo-DBMS en GIS-applicatie niet mogelijk is, dan kan de data via een bestand worden uitgewisseld. Aangezien GML hiervoor de standaard is, behandelt *hoofdstuk 5* de mogelijkheid tot het structureren van 3D Data in GML3. Aangetoond wordt dat binnen GML3 zowel de geometrie als de topologie, maar ook de symmetrie tussen geometrie en topologie gestructureerd beschreven kan worden.

Veel andere dataformaten bieden ook de mogelijkheid tot de uitwisseling van 3D data. In *hoofdstuk 6* wordt een kort overzicht gegeven van zowel een pure geometrische als tweetal pure topologische bestandsformaten. Een groot nadeel van zowel uitwisseling via GML3 als via de meer traditionele bestandsformaten is dat er geen zekerheid meer kan worden geboden dat het geleverde bestand valide is, dat wil zeggen voldoet aan de eigenschappen die oorspronkelijk werden gesteld aan het achterliggende datamodel.

Vandaar dat in de conclusies en aanbevelingen in *hoofdstuk 7* de representatie van de begrenzingen van een object door middel van een 3D-TIN wordt benadrukt. Deze wijze van representatie werkt met simpele primitieven (driehoeksbedekking), de validatie van het object is relatief eenvoudig en vanuit de 3D-TIN is een expliciete TEN volume representatie af te leiden. Als het 3D-TIN/TEN model binnen een geo-DBMS wordt opgeslagen en beheerd dan is zowel een topologische representatie als een geometrische representatie 1-op-1 te behouden. Dat biedt grote voordelen ten aanzien van de koppeling naar 3D-GIS; er is een snelle visualisatie mogelijk door het opvragen van de geometrische materialisatie van het 3D-TIN/TEN datamodel, en de validatie, indexing en ruimtelijke operaties kunnen op database niveau worden uitgevoerd. Daarnaast is vanuit de geo-DBMS een snelle uitwisseling mogelijk naar andere systemen door een export via GMS3 als naar andere 3D bestandsformaten.

Introduction

This report is the third in a series of four dedicated to a framework for implementing full 3D-topographical mapping of the Netherlands by order of the *Topografische Dienst Kadaster*.

The first report (GISt22) gave an overview of the availability of topographical height information as delivered within the *Top10Vector*. The second report (GISt28) described the topographical elements available within the successor of the *Top10Vector*, the *Top10NL*. The focus of this report is on 3D-data models and 3D-data formats within current geo-information technology with special attention to the so-called TIN/TEN approach. The final report (GISt30) addresses the research agenda on the introduction of 3D Data within the *Top10NL*.

The aim of this report is to give a description from a practical point of view and therefore it is restricted to 3D-data models for boundary representations. A theoretical background of 3D-data models is given, but only to address the different kinds of implementations. The explanation of 3D-exhange data formats is limitative to the most used within geo-DBMS and GIS practice.

This report is structured as follows: Chapter 1 gives an introduction on data modeling with respect to boundary representations. A classification on these kinds of data models is given in chapter 2. Actual examples of theoretical data models are described in chapter 3, and classified according the categorization of chapter 2. The 3D has to be structured according the data model into a Geo-DBMS. As Geo-DBMS are not prepared for this task yet, chapter 4 gives some background on geometrical and topological structures and implementation schemes to work around that problem in an efficient way. Direct connections to Geo-DBMS are not mainstream, and as 3D-data has to be exchanged, some file exchange is still necessary. As GML is the 'language' appropriate for this task, the possibilities on 3D data are given in chapter 5. We will show that GML can store both geometry and topology, but as 3D Data handling however is only possible in GML3, file exchange with 'older' but much used file formats is still common practice. Therefore some examples of these kinds of files are given in chapter 6. Chapter 6 ends with the conclusions and recommendations

The language of this report is chosen to be English; as parts of this report are cited from references, i.e. the Workshop on 3D City Modeling at UDMS 2004 (Zlatanova, 2004)) and we believe the audience for this report will be broader than a text available in Dutch. For those who want to have an indication of the contents in *Dutch*, a *Samenvatting* is given.

1 Data modeling with respect to boundary representations

1.1 2D-Boundary representations

Within the topographical modeling process real-world objects are characterized by a certain representation and stored within the geo-database. The representation depends on the purpose of the application of the geo-database, but also on the identification and data-capture process.

To illustrate this thought buildings in 2D-geodatabases are mostly represented by closed polygons, as this will simplify object-identification and color mapping in GIS. Most buildings in rural areas are identified and captured as closed polygons from aerial images or semi-automatically reconstructed to a closed polygon from loose corner-points measures.

We can describe a 2D-boundary explicitly by its polygon geometry through an ordered list of points, which are connected one-by-one by straight-line segments and the last point connects to the first. We could also choose for a kind of topology, where the points are identified as numbered nodes and a list of node-numbers defines the polygon. The geometrical description of polygons and other simple features within 2D-space is defined by the Simple Feature Specification (SFS) of the Open GeoSpatial Consortium (OGC, 2005). Validation functions are available to determine whether or not a given feature is valid, i.e. a polygon is not self-intersecting. Although this should be not that complicated, many implementation and definition problems still exists; see (van Oosterom et al., 2003) and (van Oosterom et al., 2004).

If the boundary of an object is defined, then the interior is also – implicitly – given. This interior could also be made explicit, i.e. when the boundary polygon is triangulated in a set of triangles and stored within a 2D-Triangulated Irregular Network (2D-TIN) data structure. As it is possible to derive the boundary edges out of the set of triangles it is now not necessary to store the boundary polygon explicitly. The TIN acts as the base-data structure to contain the feature representation. The implicit supposed space 'between' the two or more objects could also be made explicit by a triangulation of the covering polygon, leaving all enclosed objects by the sets of interior triangulations. Now the space is fully partitioned by triangular meshes.

1.2 3D-boundary representation

The identification and capture of real-world objects is far more complicated when we move to full 3D-applications. In 3D, opposed to 2.5D, it is not possible to assume that objects can be flattened and defined as a polygon-footprint on a surface. Instead a proper 3D boundary representation of the object is needed, meaning more than one Z-value is attached to a 2D-polygon, and also more information is needed than one Z-value attached to the vertices of the 2D-polygon.

In 3D the definition of the geometrical and/or topological description and validation of the correctness is far more complicated than in 2D. The polyhedral approach, as described in among others (Stoter, 2004) defines the boundary as a set of polygons, where each polygon should be 'flat' and valid according to the SFS of the OGC, i.e. non-self intersecting. The set of faces should enclose a single volume, meaning it should be 'watertight' or 'closed'. The validation of these kinds of boundary representations is quite complex, as it consists of checking the '2-manifold' characteristic. If we limit our polyhedra not having any inner holes this polyhedron is non-valid when Euler's Law does not hold: the number of vertices (nodes) plus the number of faces minus the number of edges is not equal to 2. However, when the equation holds it is still not certain that the polyhedron is valid as the faces can intersect with each other. To determine whether or not faces do intersect is easier said than done.

One way to simplify that problem is a pre-processing step where each face is subdivided into a set of triangles. Not only does this avoid the check on 'flat' polygons, as a triangle is flat by definition.

Triangular faces are easier to manage within a topological data model and the rendering engines work with triangles (i.e. for visualization everything is anyhow subdivided into triangles).

When an object is described by a valid, 'watertight', set of triangles (see Figure 1) the interior is also implicitly given. Similar to the 2D-scenario this interior can be made explicit by a set of tetrahedrons (Figure 2), which fill the interior completely. This task is performed by a tetrahedronization. Once this Tetrahedronized Network (TEN) is available the boundary can be derived from that data structure. This boundary is represented by a set of triangles and could be explicitly made available within a 3D-Triangulated Irregular Network (3D-TIN) as shown in Figure 3.

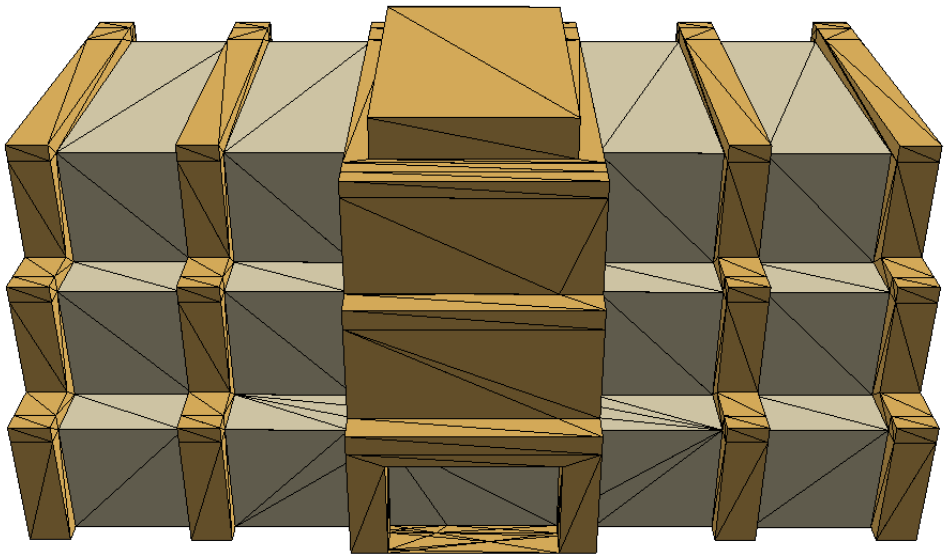


Figure 1: Valid, 'watertight' triangular boundary representation

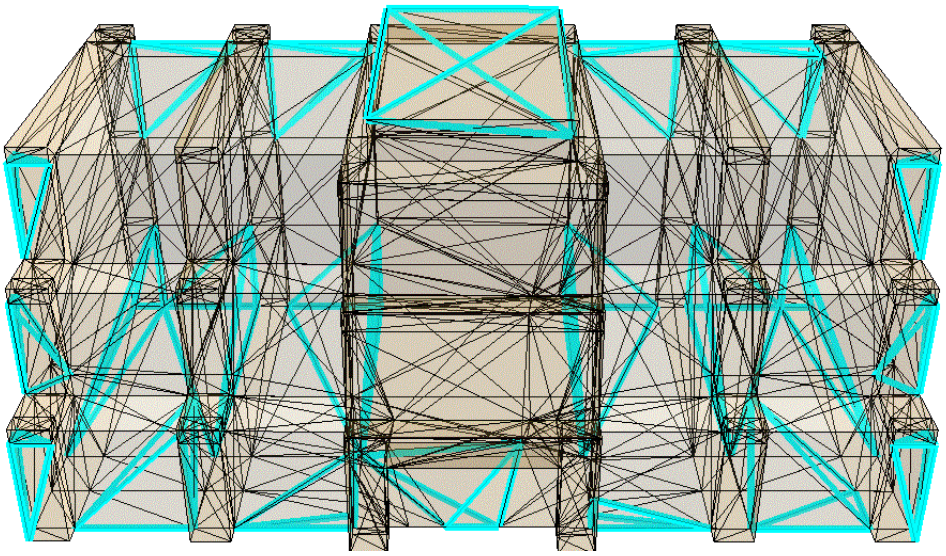


Figure 2: Boundary and Interior given as Tetrahedronized Network (TEN)

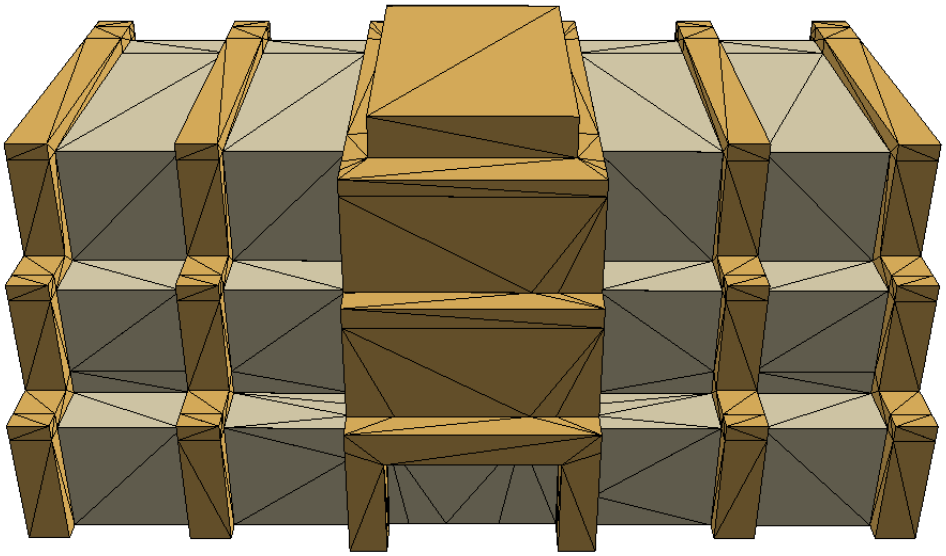


Figure 3: Boundary given as 3D Triangular Network derived from TEN model

The applicability of the data model depends also on the availability of spatial operators (Pilouk, 1996). Metric operators are built around the computational geometry, for example point-in-polygon, point-in-body, intersection of lines, and intersection of surfaces. Metric operators can be used to derive topological relationships. With respect to the Tetrahedronized Network (TEN), these problems can be reduced to the level of simplex. Point-in-polygon and point-in-body can be reduced to point-in-triangle and point-in-tetrahedron respectively. The intersection-test of bodies can be simplified to intersections between tetrahedrons, which can be further reduced to intersections between simplices of a lower dimension, for example between triangle and edges. The edge-triangle intersection test can be performed without the actual calculation of the intersecting points, see Figure 4 (O'Rourke, 1998), (Möller, 1997).

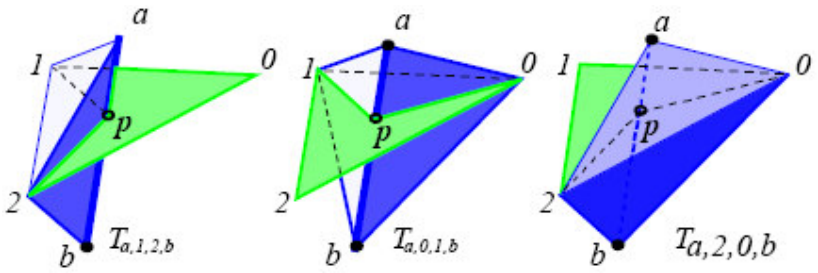


Figure 4: Edge-Triangle intersection test by calculation signed volumes

2 Classification of boundary representations

2.1 Topological / geometrical data model

We could divide the different kinds of implementations first of all by the underlying data model. The geometrical data model is implemented through a list of lists for the coordinates for each face. The topological data model is realized through a list of coordinates for the nodes or a list of coordinates for the edges at one hand and a list of references to these nodes or edges to materialize the faces at the other. How this list of references is structured depends on the core data model.

Many different 3D-topological data models exist, as the kind of applications to be supported by this data model varies. Dedicated GIS environment will opt for a combination of the topological and geometrical data model as they perform both topological queries, i.e. neighborhood adjacency, and geometrical queries, i.e. spatial searches.

The example of a 2D topological data model is realized within the ArcInfo Coverage. Here an ARC file holds the polygon boundary geometry as topological edges, which are referred to as arcs. The polygon-arc list (PAL) file lists the order and direction of the arcs in each polygon. In ArcInfo software logic is used to assemble the coordinates for each polygon (from the ARC and PAL files) for display, geometric analysis, and query operations. The polygons are assembled during run time when needed. This ESRI propriety data model is never extended to three dimensions.

The topology data model is opted very well for neighborhood relations. Other operations (like area and volume calculations, intersections) require a full geometrical description of the features, and therefore there is a shift from topological/geometrical implementations to pure geometrical implementations. When topology is needed it is computed on the fly, which is also made possible by the increased processing power of current computers.

Especially geo-DBMS will opt for the geometrical model as these database management systems will request as much information as possible from each record within one request for visualization purposes. The geometry, i.e. a 3D multipolygon description, of one object can be stored within one record, so there will be a one-to-one correspondence between the record and the object. Front-end application (like CAD or GIS on top of a geo-DBMS) will recognize and understand these geometrical objects and are able to visualize this kind of geometrical representation without conversions. Some limitations are to be treaded within editing performed by these CAD/GIS applications, as these edits have to be passed on the geo-DBMS.

2.2 Triangular faces / simple features faces

The second classification with respect to boundary representations is the type of polygons allowed to describe the polyhedron. Some implementations choose for the limitation of triangle patches, others permit a polygon description according to the simple feature definition for polygons, with an unlimited number of vertices for each

face. Although this definition will prevent a face to intersect itself, the main difficulty in 3D is that the 'flatness' of the face is not guaranteed.

2.3 Valid boundary / non-valid boundary

The insistence of a valid, watertight, boundary could be regarded as the third characteristic of a boundary representation. If this important, but hardly non-existing implemented, property is indicated and maintained within the implementation within the datastructure it is not needed to perform a test each time.

2.4 Interior modeled / interior not modeled

When the boundary is modeled by a set of triangles (3D-TIN) it will simplify the explicit modeling of the interior, which we will regard as the fourth kind of classification. A polyhedron described by a set of polygons is an example of an implicit representation of the interior. Only when the boundary is confirmed to be valid the interior is believed to exist. The TEN models the interior of the object explicit by a set of tetrahedrons.

3 Review of 3D topological data models

In the paper "Topological models and frameworks for 3D spatial objects" (Zlatanova, 2004) and also during the workshop on 3D City Modeling (Zlatanova, 2004) an overview of 3D topological models and 3D data structures is given. The topology is used primary to derive the geometrical materialization of the objects, not to store topological relationships between the objects. We will focus ourselves in section 3.1 till 3.5 to the following topological models: 3D FDS, PLC, TEN, SSM and UDM. The models are described shortly and classified in section 3.6 according section 2.

3.1 3D FDS – 3D Formal Data Structure

The *3D Formal Data Structure* is the first data structure that considers the spatial object an integration of geometric and thematic properties. A conceptual model and 12 conventions (rules for partitioning of physical objects) define the structure (Molenaar 1990). The data structure consists of three fundamental levels: *feature* (related to a thematic class), four *elementary objects* (*point*, *line*, *surface* and *body*) and four *primitives* (*node*, *arc*, *face* and *edge*). According to the conventions, arcs and faces cannot intersect. A node and an arc must be created instead. Singularities are permitted in such a way that arcs and nodes can exist inside faces or bodies. The role of the edge is dual, i.e. to define the border of a face (relationship *face-arc*) and establish an orientation for a face, which is needed to specify left and right body. The number of arcs constituting an edge is not restricted. Arcs must be straight lines and faces must be planar. The surface has one outer boundary and may have several non-nested boundaries, i.e. may have holes or islands. The body has one outer surface and can have several non-nested bodies or holes.

The 3D-FDS is not limited to triangular boundary faces. The boundary itself is according the conventions valid, but this property should be checked. The body object models the interior explicitly.

3.2 PLC – Piecewise Linear Complex

Three-dimensional geometric objects are often more complicated than polyhedra. A more general description is called *piecewise linear complex* (PLC). A PLC X is a set of vertices, segments and *facets*. Each facet is a polygonal region; it may have any number of sides and may be non-convex, possibly with holes, segments and vertices in it. PLCs have restrictions like any other complex. For a PLC X, the elements of X must be closed under intersection. For example, two segments only can intersect at a common vertex that is also in X. Two facets of X may intersect only at a shared segment or vertex or a union of shared segments and vertices (because facets are non-convex).

The PLC is not limited to triangular boundary faces. The boundary itself is according the definition closed under intersection and therefore valid, but this property should be checked. The boundary faces implicitly model the interior.

3.3 3D-TIN/TEN – 3D Triangular Irregular / Tetrahedronized Network

The *Tetrahedronized Irregular Network* (Figure 5) was introduced by (Pilouk 1996) to overcome some difficulties of 3D FDS in modeling objects with indiscernible boundaries (such as geological formations, pollution clouds, etc.). TEN follows simplex-oriented approach to represent 3D object from real world proposed by (Carlson 1987). Similarly to it, TEN has four primitives (tetrahedron, triangle, arc and node). The relationship arc-node is given by the ARC table; the TRIANGLE table contains the tetrahedron-triangle-edge link. A body object is composed of tetrahedrons, a surface object of triangles, a line object of arcs and a point object of nodes.

The surface of the object is in fact the boundary representation, and called a 3D Triangular Irregular Network (3D-TIN). If an object is defined by a valid TEN, then the interior of this object is given by the set of tetrahedrons and the boundary of that object is given by the set of triangles at the surface. This boundary is thus represented by a Triangulated Irregular Network in 3D-space, and called a 3D-TIN.

In most cases however the TEN of an object has to be build by a given boundary representation, i.e. a PLC or UDM (see below). This is quite a complex task to do; the program TETGEN (Tetgen 2005) developed by Hang Si gives an efficient implementation. To build a valid tetrahedronization, a valid boundary has to be provided. But once this TEN is build, a valid boundary 3D-TIN is also at hand.

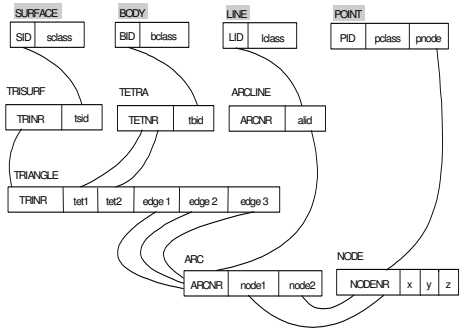


Figure 5: TEN-datastructure

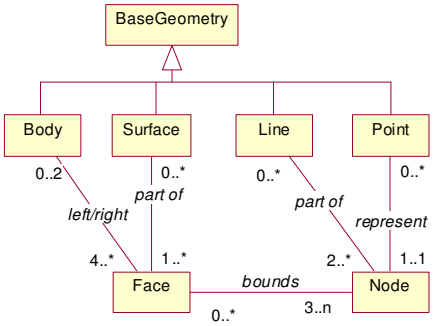


Figure 6: UDM-datastructure

3.4 SSS – Simplified Spatial Structure

The *Simplified Spatial Structure* was designed to serve web-oriented applications with many visualization queries (Zlatanova 2000). The basic objects are again four but the primitives used are only two, i.e. *node* and *face*. The motivation for omitting the arc of explicitly stored elements is that the uniqueness of the relationship arc/face in 3D space is lost, i.e. one arc can be part of more than two faces. 3D primitive is not maintained as well and convex faces represent the 3D objects. Faces must be planar. Intersecting primitives lead to new once. *Node-in-face* and *face-in-body* are explicitly stored.

The SSS is not limited to triangular boundary faces, the boundary itself is according the design valid, but the interior body is not explicitly modeled.

3.5 UDM – Urban Data Model

The *Urban Data Model* (Figure 6) represents the geometry of a body or a surface by planar convex faces (Coors, 2002). Each face is defined by a set of nodes. Two convex planar polygons are adjacent if they share at least two nodes. The orientation of a face is stored implicitly. In the relational representation of the model, every arbitrary polygon is decomposed into triangles and the face table contains only three columns, i.e. the ID's of the three nodes. The one-dimensional construction primitive (arc or edge) is not supported as well. In piecewise linear geometry models, these elements are implicitly defined by two successive points.

The UDM is limited to triangular boundary faces, the boundary itself is according the definition valid, but the interior body is not explicitly modeled.

3.6 Classification of 3D data models

At the end of each section a short indication of the 3D model according the classification in chapter 2 has been given. We will summarize these remarks in the following table:

	Faces: Triangular/ Simple	Boundary: Valid / Non-valid	Interior Implicit / Explicit
3D FDS	Simple	Non-Valid	Explicit
PLC	Simple	Non-Valid	Implicit
3D-TIN/TEN	Triangular	Valid	Explicit
SSS	Simple	Non-Valid	Implicit
UDM	Triangular	Valid	Implicit

As we have stated that a valid boundary guaranty is the core requirement on boundary representations, the 3D-TIN/TEN and the Urban Data Model (UDM) are the data models of choice. In fact the UDM is a kind of a 3D-TIN, but not derived from the interior TEN. The UDM can however act as the input for the TEN creation programs. Boundary representations modeled as a 3D-TIN or as an UDM are therefore the preferred choice.

4 Structuring 3D data in geo-DBMS

Clearly, there are a lot of reasons to use geo-DBMS in many systems: multi-user control on shared data and crash recovery automatically locks of single objects while using database transactions, advanced database protocols mechanisms to prevent the loss of data, data security, data integrity and operations that comfortably retrieve, insert and update data. 3D modeling of objects and their relationships result in a lot of data and logically the question is whether DBMS can maintain and analyze 3D geodata. The expectations are that the developments within the DBMS will become at stage sufficient for management of 3D spatial data.

As mentioned in the previous section OGC has suggested maintenance of both geometry and topology. Apparently for some period of time two models (topology and geometry) will be maintained in the geo-DBMS. Although there is believe a topological model might be sufficient (geometry can be derived from topology), in the same way one could argue that the same is true for a geometry model, because topology can be derived from geometry. In this respect, one can choose between three different options:

1) Storing the topological model in geo-DBMS (and deriving geometry from it).

2) Storing the geometric model in geo-DBMS (and deriving topology from it).

3) Storing both models in the geo-DBMS.

Advantages and disadvantages of these approaches as well as the consequences for spatial analysis will be discussed in detail.

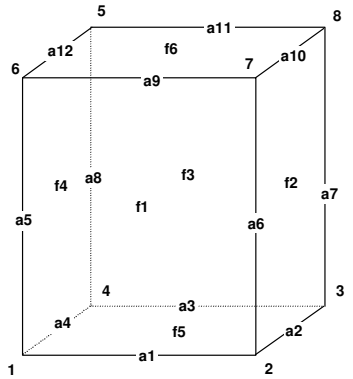


Figure 7: Topological representation

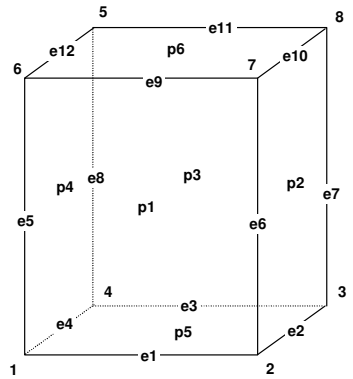


Figure 8: Geometrical representation

4.1 Topological representation

In approach 1 the topological relationships between object parts, such as 'all surfaces belonging to the boundary of a 3D volume object' are stored in a topology model as

described above (see Figure 7). Thus the location of objects, i.e. the x-,y-,z-coordinates of its defining points, are not part of the topology model and usually they are stored separately. The advantage of this approach is that relationships between objects can quickly be identified by their topological properties. Considering the notations given (see Figure 8):

If the coordinates of the box are stored with the nodes:

```
node1: 0, 0, 0
node2: 1, 0, 0
node7: 1, 0, 1
node6: 0, 0, 1
```

then the arcs will be represented by:

```
arc1: n1, n2
arc6: n2, n6
arc9: n7, n6
arc5: n6, n1
```

and the faces by:

```
face1: a1, a6, a9, a5
face2: a2, a7, a10, a6
```

If one needs to know the common arc between face1 and face2, then it is sufficient to check the arcs in the list of the two faces. However, geometric database queries such as "return the geometry" cannot profit from the explicit knowledge of the objects topology. Needed now are the coordinates of the object.

4.2 Geometrical representation

Approach 2 prevents the disadvantage of approach 1, however, topological database queries cannot be answered efficiently by providing a geometric model only. Using the notations of geometry (figure 3b), the same faces of the box will be described by:

```
vertex 1: (0,0,0)
…..
edge 1: (0,0,0, 1,0,0)
….
polygon 1:
(0,0,0, 1,0,0, 1,0,1, 0,0,1)
polygon 2:
(1,0,0, 1,1,0, 1,1,1, 1,0,1)
```

In case the geometry is needed for rendering the box, then the coordinates of the faces can be used. For wire frame visualization it might be sufficient to draw lines using the edges. However, the query 'which is the common edge' will require first a triple coordinate comparison to find the common arc and second comparing the triple with the triple from the edges to identify the id of the edge.

4.3 Topological/Geometrical representation

Approach 3 still seems to be the best solution: it allows flexible topological and geometric database queries executed efficiently by accessing the topology and geometry model in the 3D Geo-DBMS. With these two models all relevant types of spatial database queries can be executed:

- Topological database queries (neighborhoods etc.);
- Metric database queries (distances);
- Geometric database queries (e.g. spatial search inside a 3D box).

In this respect, an important aspect of data management is the implementation of the spatial functionality, i.e. topologically based analyses or geometrically based analyses. Topologically based analyses are beneficial for all kind of operations making use of neighborhood relationships. Geometric based analyses will be necessary for object construction and to build the topology.

4.4 Implementation of the geometrical model in geo-DBMS

3D data types are currently not supported by DBMS, however one still can use the advantages of the other spatial data types (3D polygons). Since storage of 3D polygons is supported, 3D spatial objects can be defined as polyhedrons (body with flat faces).

4.4.1 Using 3D data types

Mainstream DBMSs (Oracle, Postgres, IBM, Ingres, Informix) manage 3D points, lines and polygons. Using 3D polygons, 3D objects can be represented as polyhedrons in two ways: as a list of data type *polygons* or as data type *multipolygon/collection*.

In the first option (defining a 3D objects as a list of 3D polygons) two tables are used: a table 'BODY' and a table 'FACE'. In the table 'BODY' the 3D spatial object is defined by a set of records containing an unique pointer to a faces closing the volume of a body. The table 'FACE' contains geometry of faces stored as 3D polygons. This model is partly a topological model, since the body is defined by references to the faces (faces share two bodies). The two tables follow:

```
create table BODY (
 bodyID number(38) not null,
 faceID number(38) not null
);
create table FACE (
 faceID number(38) not null,
 shape mdsys.sdo_geometry not null
);
```

To be able to obtain all the faces of a bodywith bodyID=1, the following SQL query has to be executed:

```
SQL> select body_id,face_id,shape from body b, face f where b.face_id=f.face_id
and b.body_id=1;

BODY_ID   FACE_ID

---------- ----------                -- one complex object (body_id=1)

SHAPE(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)

----------------------------------------------------------------------------
```

```
         1          1                -- Face 1 from Body 1

SDO_GEOMETRY(3003,                          -- 3003 indicates a 3D polygon
NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARRAY(
82220.96, 455098.11, -25, 82221.36, 455098.44, -25, 82232.88, 455106.96, -25,
82238.93, 455099.08, -25, 82242.01, 455101.41, -25, 82237.26, 455107.61, -25,
82247.5, 455115.01, -25, 82253.82, 455106.78, -25, 82256.67, 455108.99, -25,
……..
82220.96, 455098.11, -25))
         1          2                -- Face 2 from Body 1
SDO_GEOMETRY(3003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(82220.96,  455098.11,  -25,  82220.96,  455098.11,  -1,  82221.36,  455098.44,  -1,
82221.36, 455098.44, -25, 82220.96, 455098.11, -25))
Etc…..
```

4.4.2 Using multipolygon/collection

In the second representation a body is stored using data type *multipolygon/collection* as one record. In this case only one table is necessary, i.e. BODY. Note, that the type of geometry used is not visible in the *create* statement:

```
create table BODY (
 bodyID number(38) not null,
 shape mdsys.sdo_geometry not null
);
```

To obtain the geometry of the same body with bodyID=1, a much simple query has to be executed.

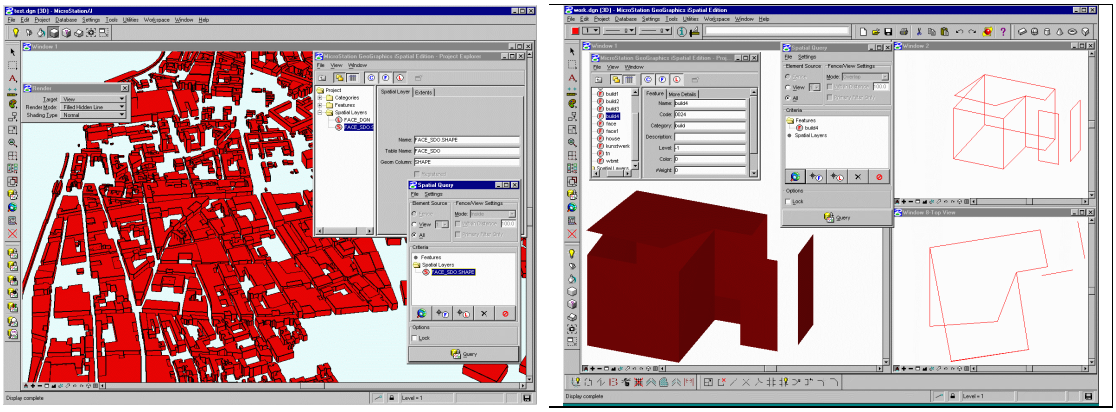
```
SQL> select body_id,shape from object3d_multipol where body_id=1;

BODY_ID SHAPE(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, ),SDO_ELEM_INFO,SDO_ORDINATES)
--------   ------------------------------------------------------------
         1
-- Body 1
SDO_GEOMETRY(3007,
-- 3007 indicates a 3D multipolygon
NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1, 109, 1003, 1, 124
-- the offset of the polygons is specified
, 1003, 1, 139, 1003, 1, 154, 1003, 1, 169, 1003, 1, 184, 1003, 1, 199, 1003, 1,
574, 1003, 1, 589, 1003, 1, 604, 1003, 1, 619, 1003, 1, 634, 1003, 1),

SDO_ORDINATE_ARRAY(
82220.96,  455098.11,  -25,  82221.36,  455098.44,  -25,  82232.88,  455106.96,  -25,
82238.93,  455099.08,  -25,  82242.01,  455101.41,  -25,  82237.26,  455107.61,  -25,
82247.5, 455115.01, -25, 82253.82, 455106.78, -25, 82256.67, 455108.99, -25,
……
82086.74, 455275.92, -25, 82117.4, 455236.1, -25, 82115.65, 455234.76, -25,
82136.77, 455207.36, -25, 82178.4, 455153.55, -25, 82220.96, 455098.11, -25,
-- end of 1st polygon
82220.96, 455098.11, -25, 82220.96, 455098.11, -1, 82221.36, 455098.44, -1,
82221.36,455098.44, -25, 82220.96, 455098.11, -25, 82221.36, 455098.44, -25,
--end of 2nd polygon
```

```
.........
etc.
```

An apparent advantage of the 3D multipolygon approach is the one-to-one correspondence between a record and an object. Furthermore the 3D multipolygon (compared to list of polygons) is that it is recognized as one object by front-end applications (GIS/CAD). For example, a 3D multipolygon is visualized as a 'group' of objects in Microstation Geographics. However, in case of editing the objects still has to be ungrouped into composing faces (figure 5).



5: Visualization of 3D objects: 3D polygon and 3D multipolygon

An additional advantage of the 3D multipolygon approach is the one-to-one correspondence between a record and an object. Disadvantage of these approaches is that the topology structure between objects cannot be used, which implies that there is redundant storage of edges (and in the 3D multipolygon solution also of faces).

4.5 Implementation the topological model in geo-DBMS

A user-implemented full topological model can also be defined in a DBMS. In this research we use the Simplified Spatial Model (Zlatanova, 2000). A 3D geometry object is therein defined as a polyhedron consisting of nodes and faces. The model consists of 3 tables: BODY, FACE and NODE (note that there is no edge table). Each table contains references to other tables: the BODY-table contains references to the faces and the FACE-table contains references to the nodes (with their co-ordinates).

A pl/sql script was written to generate the BODY, FACE and NODE table:

```
SQL> select * from body order by bid,seqf;

   BID        FID       SEQF
---------- ---------- ----------
     1          1          1
     1          2          2
     1          3          3
.........
     1         11         11
     1         12         12
     1         13         13
     2         13          1
```

```
    2         14          2
    2         15          3
etc……

SQL> select * from face order by fid,seqn;

FID        NID       SEQN
---------- ---------- ----------
    1          1          1
    1          2          2
    1          3          3
……
    1         11         11
    1          1         12
    2          1          1
    2         12          2
    2         13          3
    2          2          4
    2          1          5
……

SQL> select * from node;
NID                   XYZ_LIST
-----------            ---------------
1              XYZ_LIST_T(82220.96, 455098.11, -25)
XYZ_LIST_T(82221.36, 455098.44, -25)
XYZ_LIST_T(82232.88, 455106.96, -25)
……
```

Note that SEQF (sequence face) has no use (at all) because the order of faces within a body does not matter. Also note that faces are shared between bodies as it should be in a full topological model (e.g. face 13 is shared between body 1 and body 2) and nodes are shared between faces (e.g. node 2 is shared between face 1, 2 and 3).

Advantage of this approach is that topology structure management can be used in the storage and retrieval of the data. For example, by means of the shared (horizontal) faces one can easily find the neighbors of a spatial object. Disadvantages of using a topological model are:

- The data model needs three tables instead of just one (as in the 3D multipolygon case);
- Since the DBMS does not recognize topology, inserting the data requires a lot of effort and experience or other software: also the consistency of the data has to be checked by other software;
- Querying can be very difficult at SQL level (topology is not recognized by DBMSs). For geometric queries it is always required to generate a realization of the object, instead of being able to use the spatial queries available in the DBMS directly.

4.6 From topology to geometry in geo-DBMS

To realize the geometry of the 3D spatial objects, based on the topological tables, a function has been written. The nodes of one 3D spatial object are retrieved by the following query (Zlatanova and Verbree, 2000):

```
/* for the body bid=1*/

select body.bid,face.fid, face.seqn, node.nid, node.xyz_list
from body, face, node where body.bid=1;
```

After this, the obtained nodes are translated to either a complex object of 3D faces (3D polygons) or a 3D multipolygon of the object. Having the geometry it is possible to visualize/edit the data in GIS and CAD software and to perform spatial queries in the DBMS:

```
SQL>    select distinct parcel_number, municipality , osection
        from parcel p, object3d obj3d
        where      sdo_geom.relate(return_polygon(p.object_id),      'anyinteract',
        obj3d.shape,1) = 'TRUE';
```

5 Structuring 3D Data in GML3

Even though geo-DBMS' now are capable to store 3D-data within topological and geometrical data models, these data has to be exchanged. The neat way to do that is to use a kind of a direct access to the geo-DBMS, but as these are not capable to 3D data, file-exchange is still current practice.

For 3D-data GML 3.0 is not accepted as the default yet, but as GML is the standard to be used, it is worth full to consider the possibilities. We will give here a short summery of the structuring of surfaces, solids, topology and the symmetry between the geometry and topology in GML 3.0. We have cited this description from the book on GML by Ron Lake et al (Lake, 2004).

5.1 Surfaces, Surface Patches and Solids in GML3

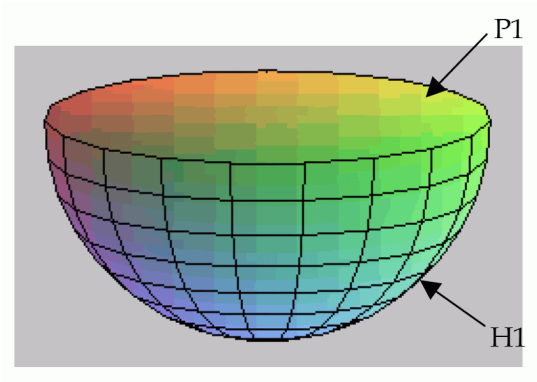


Figure 9: A solid bounded by a compositeSurface

Surfaces, Surface Patches and Solids Surfaces and solids are supported in GML3. The boundary of a `gml:Solid` is a `gml:CompositeSurface`, which consists of one or more surface members that enclose the Solid without overlap or open gaps. Figure 9 shows an example of a solid half-ball S1 having a composite surface as its boundary consisting of two surfaces, the lower hemisphere `H1` and the equatorial plane `P1`.

A `gml:Surface` such as the lower hemisphere `H1` is encoded as a collection of surface patches, where each patch member is usually a small (flat) polygon. In this case, the surface looks like the surface of a cut gem. However, a more accurate representation of H1 can be defined in an application schema using spherical surface interpolation. An example of this is given in the next section. The following example shows a sample encoding of the `Solid S1`. Note that a collection of small polygonal patches are used to represent the upper hemispherical `Surface H1`:

```
<gml:Solid gml:id="S1">
  <gml:exterior>
    <gml:CompositeSurface>
      <gml:surfaceMember>
        <gml:Surface gml:id="H1">
          <gml:patches>
            <gml:PolygonPatch>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:coordinates>
```

```
1.,0.,0. .95,.31,0 .94,.30,-.14 .94,0,-.14 1.,0.,0.
                  </gml:coordinates>
                </gml:LinearRing>
              </gml:exterior>
            </gml:PolygonPatch>
            <gml:PolygonPatch>
              ...
            </gml:PolygonPatch>
              ...
          </gml:patches>
        </gml:Surface>
      </gml:surfaceMember>
      <gml:surfaceMember>
        <gml:Surface gml:id="P1">
          <gml:patches>
            <gml:PolygonPatch>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:coordinates>
.95,  .31,  0,  .81  ,.58,0,  .60,.80,0,  .36,.93,0,  0.,1.,0,  -.36,.93,0,  -
.60,.80,0,  -.81,.58,0,  -.95,.31,0,  -1.,0.,0,  -.95,-.31,0,  -.81,-.58,0,  -
.60,-.80,0,  -.36,  -.93,0,  0.,-1.,0,  .36,-.93,0,  .60,-.80,0,  .81,-.58,0,
.95,-.31,0, 1.,0.,0.
                  </gml:coordinates>
                </gml:LinearRing>
              </gml:exterior>
            </gml:PolygonPatch>
          </gml:patches>
        </gml:Surface>
      </gml:surfaceMember>
    <gml:CompositeSurface>
  </gml:exterior>
</gml:Solid>
```

5.2 Topology in GML3

In GML, spatial topology is modeled using basic building blocks—nodes, edges, faces, and solids—called topology primitives, together with a description of their connective relationships to one another. The GML topology primitive types, `Node`, `Edge`, `Face`, and `TopoSolid`, are often used to represent the geometry primitives, `Point`, `Curve`, `Surface`, and `Solid`, respectively. The connectedness of nodes, coincidence of edges, and adjacency of faces and solids are some of the connective relationships between primitives that topology is concerned with. Unlike GML geometry, topology does not encode any coordinate values—so `<pos>` and `<coordinates>` tags are absent. The topology model is not concerned with the position of nodes, direction of edges, nor the shape of faces and solids.

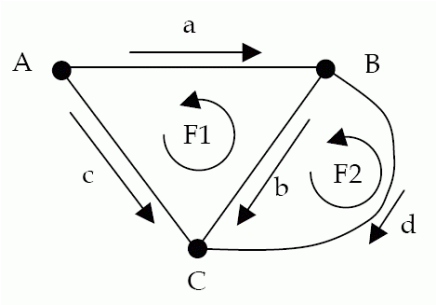


Figure 10: A simple topology network

Figure 10 shows an example of a simple topology model for a road network. This example has three `Nodes` (`A`, `B`, `C`), four `Edges` (`a`, `b`, `c`, `d`) and two `Faces` (`F1`, `F2`).

In GML 3.0, the nodes can be encoded as follows:

```
<gml:Node gml:id="A"/>
<gml:Node gml:id="B"/>
<gml:Node gml:id="C"/>
```

The `Nodes` are only required to be uniquely identified using the `gml:id` attribute. The Edges can be encoded as follows:

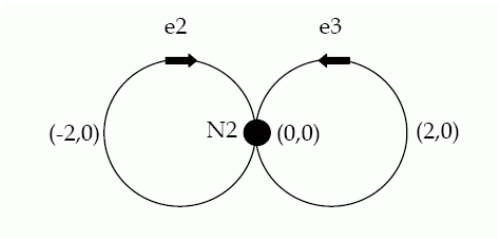
```
<gml:Edge gml:id="a">
    <gml:directedNode orientation="-" xlink:href="#A"/>
    <gml:directedNode orientation="+" xlink:href="#B"/>
</gml:Edge>
<gml:Edge gml:id="b">
    <gml:directedNode orientation="-" xlink:href="#B"/>
    <gml:directedNode orientation="+" xlink:href="#C"/>
</gml:Edge>
<gml:Edge gml:id="c">
    <gml:directedNode orientation="-" xlink:href="#A"/>
    <gml:directedNode orientation="+" xlink:href="#C"/>
</gml:Edge>
<gml:Edge gml:id="d">
    <gml:directedNode orientation="-" xlink:href="#B"/>
    <gml:directedNode orientation="+" xlink:href="#C"/>
</gml:Edge>
```

In the GML topology model, each face is defined by its boundary, which consists of a list of directed edges. The directed edges in the boundary of each face are traversed in a counter-clockwise direction as indicated by the arrow surrounding `F1` and `F2` in Figure 3-4. The orientation of each directed edge in the boundary of a face is either "+" or "-", depending on whether the inherent direction of the edge agrees or disagrees with the counter-clockwise orientation of the face. For example, the boundary of the `Face` labelled `F1`, when traversed counter-clockwise, corresponds to the directed edges in the set {c,-b,-a}. The faces are encoded in GML 3.0 as follows:

```
<gml:Face gml:id="F1">
  <gml:directedEdge orientation="+" xlink:href="#c"/>
  <gml:directedEdge orientation="-" xlink:href="#b"/>
  <gml:directedEdge orientation="-" xlink:href="#a"/>
</gml:Face>
  <gml:Face gml:id="F2">
  <gml:directedEdge orientation="+" xlink:href="#b"/>
  <gml:directedEdge orientation="-" xlink:href="#d"/>
</gml:Face>
```

5.3 Symmetry between Topology and Geometry in GML3

In GML 3.0, there is symmetry between the geometric and topological primitives. The geometry primitives—`Point`, `Curve`, `Surface`, and `Solid`—are often referred to as geometric realizations of—`Node`, `Edge`, `Face`, and `TopoSolid`, respectively. These geometric realizations are expressed using the following properties: `pointProperty`, `curveProperty`, `surfaceProperty` and `solidProperty`.



For example, suppose coordinates are assigned to the following simple topology network in Figure 11 as shown.

Figure 11: Symmetry topology and geometry

The simple topology network can be encoded including the geometric information as follows:

```
<gml:Node gml:id="N2">
  <gml:pointProperty>
    <gml:Point>
      <gml:coordinates>
        0,0
      </gml:coordinates>
    </gml:Point>
  </gml:pointProperty>
</gml:Node>

<gml:Edge gml:id="e2">
  <gml:directedNode orientation="-" xlink:href="#N2"/>
  <gml:directedNode orientation="+" xlink:href="#N2"/>
  <gml:curveProperty>
    <gml:Curve>
      <gml:segments>
        <gml:Circle>
          <gml:coordinates>
            0,0 -2,0 -1,1
          </gml:coordinates>
        </gml:Circle>
      </gml:segments>
    </gml:Curve>
  </gml:curveProperty>
</gml:Edge>

<gml:Edge gml:id="e3">
  <gml:directedNode orientation="-" xlink:href="#N2"/>
  <gml:directedNode orientation="+" xlink:href="#N2"/>
  <gml:curveProperty>
    <gml:Curve>
      <gml:segments>
        <gml:Circle>
          <gml:coordinates>
            0,0 2,0 1,1
          </gml:coordinates>
        </gml:Circle>
      </gml:segments>
    </gml:Curve>
  </gml:curveProperty>
</gml:Edge>
```

6 Other geometrical and topological 3D-data formats

6.1 Pure Geometrical 3D-data formats

Within geometrical 3D-data formats each feature is basically described by a list of coordinates.

An example of this kind of storage is the ESRI shapefile. Within the header of this file the type of the features is given, i.e. pointZ, polylineZ or polygonZ. The 'Z' denotes that the points and the vertices of the polylines and polygons have a Z-value attached to them.

The geometry given by the shapefile could define a boundary of a 3D-feature, but no guarantee is given this boundary is 'watertight'. This validation has to be performed by the logic of the program using this shapefile.

Other drawbacks of pure geometrical 3D-data formats are duplication of coordinates, no standardized underlying data model, and lots of proprietary implementations.

6.2 Geometrical / Topological 3D-data formats

As described in section 4 the topological 3D-dataformats has a separate section, or even a separate file, to store the coordinates of the points and a separate section to store the topological relationships. As most 3D-boundary representations are limited to triangular facets, this topology can be quite simple. We will illustrate these structures by a few well-known examples.

6.2.1 PLY files

The PLY file format is a simple object description that was designed as a convenient format for researchers who work with polygonal models. Early versions of this file format were used at Stanford University and at UNC Chapel Hill.

A PLY file consists of a header followed by a list of vertices and then a list of polygons. The header specifies how many vertices and polygons are in the file, and also states what properties are associated with each vertex, such as (x,y,z) coordinates, normals and color. The polygon faces are simply lists of indices into the vertex list, and each face begins with a count of the number of elements in each list. In most cases this number will be equal to 3, as the models are given as 3D-TIN surfaces.

```
ply
format ascii 1.0
element vertex 1177                 # total vertices = 1177
property float32 x
property float32 y
property float32 z
element face 2256                   # total faces = 2256
```

```
property list uint8 int32 vertex_indices
end_header
1.38137 0 2.45469                    #list of coordinates
1.4 0 2.4
1.35074 -0.375926 2.4
1.33276 -0.370922 2.45469
..
..
3 0 1 2                              # list of faces
3 0 2 3
3 4 0 3
3 4 3 5
..
```

Lots of other file-based structures are similar to this approach, i.e. VRML.

6.3 VRML / X3D

VRML (Virtual Reality Modeling Language) respectively its successor X3D (Extensible 3D) were introduced by the Web3D Consortium to distribute interactive virtual worlds on the web. Both are mark-up languages and standardized, whereby X3D is fulfilling the concepts of XML. Besides X3D is specified more modular. The rendering concept is mainly based on a scene graph definition and a node structure (Web3D Consortium, 2004).

The most common use of VRML is within a client-side browser/plug-in implementation. Unfortunately plug-in vendors are hesitating with shipping X3D browsers.

7 Conclusions and recommendations

As shown in this report the choices made in modeling, Geo-DBMS structuring, file exchange and applications of 3D-data within geo-information technology do have its influence on each other. First of all it should be noted that we have restricted ourselves to the so-called boundary representations of real-world objects, and we discussed properties of several 3D topological models.

Among them, we give preferences to the 3D-Triangulated Irregular Networks (3D-TIN). This kind of representation operates with the simplest primitives (triangles) ensuring planarity and simple consistency check operations. It is easy to validate an object and it is relatively straightforward to represent the interior of the object as a Tetrahedronized Irregular Network (TEN).

Note, the 3D-TIN representation for the boundary of an object and the TEN representation for the interior are duals. This means it is possible to obtain the interior by its boundary and vise versa. If the interior of the objects is structured within a TEN, the boundary could be derived with ease and structured as a 3D-TIN. This 3D-TIN/TEN approach combines the advantages of geometrical and topological modeling.

Modeling is one part of the problem, structuring the model in a geo-DBMS is the second. As geo-DBMS's are now more and more used to share geo-data, it should be made possible to store both the 3D geometry and 3D topology. The current geo-DBMS's are capable to store 3D-geometries as a list of 3D polygons or within a multipolygon. Structuring a TEN with this kind of geometries is quite simple, as each tetrahedron consists out of 4 triangular facets. This will however result in a huge storage capacity due to repetition of coordinates.

A better way to store a TEN within a geo-DBMS is to opt for a topological implementation, where the faces and nodes (and if desired the edges) are structured according the underlying datamodel. If necessary we can materialize the faces, edges and nodes to their corresponding polygons, lines and points. Further tests are needed to investigate which of the two implementations (geometrical or topological) will perform better. Since the current geo-DBMS do not natively support 3D objects, validation operations have to be developed in both implementations.

Besides the validation functions, a set of geometrical and topological operators and functions have to be developed to employ the specific advantages of structuring the data according the 3D-TIN boundary and TEN interior approach. This could lead to a special 3D-TIN/TEN GIS where the operations (visualization, area and volume calculations, topological relationships, spatial analysis, etc) are performed on top of the TIN/TEN datamodel.

The advantage of 3D-TIN/TEN modeling is its similarity with the data-exchange common-known 3D-file formats, like .ply and .vrml, where a list of the coordinates of the nodes is followed by a face-list. For a more standardized and structured way of 3D-Data exchange one could opt to store 3D-TINs and TEN data-files within

GML3, as this is the exchange format for geo-data and it gives the capabilities to store both the topology as the geometry.

Storing data within a file however is not recommended. It will result in a loss of control in the validness of the data. When this data is read in the geo-database according the underlying data model we should validate the input according to both the structure of the data-model and the presumed characteristics (rules) - like 'watertight' - of the 3D-TIN and TEN structures.

References

Carlson, E.(1987): Three dimensional conceptual modelling of subsurface structures, Technical Papers of ASPRS/ACSM Annual Convention, Baltimore, Vol. 4 (Cartography), pp. 188-200.

Coors, V., 2002. Resource-adaptive 3D maps for LBS. Proceedings of the 23rd Urban and Regional Data Management Symposium '30 years of UDMS - looking back, looking forward'. Prague, Czech Republic, pp. I.29-I.

Lake, Ron, David. S Burggraf, Milan Trninic and Laurie Rae. (2004). Geography Markup Language (GML), Wiley, ISBN 0-470-87153-9

Möller, Tomas. (1997). "A fast triangle-triangle intersection test", Journal of Graphics Tools, 2(2)

Oosterom, P.J.M. van, C.W. Quak and T.P.M. Tijssen. (2003). Polygons: the unstable foundation of spatial modeling, ISPRS Joint Workshop on "Spatial, Temporal and Multi-Dimensional Data Modeling and Analysis", Québec, Canada.

Oosterom, P.J.M. van, Wilko Quak, and Theo Tijssen. (2004). About Invalid, Valid and Clean Polygons, Developments in Spatial Data Handling, Fisher, Peter F. (Ed.), pp: 1-16.

Open Geospatial Consortium (2004): available at : http://www.opengeospatial.org/

O'Rourke J. (1998). Computational Geometry in C (Second Edition), Cambridge University Press, Cambridge

Pilouk, M. (1996). Integrated modeling for 3D GIS, PhD thesis, ITC, the Netherlands

Stoter, J.E. (2003). 3D Cadastre. PhD thesis, Delft University of Technology, Nederlandse Commissie voor Geodesie (NCG).

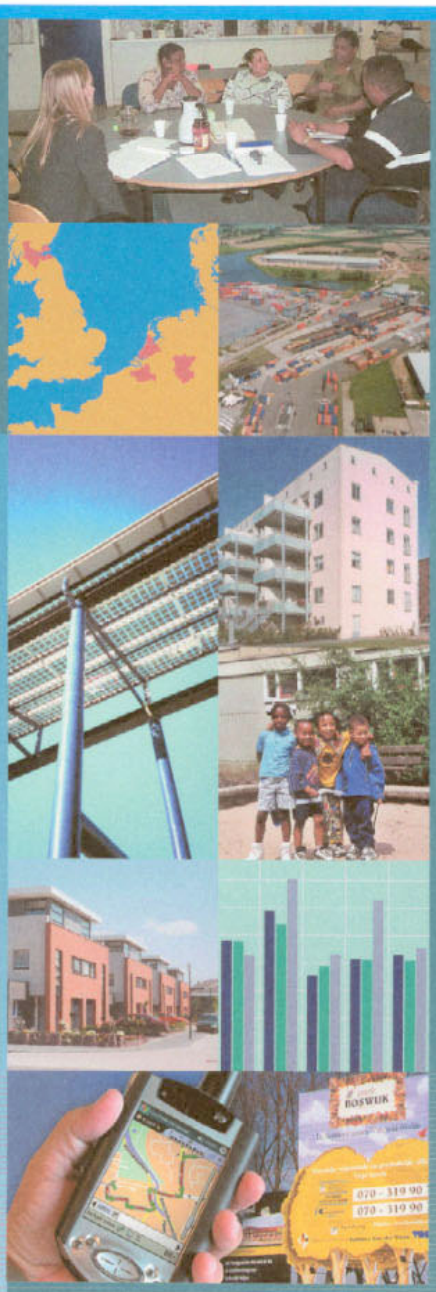
TetGen. (2005): Available at: http://tetgen.berlios.de/

Web3D (2005): Available at: http://www.web3d.org

Zlatanova, S. and E. Verbree (2000): A 3D topological model for augmented reality , in: Proceedings of the Second International Symposium on Mobile Multimedia Systems & Applications, 9-10 November, Delft, The Netherlands, pp.19-26.

Zlatanova, S. (2000): 3D GIS for urban development, PhD thesis, ITC publication 69, Enschede, the Netherlands, 222 p.

Zlatanova, S. (2004). Workshop on 3D-City Modeling, UDMS



OTB Research Institute for Housing, Urban and Mobility Studies
Delft University of Technology
Jaffalaan 9, 2628 BX Delft, The Netherlands
Postbus 5030, 2600 GA Delft, The Netherlands
Telefoon +31 (0)15 278 30 05
Fax +31 (0)15 278 44 22
E-mail mailbox@otb.tudelft.nl
www.otb.tudelft.nl