Petra Pokrovac ◆ MASTER THESIS ◆ 2021.

UNIVERSITY OF ZAGREB

FACULTY OF GEODESY

Petra Pokrovac

COMPARISON OF DATA ENCODINGS TO DELIVER 3D TERRAIN MODELS AS TRIANGULATED IRREGULAR NETWORKS

Master thesis

Zagreb, 2021.

UNIVERSITY OF ZAGREB

FACULTY OF GEODESY

Petra Pokrovac

COMPARISON OF DATA ENCODINGS TO DELIVER 3D TERRAIN MODELS AS TRIANGULATED IRREGULAR NETWORKS

Master thesis

Zagreb, 2021

UNIVERSITY OF ZAGREB

FACULTY OF GEODESY

Na temelju članka 19. Etičkog kodeksa Sveučilišta u Zagrebu i Odluke br. 1_349_11 Fakultetskog vijeća Geodetskog fakulteta Sveučilišta u Zagrebu, od 26.10.2017. godine (klasa: 643-03/16-07/03), uređena je obaveza davanja „Izjave o izvornosti“ diplomskog rada koji se vrednuju na diplomskom studiju geodezije i geoinformatike, a u svrhu potvrđivanja da je rad izvorni rezultat rada studenata te da taj rad ne sadržava druge izvore osim onih koji su u njima navedeni.

IZJAVLJUJEM

Ja, **Petra Pokrovac**, (JMBAG: 0007179656), rođen/a dana 15.09.1997. u Šibeniku, izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi tog rada nisam koristio drugim izvorima osim onih koji su u njemu navedeni.

U Zagrebu, dana ____________ ______________________

Potpis studenta / studentice

I. AUTHOR	
Name and surname:	Petra Pokrovac
Date and place of birth:	September 15th 1997, Šibenik, Republic of Croatia
II. MASTER THESIS	
Title:	Comparison of Data Encodings to Deliver 3D Terrain Models as Triangulated Irregular Networks
Number of pages:	81
Number of tables:	5
Number of figures:	47
Number of bibliographic data:	35 + 45 URLs
Institutions and places where the work was made:	Hochschule Bochum, Bochum Faculty of Geodesy, Zagreb
Supervisors:	doc. dr. sc. Vesna Poslončec-Petrić prof. dr. rer. nat. Benno Schmidt
Research mentor:	mag. ing. geod. et. geoinf. Iva Cibilić
III. GRADE AND DEFENCE	
Date of theme assignment:	February 12th 2021
Date of thesis defence:	Septemper 17th 2021
Defence committee members:	doc. dr. sc. Vesna Poslončec-Petrić
	doc. dr. sc. Andrija Krtalić
	izv. prof. dr. sc. Robert Župan

Acknowledgments

Throughout of writing of this thesis I have recived a great deal of support and assistence.

I would first like to thank my supervisor doc. dr. sc. Vesna Poslončec-Petrić for her guidance and support during the creation of this thesis. I am also thankful to research mentor mag. ing. geod. et. geoinf. Iva Cibilić for her understanding, advices and help in the creation of my thesis.

I would also like to thank to prof. dr. rer. nat. Benno Schmidt from Hochschule Bochum in Germany for his assistance and patience. His expertise was invalualbe in formulating the research questions and metodology, and also his feedback brought my work to a higher level.

In addition, I would like to acknowledge my family and friends on unconditional support and understanding throughout all the years of my study.

Comparison of Data Encodings to Deliver 3D Terrain Models as Triangulated Irregular Networks

Abstract: *Models describing the Earth surface or other surfaces related to it, such as underwater terrain, pedologic and geologic layers, are often given as triangulated irregular networks (TINs). Such models are used in numerous technological contexts such as GIS, CAD, BIM, 3D computer graphics, scientific visualization and others. The problem arises when the exchange of TIN data between applications is required, because there is no generally accepted standard. Different disciplines favour their formats and data encodings, so GIS applications primarily use GML formats, IFC is used in the BIM context, DXF in the CAD world, while VRML/X3D is mostly used in the 3D graphics community. This paper aims to investigate how formats are conceptually different, what semantic differences exist between the required formats, what information can be stored in which format and which format is most suitable for a particular task depending on the final purpose of application. It also deals with the issue of mutual format conversion and the loss of semantic information in that process.*

Keywords: *3D file formats, format conversion, semantic differenece, triangulated irregular network*

Usporedba podataka za kodiranje za isporuku 3D modela terena kao nepravilne mreže trokuta

Sažetak: *Modeli koji se koriste za opisivanje Zemljine površine ili površine povezane s njom, kao što su podvodni tereni, pedološki i geološki slojevi, često su prikazani kao nepravilne mreže trokuta (TIN). Takvi se modeli koriste u brojnim tehnološkim područjima kao što su GIS, CAD, BIM, 3D računalna grafika, znanstvena vizualizacija i drugo. Problem nastaje kada je potrebna razmjena TIN podataka između aplikacija jer ne postoji općeprihvaćen standard. Različite discipline favoriziraju vlastite formate i kodiranje podataka, pa tako GIS aplikacije primarno koriste GML formate, IFC se koristi u BIM kontekstu, DXF u CAD svijetu, dok se VRML/X3D najviše koriste u 3D grafičkoj zajednici. Cilj ovog rada je istražiti kako se formati konceptualno razliku, koje semantičke razlike postoje između traženih formata, koje informacije mogu biti pohranjene u koji format te koji je format najprikladniji za određeni zadatak ovisno o konačnoj svrsi primjene. Također cilj rada je istražiti međusobne pretvorbe formata te gubitke semantičkih informacija u tom procesu.*

Ključne riječi: *3D formati podataka, nepravilna mreža trokuta, semantičke razlike, pretvorba formata*

TABLE OF CONTENTS

1. INTRODUCTION

Triangulated irregular network (TIN) represents a surface as a set of non-overlapping contiguous triangular facets of irregular sizes and shapes. This network is widely used in geographical information systems (GIS) and other related fields to represent topography and terrain (Vivoni, 2003). Models that describe the surface of the Earth or other surfaces related to it such as underwater terrain, pedologic or geologic layers are often given as triangulated irregular networks. In addition to GIS, numerous other technological contexts such as BIM, CAD, 3D computer graphics, scientific visualization, gaming applications and others use these models.

The problem arises when the exchange of TIN data between applications is required, because there is no generally accepted standard. Different disciplines favuor their formats and data encodings, so GIS applications primarily use GML formats, IFC is used in the BIM context, DXF in the CAD world, while VRML/X3D is mostly used in the 3D graphics community. This paper deals precisely with this problem.

The second chapter of the paper lists the basic concepts of terrain and surface models, elements for representing the Earth's surface and the structure of digital models. In the third chapter triangular irregular network is described. The fourth chapter deals with the 3D format in a general way. In the fifth chapter, each 3D file format is described in detail separately. The sixth chapter deals with the comparison of 3D file formats based on a number of criteria. In the seventh chapter, the topic of mutual conversion of formats is elaborated, whether the conversion is possible and whether there are losses and which, if it is possible.

2. 3D TERRAIN MODELS

The digital terrain model is a static presentation of the continuous ground surface by an outsized number of selected points with known X, Y, Z coordinates in an arbitrary reference frame.

The basic concepts of terrain and surface models are defined (Krtalić et al., 2019 stated in Maune, 2001):

- as Digital Elevation Model (DEM) – a term that contains digital topographic data in various forms,
- as Digital Terrain Model (DTM) – synonymous for a digital elevation model of a scene where is only bare ground (natural relief),
- as Digital Surface Model (DSM) – show's the elevations of the tops of reflective surfaces, such as the Earth's surface (artificial objects) and vegetation.

In addition to these models, there is also normalized Digital Surface Model (nDSM), which is defined as the difference between DSM and DTM (Figure 2.1). nDSM indicates objects in relative relation to the ground (artificial objects and vegetation such as trees) or objects on the ground (flat areas, roads, etc.).

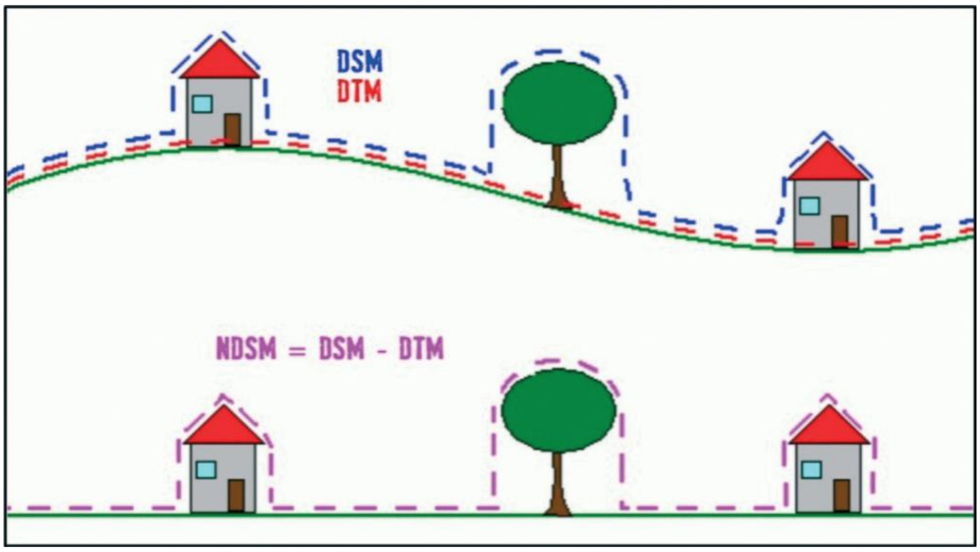


Figure 2.1. DSM, DTM, nDSM and their differences (Krtalić et al., 2019)

The most important specific elements for representing the Earth's surface are:

- pits – local minima (eg valleys or sinkholes),
- peaks – local maxima (hills and mountain peaks),
- ridge lines – lines that connect local maxima in cross section,
- course lines – lines that connect local minima in cross section (river valleys, flow lines),
- break lines – places where the change of slope is emphasized,
- crossings, saddles – crossing points of watersheds,
- contours – lines of equal height above the reference level (level of the sea or geoid),

- slope lines – falling gravitational flow lines, under right angle to the contours,
- planes – areas of relief where all altitude values are equal.

2.1 Input data

Methods and measurement techniques for collecting input data on Earth surface and for the aim of making its digital model can be divided on:

- vectorization of topographic maps (isohypses, elevations),
- field measurement for the collection of position and altitude data (tachymetry, GNSS),
- photogrammetric method of measurement (terrestrial, aerial, satellite),
- radar data collection methods,
- collect and process LiDAR data.

Each of the above methods of data collection has its aim and purpose in form of required accuracy, accessibility of instruments and available funds, on which their choice also depends.

2.2 Digital model structure

The structure of digital models is most often divided into two groups:

- Triangulated Irregular Network (TIN),
- GRID – regular grid of points (Figure 2.2.).

The TIN structure of a digital model is vector and implies an explicit definition of topology, i.e. the use of edges and nodes to determine spatial relationships between individual triangles in a grid. On the other hand, GRID has a raster structure of data, which interpolate values between known points collected at regular intervals, which also defines the spatial resolution of future DMR that is the minimum area for which the developed model can provide a result. Both models have their advantages and disadvantages, but in this thesis attention will be only on the TIN model.

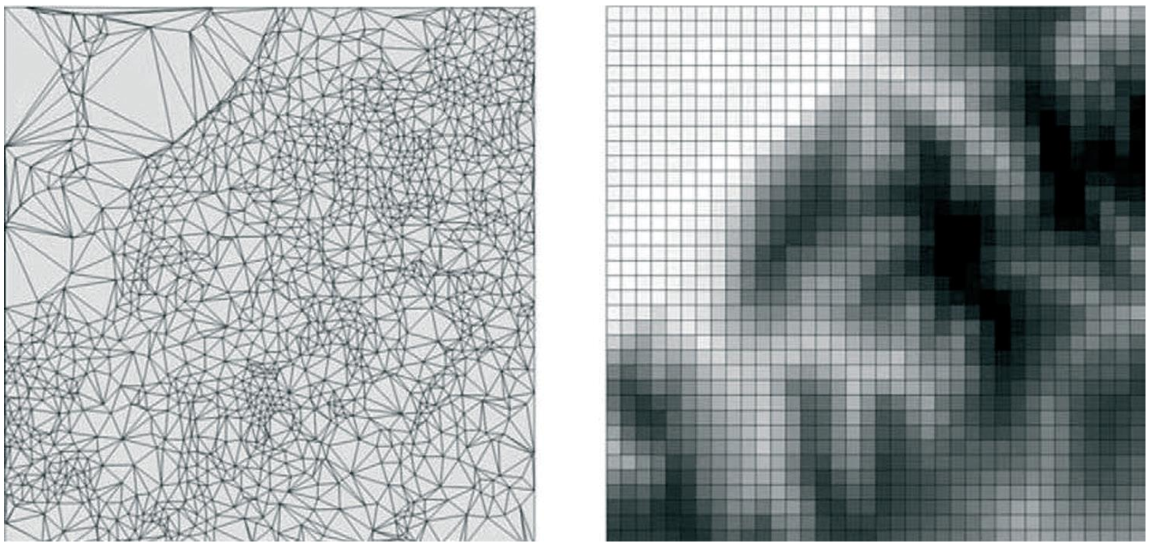


Figure 2.2. TIN structure of DEM (left) and GRID structure of DEM (right) (Krtalić et al., 2019)

3. TRIANGULATED IRREGULAR NETWORK (TIN)

Triangulated Irregular Network (TIN) is used in a Geographic Information System (GIS) for the representation of a surface. It is a representation of land surface and sea bottom, made up of irregularly distributed nodes (vertices) and lines with three-dimensional coordinates (x, y, z) that are arranged in a network of triangles that are not overlapping and do not share common intersection points. Points in TIN are distributed variably based on an algorithm that determines which points are most necessary to an accurate terrain representation. Because of that, data input is flexible and fewer points need to be stored than in a raster-based network. Also, TINs have disadvantages and are not convenient for every project. TINs requires a lot of processing time because of complex structure and high memory requirements. One more disadvantage is, because TINs are linear, many edges will appear jagged, distorting the image. Also, because of the irregularity of the TIN, the organization, storage, and application of its data are more complicated than that of the regular grid DEM (URL 2).

TIN represents a topographic elevation surface with elevations at corners of triangles. In the model, in addition to the elevations, are also stored the planimetric position and the topological relationship between nearby triangles. In regions where is little variation in surface height, the points can be widely spaced, while in areas with more intense variation in height density is increased.

The vertices are connected with a series of edges to make a triangle network. There are different methods of interpolation of triangles, such as Delaunay triangulation, distance ordering, simulated annealing algorithm and radiation scanning algorithm (URL 3).

The most commonly used triangulation in practice is Delaunay triangulation (Figure 3.1.). It is unique, because in Delaunay triangulation there will be no other points in the circumcircle of any triangle. A triangle is made up of the three nearest points and each segment of the line does not intersect. If the diagonals of the convex quadrilateral formed by any two nearby triangles are interchangeable, then the smallest angle between the six internal angles of the two triangles will not become larger. If the smallest angle of each triangle in the triangulation is ordered ascending, the organization of this triangulation will receive the largest value. Moving, adding or deleting a vertex will only have an impact on the nearby triangle.The result is that long, thin triangles are avoided the maximum amount as possible, triangles are as equi-angular as possible (URL 4).

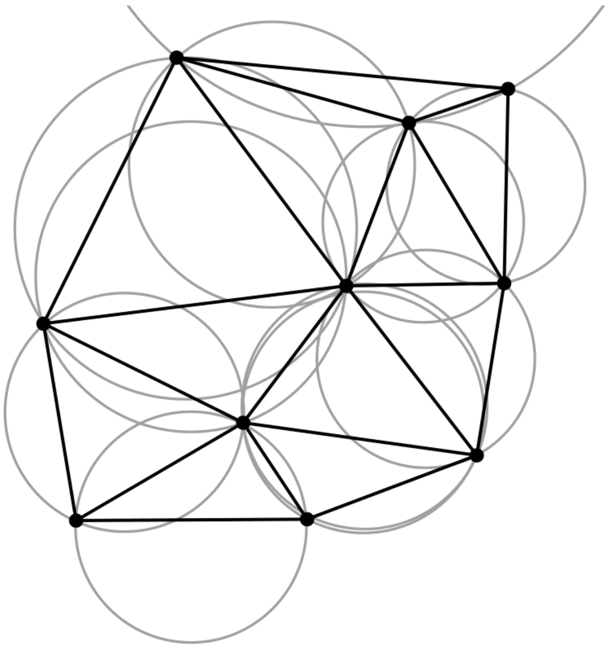


Figure 3.1. Delaunay triangulation (URL 5)

TINs are generated from points, polygons and lines. Mass points are points that are used in defining the TIN. Areas with constant elevation, such as water surfaces are called exclusion polygons. Breaklines are lines such as shorelines and streams. Breaklines can be soft or hard. Soft features are, for instance, ridgelines on rolling hills. Hard features are roads, streams and shorelines which indicate a significant break in slope. When a TIN is formed, mass points become nodes, breaklines and exclusion polygon boundaries become triangle edges (URL 6).

4. 3D FILE FORMATS

3D file formats collect extensive details about 3D models with their attributes. These formats have large applications in different fields such as video gaming, film, 3D printing, academic projects, engineering visualization, architectural projections and numerous scientific applications. A 3D file format encodes geometric information of a model and creates it readable on computers. It is possible to store texture and material information of the model apart or within it (URL 7).

4.1 Geometry

The geometry of 3D model is a collection of points, lines and planes, and they are called vertices, edges and polygons (or faces). 3D geometries are composed of more than thousands of triangles or polygonal meshes. A 3D file format compresses this information and makes it machine-readable (URL 8).

There are three different ways of encoding surface geometry. They are called approximate mesh, precise mesh and constructive solid geometry (URL 9).

4.1.1 The approximate mesh

In this encoding, the surface is first covered with mesh that contains tiny imaginary polygons. Triangles are most usually used shape. The vertices of the triangles and the outward normal vector of the triangles are saved in the file. This represents the geometry of the surface of the target model. The process of covering a surface with a geometric shape that is not overlapping is called „tessellation“. Because of that, these file formats are also known as tessellated formats. The triangles approximate the smooth geometry of the surface. Therefore, this is called approximate format. The approximation is better the smaller the triangles are, but implies that the file needs to save more number of vertices and normal vectors.

4.1.2 The precise mesh

There are situations where an approximate mesh is not enough and one needs precise encoding of the surface geometry. In the precise file formats, polygons are not used. Instead of them, Non-Uniform Rational B-Spline patches (NURBS) is used. These parametric surfaces are created of a small number of weighted control points and a set of parameters called knots. From knots, a surface can be computed mathematically by smoothly interpolating over the control points. While the precise mesh is accurate at any resolution, they are displayed more slowly and should be avoided in applications where speedy rendering is important.

4.1.3 Constructive solid geometry

This type of file format does not include meshes at all. In this format, 3D shapes are built by performing Boolean operations (addition or subtraction) of primitive shapes such as cubes, spheres etc. Constructive solid geometry is excellent for designing 3D models and is user-friendly.

4.2 Appearance

Appearance means textures and materials. Textures and materials are applied to 3D geometries to add color, character and improve the reality of 3D objects. This is used to make feelings of smoothness, sheen, shadows and reflectivity. In 3D models, textures and materials can be stored individually or composed into bundles. They are typically stored as bitmaps (images) or procedural textures (mathematical formulas) and can be referenced within a 3D file format or stored separately for external reference.

4.3 Scene

The scene provides information about cameras, light sources and other important objects. The camera is defined with camera properties such as four parameters for magnification and principal point, the 3D position of the camera, a 3D vector indicating the direction it is pointing, and another 3D vector indicating the up direction. A 3D file format can reference or store information about the scene in which 3D objects appears. This data contains the encoding of light source infomation such as intensity and location (McHenry and Bajcsy, 2008).

4.4 Proprietary or Open Source

Proprietary formats are designated as trade secrets and created by companies to force to buy particular hardware or software and limit interoperability. Before closed operating systems such as iOS and Windows existed, there was a tradition in software development to contribute innovations in software engineering to the public domain. Today, open-source 3D file formats are made for data and file portability across devices and applications. An open-source format exists in the public domain and can be used for proprietary, free and open-source software by following a standard licensing agreement. Commonly, there is no charge or licensing fee for using open-source software standards. Converting proprietary formats into open-source formats is hard because the encoding format of data is often locked to prevent reverse engineering.

4.5 The ability to display „true 3D model“ or „2.5D model“

Some 3D file formats have the suitability to give a true 3D model, but some of them do not. The second one is the 2.5D model. The main difference between these two models is that 2.5D is typically a 2D representation changed in some way to present itself as a 3D illusion. 2.5D generally does not have depth perspective in Z, but rather is a trick to simulate depth perspective in XY. 2.5D effect can be rendered by using shadows applied to a 2D objects (URL 14).

2.5D model stores only one elevation value (z) for any (x,y) location. Topologically, the surface shown by TIN is a 2-manifold. This means that each edge of the TIN on only one or two triangles and triangles that are falling on the vertices form a closed or open fan (Figure 4.1).

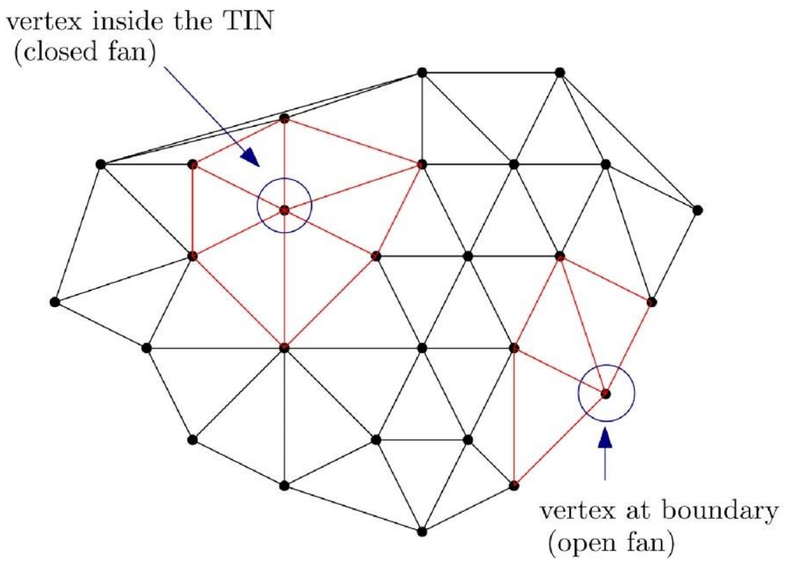


Figure 4.1. 2-Manifold TIN (URL 18)

However, some features are not possible to represent in 2.5D model, such as vertical walls, balconies, tunnels, etc., because, it would result in data loss. A model called 2.5D+ has more than one value (z) for the same x,y value and serves for modeling vertical walls of natural or artificial objects. It is still a 2-manifold model. 2.75D model is a 2.5D+ model but extended to model any surface with 2-manifold with features as balconies and overhangs.

Mentioned terrain representations provide the geometrical model of a terrain and do not include explicit representation of individual terrain features such as land use, buildings, roads and water bodies. An overview of terrain features is needed to support semantic queries about those features. In order to identify these individual terrain features, we must define them as discrete objects and provide their characteristics and relationships with other features explicitly through semantics. From an object perspective, terrain can be seen as a container inhabited by these objects, each with an identity, spatial embedding and attributes. This terrain representation is called 3D model.

Figure 4.2. shows different types of terrain representations.

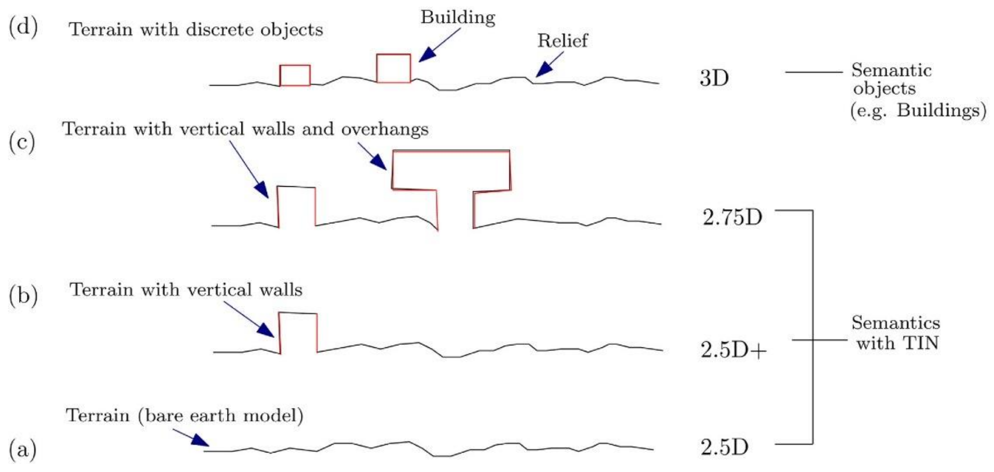

(d) Terrain with discrete objects
Building
Relief
3D
Semantic objects (e.g. Buildings)
(c) Terrain with vertical walls and overhangs
2.75D
(b) Terrain with vertical walls
2.5D+
Semantics with TIN
Terrain (bare earth model)
(a)
2.5D

Figure 4.2. Different TIN representations for modelling terrains (URL 18)

4.6 Level of Detail (LOD)

Several LOD concepts differ from each other.

In computer graphics, the term level of detail presents the complexity of displaying a 3D model. The level of displayed details can be reduced when the model moves away from the viewer. However, it can be changed according to other parameters, such as object importance, speed or viewpoint position. LOD reduces the number of vertices that need to be processed and avoid the problem of micro triangles (URL 13). The Figure 4.3. shows how the number of triangles increases with respect to the number of details shown.

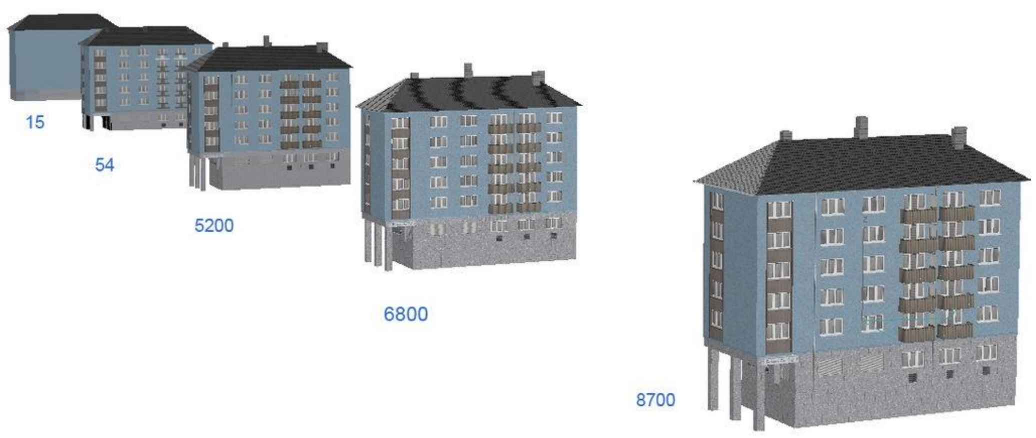


Figure 4.3. Representation of number of triangles in different LODs (URL 13)

In CityGML, the term LOD refers to a predefined, unified, scale-independent specification of the level in which architectural details of buildings are given. The levels of detail that can be displayed are described below (Figure 4.4):

1. LOD0 is the coarsest level for a digital terrain model. LOD0 defines a 2.5D representation of features. A 2.5D geometry describes by polygons built-in in 3D space where for each (x,y) coordinate there is at most one height value on the polygon (i.e., vertical polygons are excluded). Volume objects such as buildings are represented by a single horizontal polygon.
2. LOD 1 is a block model, building block (raised area). In this model, volume objects such as buildings and vegetation are modelled in a generalized manner as prismatic models of blocks with vertical walls and horizontal „roofs“. Objects such as roads and water surfaces are still defined as 2.5D surfaces.
3. LOD 2 is a 3D model of external shell, roof structures and simple textures. When it comes to buildings, this model also shows balconies and dormers. Vegetation objects are shown with more details. Water surfaces may differ thematically and roads are still represented as 2.5D, but transport objects have been improved by explicit modelling of traffic areas and ancillary traffic areas.
4. LOD 3 is an architectural model, a 3D model of the external shell with texture. This model is the most detailed level for the farthest shape of objects. Roads and land use

objects are represented as very detailed 2.5D surfaces. Water surfaces and building objects are also shown in a very detailed way.

5. LOD 4 is an interior model. It is a 3D model of the building with interior structures such as rooms, floors and others. The resolution of textures, which can be mapped to almost any feature, also increases with higher LOD (Groger and Plumer, 2012).

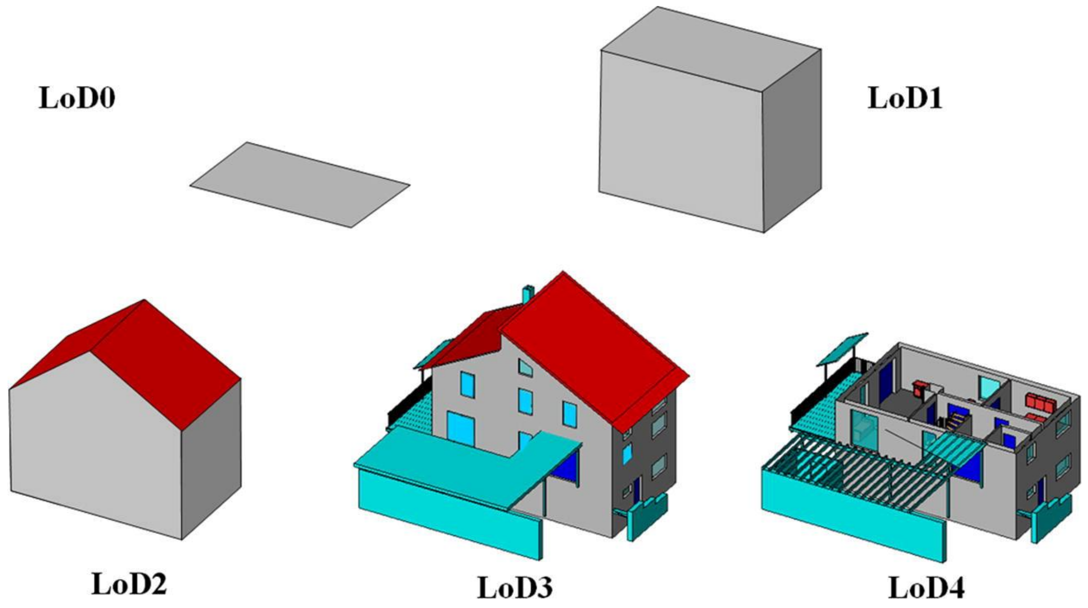


Figure 4.4. Showing different levels of detail on the example of a building (Groger and Plumer, 2012)

In Building Information Modeling (BIM) applications, there is another term of LOD. In this case, the LOD gives the percentage amount of information that a BIM element contains.

4.7 Support of TIN tiling concept

Tiling TIN data structure aims to consider the most targeted part of the TIN that will have a quite small number of objects (triangles and/or vertices) instead of taking into account the whole TIN. The TIN area will be divided between several tiles. Therefore the division will be up to the level of the triangle, i.e. the triangle will not divide. Each tile must know the triangles that belong to it, or vice versa: each triangle needs to know its tile. Each tile is responsible for the area it has been assigned.

The goal here is to assign one tile to each triangle of the TIN. This means that any triangle is 'owned' by only one tile, however, each tile has more than one triangle. Each triangle is assigned to the tile with which its area of overlap is greatest. If two or more tiles have the same overlap area, assign a triangle to one of them in a systematic way. Figure 4.5. illustrate how a TIN overlapped with a tiling system.

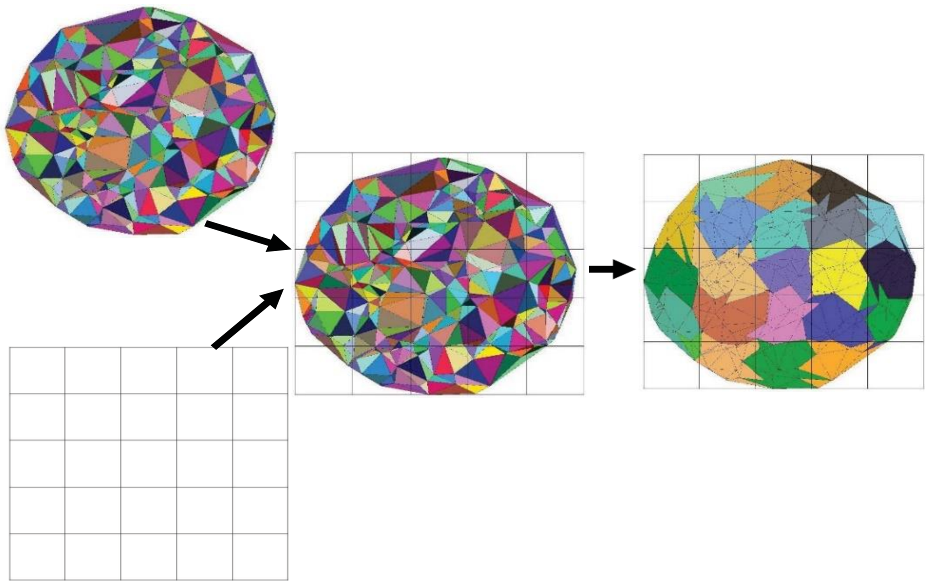


Figure 4.5. Tiling TIN structure (Al-Salami, 2009)

4.8 Attributive thematic data and metadata

Attributive thematic data can be any data associated with specific locations, places or objects.

Metadata is simply data about data. It is a description and context of the data. It helps to find, better understand and organize data. Typical metadata elements are title, description, author, date, who last modified it and when, etc. In the context of terrain modelling, metadata describes information connected with a particular terrain dataset or element of that dataset, e.g. name, location, resolution, size, attribution, data accuracy, format, copyright notices, etc (Reddy et al., 1999).

The international standard ISO 19115 is metadata standard in geographical domains. The problem is its use is mostly limited to 2D datasets and 3D datasets very rarely have stored metadata information. One of the reasons is that ISO 19115 specifications do not cover several aspects of 3D datasets.

The Federal Geographic Data Committee (FGDC) has developed a comprehensive standard for storing metadata related to digital geospatial data. This standard is called Content Standard for Digital Geospatial Metadata (CSDGM).

Figure 4.6. shows the division of metadata into mandatory, mandatory if applicable and optional.

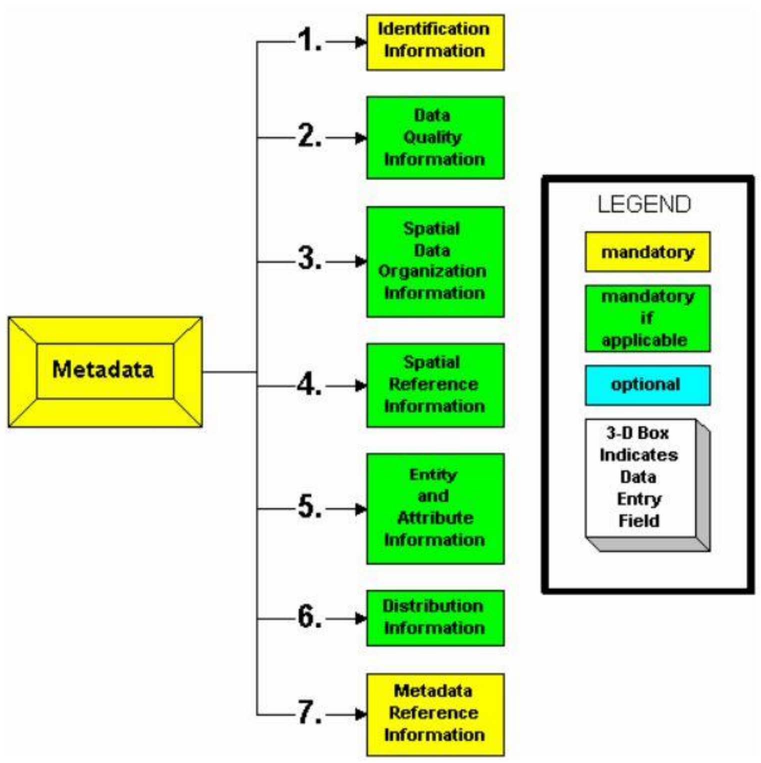

Metadata
1. Identification Information
2. Data Quality Information
3. Spatial Data Organization Information
4. Spatial Reference Information
5. Entity and Attribute Information
6. Distribution Information
7. Metadata Reference Information
LEGEND
mandatory
mandatory if applicable
optional
3-D Box Indicates Data Entry Field

Figure 4.6. Classification of metadata (URL 15.)

5. DESCRIPTION OF DATA ENCODINGS

In this chapter will be described in detail each of the file format that are used for delivering 3D Terrain Models as Triangulated Irregular Network. The formats to be described are VRML, X3D, CityGML, GML3, I3S, IFC, DXF and Wavefront OBJ.

5.1 The Virtual Reality Modeling Language (VRML)

The Virtual Reality Modeling Language (VRML) is a file format for presenting interactive 3D world objects over the Word Wide Web (WWW). It is used to create three-dimensional representations of complex scenes such as illustrations, definitions and virtual reality presentations. VRML can represent static and animated objects and it can have hyperlink to other media such as sound, movies and images. Many 3D modelling applications can save objects and scenes in VRML format (URL 1).

VRML 1.0 is the first release of VRML and it provides a means of creating and viewing static 3D worlds. VRML has been widely accepted as an international standard of interactive visualization since 1997, when ISO VRML97, or so-called VRML 2.0 is released. VRML 2.0 is a more advanced version and provides much more. The main goal of VRML 2.0 is to provide a more exciting and interactive user experience than is possible within the static boundaries of VRML 1.0. Its improvements are enhanced static worlds, interaction, animation, scripting and prototyping. VRML has been replaced with Extensible 3D (X3D) format. X3D is an XML based 3D file format. It supports all features of the VRML and also has some additions.

VRML code is written in plain text. Extenstion of VRML files is .wrl. These files could be viewed and adapted in a plain text editor, but the produced 3D renderings must be viewed in a VRML browser or other program, such as a web browser with a VRML plugin (URL 11).

This programming language can be used in software, such as CAD for importing elements and Cortona3D Viewer, which is an application that works as a VRML plugin for browsers and office applications. There is also an open source viewer called FreeWRL for Windows, Linux, OSX and Android and a free, open source software OpenVRML (URL 12).

The requirements of the VRML are as follows (Bell et al., 1996):

- authorability – allows the development of application editors and generators and the import of data from other industrial formats,
- completeness – provides all the information needed for implementation and address a complete set of features for wide acceptance in the industry,
- composability – the ability to use VRML elements in combination and thus enabling reuse,
- extensibility – the ability to add new elements,
- implementability – the ability to implement on a wide range of systems,
- multi-user potential – should not exclude the implementation of multi-user environments,
- orthogonality – should be independent of each other or if they are dependent, they should be structured and very good defined,
- performance – the elements should be created with the attention on interactive performance on different types of computing platforms,

- scalability – VRML elements should be created for unlimited large compositions,
- standard practice – standardized should be only elements that reflect existing practice, which is necessary for support in existing practice or that are needed to support suggested standards,
- well-structured – elements should have a well-defined interface and avoid elements with multi uses and side effects.

The scope and field of application include the following:

- a mechanism for saving and transporting two-dimensional and three-dimensional data elements,
- elements for representing two-dimensional and three-dimensional primitive information,
- elements for defining characteristics of that primitive information,
- elements for modelling and viewing 2D and 3D information,
- a container mechanism for including data from other metafile formats,
- mechanisms that define new elements which extend the capabilities of the metafile to support additional types and forms of information.

5.1.1 Nodes

The basic building block for VRML is a node. Nodes have fields. Fields serve as attributes and define the constant state of the node. There are 54 types of nodes, which can be divided into three groups: grouping nodes, children nodes and attribute nodes. Child nodes are in grouping nodes as an attribute. Grouping nodes sometimes contain other grouping nodes and also may contain other grouping nodes as children. A grouping node defines a coordinate system for its children's nodes relative to its parent coordinate system. The parent relationship supports a sequence of transformations that, when connected, position children nodes in the file's world coordinate space. Attribute nodes serve as attributes for other nodes (Taubin et al., 1998).

Nodes can receive and send messages to other nodes. These messages are called events. Events are commonly associated with the setting and changing of a node's fields. A route is a connection between a receiving node and a transmitting node. Routes are defined in the VRML file.

In VRML also exist one node that is called script node. The script node is special in that a user may augment it by defining additional events and fields. The uniform resource locator (URL) field of the script node contains program logic. The program logic defines the behaviour of the script node. This allows the script node to send and receive an event.

VRML supports the definition of new node types, called prototypes. Existing node types can be built-in or previously defined prototypes. The combination of prototypes and script nodes provides a powerful mechanism to encapsulate content and behaviour in a reusable entity.

One of the most important features of VRML is its support of the World Wide Web. VRML has several nodes that use URL's to connect all nodes to the network.

These nodes include:

- Inline node for additional VRML content,

- Hyperlinks to other URL's (Anchor node),
- AudioClip for audio content,
- JPEG or PNG files for images,
- VideoTexture node for audio and video files.

VRML supports different types of sensor nodes, such as environmental sensors, pointing-device select sensors and pointing-device drag sensors. Also, it supports a variety of interpolator nodes for use in creating linear animation supporting interpolation in colors, coordinates, normals and orientations.

This file format does not provide the capability to define units of measure. All linear distances are reputed to be in meters and all angles are in radians. Units of time are determined in seconds.

Color nodes define a set of RGB colors to be used in the fields of the other node. They are only used to specify multiple colors for one part of the geometry because it is a different color for each face or top of the IndexedFaceSet.

Material node specifies surface material properties for jointed geometry nodes. Also, it is used by the VRML lighting equations during rendering. Material node is used for determining the total number of material parameters of lighted geometry. All fields in the material node are in the range of 0.0 to 1.0. If the material node and the color node are listed for geometry, then color should ideally replace the component of the material.

Textures have the advantage over colors, specifying both texture and color node for geometry results in neglecting the color node.

Shape nodes define the geometry in the world. The shape node contains exactly one geometric node in its geometric field. This node must be one of the these node types: Box, Cone, Cylinder, ElevationGrid, Extrusion, IndexedFaceSet, PointSet, Sphere, and Text.

Some of the geometry nodes also contain Coordinate, Color, Normal and TextureCoordinate as geometry property nodes. All geometry nodes are specified in a local coordinate system and are affected by parent transformations.

5.1.2 Coordinate Systems and Transformations

VRML uses a Cartesian, right-handed, three-dimensional coordinate system. By default, objects are projected onto a two-dimensional display device by projecting them in the direction of the positive Z-axis, with the positive X-axis to the right and the positive Y-axis up. Transformation of a camera or modelling may be used to change this default projection (URL 10).

VRML worlds may include an arbitrary number of local (or „object-space“) coordinate systems, defined by modelling transformations using nodes such as Translate, Rotate, Scale, Transform, and MatrixTransform. Given a vertex V and a series of transformations such as:

- Translation T,
- Rotation R,
- scaleFactor S,

the vertex is transformed into world-space to get V' by applying the transformations in the following order:

$$V' = T \cdot R \cdot S \cdot V \tag{1}$$

$$V' = V \cdot S \cdot R \cdot T \tag{2}$$

The first formula (1) is used if the vertices are considered as column vectors and the second (2) if the vertices are considered as row vectors.

VRML also has a „world“ coordinate system and viewing or „Camera“ coordinate system. The different local coordinate transformations map objects into the world coordinate system. This is where the scene is assembled. Then is the scene viewed through a camera and is introduced another conceptual coordinate system. In VRML nothing is specified using these coordinates. They can rarely be found in optimized implementations where all the steps are connected. However, having a clear model of the objects, the world and the camera spaces will help authors.

Since VRML has a Cartesian local coordinate system, there is a need to connect that coordinate system to any one of the standards, global coordinate systems. This problem is solved by GeoVRML. It is a file format, an open extension to VRML that supports geographic coordinate systems and the fusion of different data sources (Goodchild and Kimerling, 2002).

Node coordinate defines a set of 3D coordinates, which will be used in the *coord* field of vertex-based geometry nodes (IndexedFaceSet, IndexedLineSet).

5.1.3 TIN representation in VRML

The IndexedFaceSet node is equivalent to cartographic 3D Triangular Irregular Networks (TINs). It is a 3D shape created by constructing faces (polygons) from vertices specified in the *coord* field. The coordinate field must include a Coordinate node. IndexedFaceSet is using indices in its *coordIndex* for determining polygonal faces. Index -1 means that the current face is finished and the next one begins. If the largest index in the *coordIndex* field is N, then the Coordinate node has to contain N + 1 coordinates. IndexedFaceSet is described in the local coordinate system and is affected by parent transformations.

The IndexedFaceSet node has three SFBool fields that provide hints about shape. These fields are ccw, solid and convex. The ccw field indicates that the vertices are ordered in a counter-clockwise direction when the shape is viewed from the outside. In this case, field value is TRUE. Otherwise, field value is FALSE and the vertices are ordered in a clockwise direction. The solid field indicates if the shape encloses a volume (TRUE). The convex field indicates whether all faces in the shape are convex (TRUE).

IndexedFaceSet uses all four of the geometric property nodes for specifying vertex coordinates, colors per vertex, normals per vertex and texture coordinates per vertex.

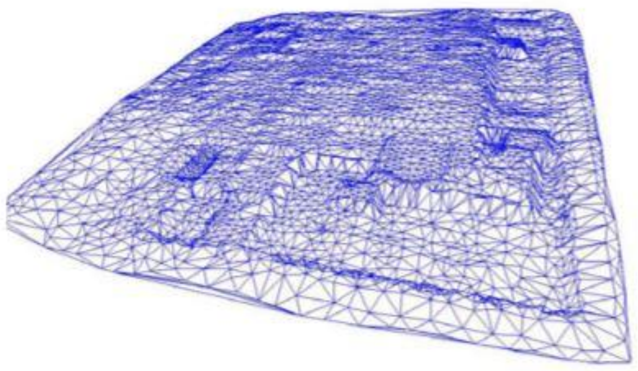


Figure 5.1. The terrain represented as TIN (Gomez and Gonzalez, 2006)

Figure 5.1. shows a Triangular Irregular Network generated from a point cloud input.

5.1.4 LOD and support of tiling concepts

The LOD node determines various levels of detail or complexity for a given object and provides suggestions for browsers to automatically choose the appropriate version of the object based on the distance from the user.

This node contains fields level, range and center. The level field has a list of nodes that represent the same object or objects at varying levels of detail, from highest detail to lowest. The range field determines the ideal distances to switch to between the levels.

The center field is a translation offset in the local coordinate system that specifies the center of the LOD objects for calculating the distance. In order to calculate which level to display, it is first necessary to calculate the distance from the point of view, then transform it into the local coordinate space of the LOD node, to the center point of the LOD. The first level of the LOD is drawn when the distance is less than the first value in the range field. If the value is between the first and second value of the range field, then the second level is drawn, and so on. If in the range field is N values, the LOD should have N+1 nodes in its level field.

Transitions from one level of detail to another have to be smooth. Browsers may adapt which level of detail is displayed to keep interactive frame rates, to display an already-fetched level of detail while a higher level of detail is fetched. LOD nodes continue to receive and send events (routes), no matter which LOD level is active.

The LOD nodes are also used for division the terrain model by providing an additional internal tiling within each VRML world. This process of the tiling is used to allow multiple data sets of varying resolution to be provided for each LOD node and it allows that VRML browser can render each LOD node at a suitable resolution. The nodes that are further from the user are rendered at a lower resolution which reduces the rendering effort (Leung and Coddington, 1998).

Figure 5.2. shows four different resolutions of a digital map where each level has been segmented into a regular grid of tile and then the TIN is created independently for each tile.

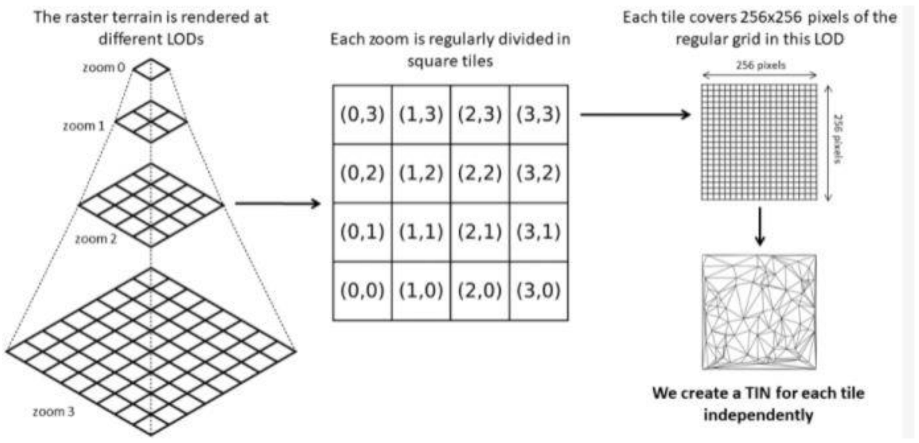

The raster terrain is rendered at different LODs
zoom 0
zoom 1
zoom 2
zoom 3
Each zoom is regularly divided in square tiles
(0,3) (1,3) (2,3) (3,3)
(0,2) (1,2) (2,2) (3,2)
(0,1) (1,1) (2,1) (3,1)
(0,0) (1,0) (2,0) (3,0)
Each tile covers 256x256 pixels of the regular grid in this LOD
256 pixels
256 pixels
We create a TIN for each tile independently

Figure 5.2. Four different resolutions and TIN (Campos et al., 2020)

The problem occurs when there are a large number of levels of detail, because VRML browsers load all scenes at once. To solve this problem, two approaches have been developed for movement through the LOD hierarchy.

First approach is Anchor/LOD tree files. Here three or four levels are loaded at the same time which gives an acceptable compromise between download time and interactivity. Therefore, tree files are produced that each contains a small LOD hierarchy where each tile is an anchor node in a higher resolution tree file. This approach allows navigating deep LOD hierarchies, but has the disadvantage of requiring the user to click on areas to receive larger resolution, but then only a small area of interest is visible.

Second approach is called QuadLOD tree files (Figure 5.3.). This is a more practical approach and it used Java script to manage the loading and unloading of data. This node implements a specialized quad-tree LOD facility where the four higher-resolution children of a tile are loaded only when the user enters a specific proximity volume around the tile. These children are also unloaded when the user leaves this volume.

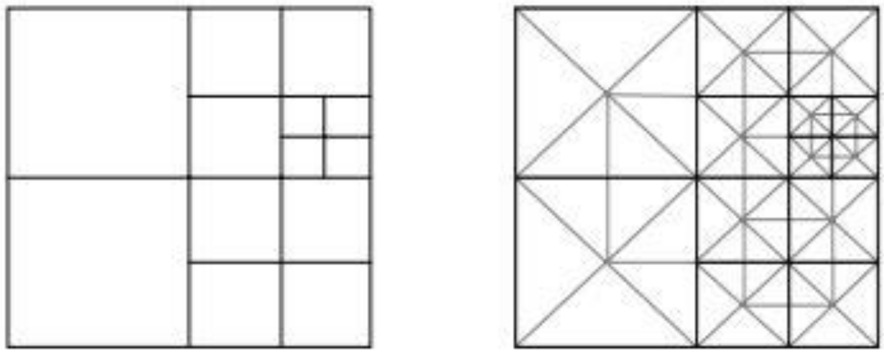


Figure 5.3. Matching triangulation of a restricted quadtree subdivision (Pajarola, 2002)

5.1.5 Attributive thematic data and metadata

VRML has basic support for attributes.

In VRML 2.0 Info node is replaced with the WorldInfo node. It provides the scene's title and other information about the scene, such as information about the author and copyright information. This node was for many years the only way to persistently save metadata in scenes. Since each WorldInfo node could include a set of arbitrary string values, any type of metadata data could be included. That is true. However, without an internal structure and without the ability to change such strings during execution, WorldInfo has limited practical use from a metadata point of view and is rarely used. This led to the definition of typed Metadata nodes which are used in X3D file format.

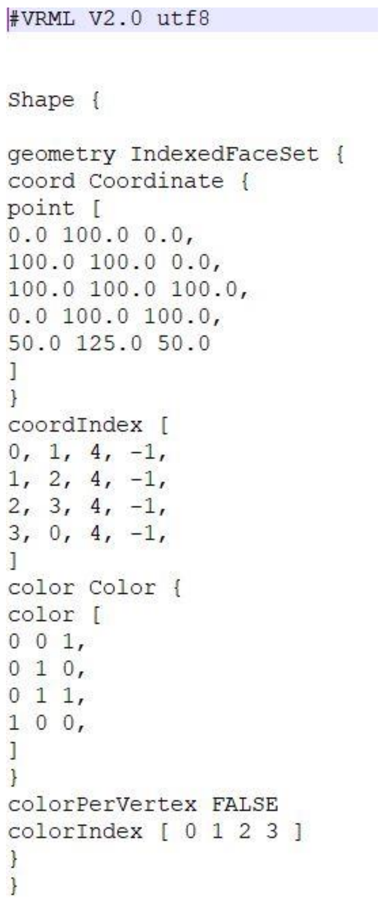
5.1.6 Sample file

Figure 5.6. shows sample file in which there are five vertices with X, Y, Z coordinates and four faces. Most VRML Viewers cannot handle large X, Y, Z coordinates (Table 2). Thus, for this example, an offset of '-200000.0 0.0 -5500000.0' has been added. To create a TIN IndexedFaceSet is used.

Sample file is written in Notepad++ (Figure 5.4.) and visualized in the FreeWRL programme (Figure 5.5.).

Table 1. Coordinates

X	Y	Z
200000.0	5500000.0	100.0
200100.0	5500000.0	100.0
200100.0	5500100.0	100.0
200000.0	5500100.0	100.0
200050.0	5500050.0	125.0



```
#VRML V2.0 utf8


Shape {

geometry IndexedFaceSet {
coord Coordinate {
point [
0.0 100.0 0.0,
100.0 100.0 0.0,
100.0 100.0 100.0,
0.0 100.0 100.0,
50.0 125.0 50.0
]
}
coordIndex [
0, 1, 4, -1,
1, 2, 4, -1,
2, 3, 4, -1,
3, 0, 4, -1,
]
color Color {
color [
0 0 1,
0 1 0,
0 1 1,
1 0 0,
]
}
colorPerVertex FALSE
colorIndex [ 0 1 2 3 ]
}
}
```

Figure 5.4. Sample file

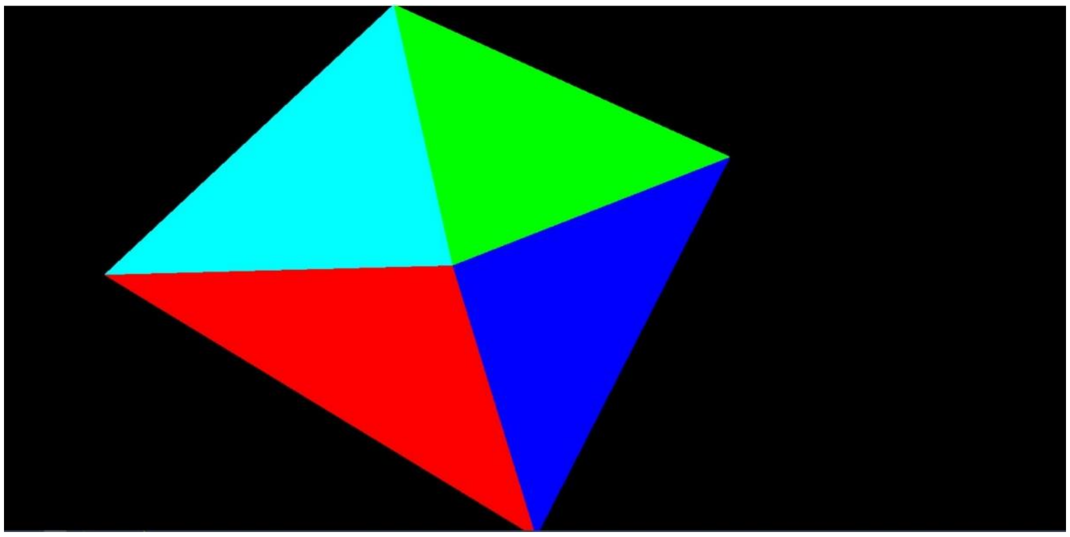


Figure 5.5. File visualization

5.2 Extensible 3D (X3D)

Extensible 3D Graphic is an improved version of the VRML format and they have many similarities. For this reason, only X3D enhancements that VRML format does not have will be described in this subchapter.

X3D is a free open standard for viewing, printing, publishing and archiving 3D models on the Web. X3D is an ISO standard and it fully represents three-dimensional data. This standard can run on many platforms and render 3D models in most web browsers without requiring additional applications (URL 31).

It provides classes and attributes for supporting a wide array of application domains such as scientific visualization, CAD, GIS, animation, 3D printing, Augmented and Virtual Reality.

This format may describe how to connect real-world locations to elements in the X3D world as specifying nodes particularly set for geospatial applications. It supports a limited set of coordinate reference systems.

X3D file format supports geometry definitions such as sphere, cylinder, cone, polygonal meshes and non-uniform rational basis splines (NURBS). X3D does not support constructive solid geometry directly.

This file format has a lot of nodes and fields for controlling the appearance of objects, including material attributes, texture mapping and fill properties.

5.2.1 Attributive thematic data and metadata

In an X3D file a metadata node can be used to provide information about any node in an X3D graph and it is placed as a child of the node that describes. Metadata nodes are persistent, meaning that their values remain available and accessible after loading. Metadata nodes do not affect the visual representation of the scene. The X3D metadata nodes include MetadataBoolean, MetadataDouble, MetadataFloat, MetadataInteger, MetadataString and MetadataSet (URL 16).

Other mechanisms for placing information on the scene include <meta> tags located in the <head> section of the document. Meta tags provide attribute-value pairs of information about the overall scene.

Another way to store metadata are comments. Comments are unstructured text that can provide useful information to authors. They are not retained when the X3D scene is analysed and are not presentable to users when the scene is loaded.

X3D has no native capability for custom attributes, but allows the linking of attributes in other XML files for that capability.

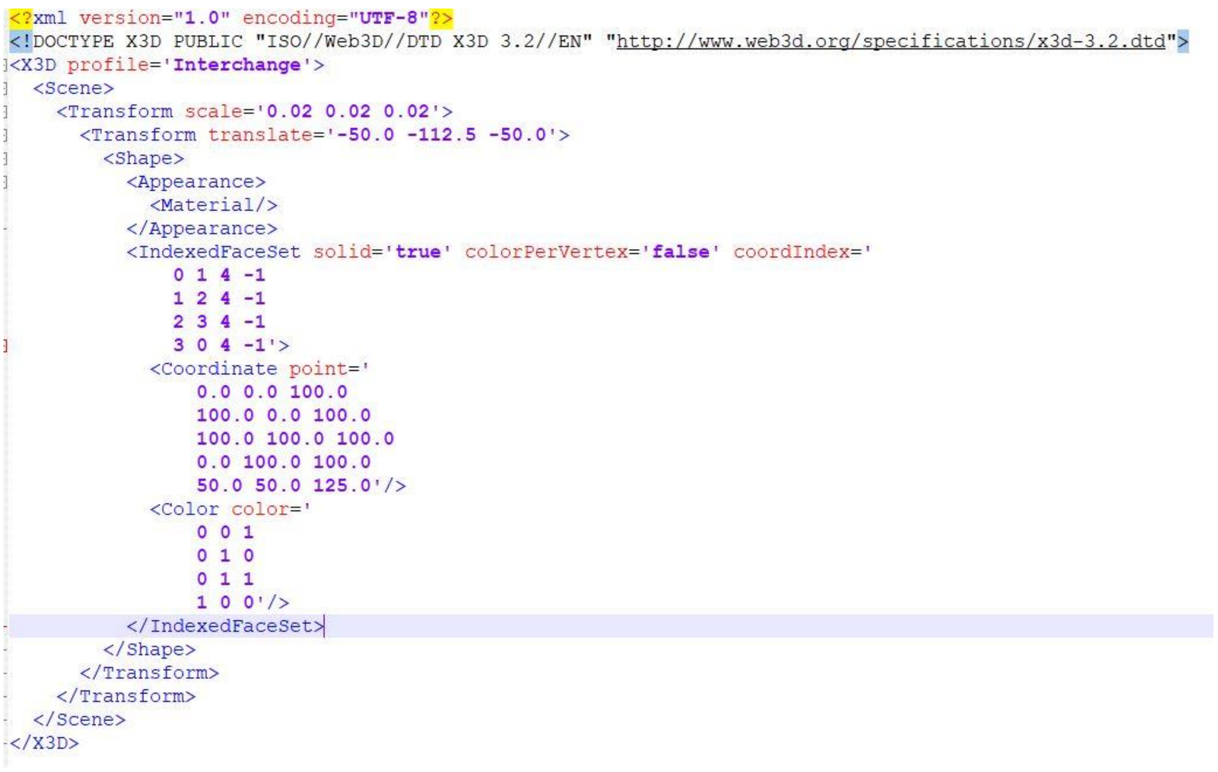
5.2.2 Sample file

Figure 5.6. shows a sample file in which there are five vertices with X, Y, Z coordinates and four faces. Most X3D Viewers are no able to handle large X, Y, Z coordinates (Table 2). Thus, for this example, an offset of '-200000.0 0.0 -5500000.0' has been added. To create a TIN IndexedFaceSet is used.

Sample file is written in Notepad++ (Figure 5.6.) and visualized in the FreeWRL programme (Figure 5.7.).

Table 2. Coordinates

X	Y	Z
200000.0	5500000.0	100.0
200100.0	5500000.0	100.0
200100.0	5500100.0	100.0
200000.0	5500100.0	100.0
200050.0	5500050.0	125.0



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN" "http://www.web3d.org/specifications/x3d-3.2.dtd">
<X3D profile='Interchange'>
  <Scene>
    <Transform scale='0.02 0.02 0.02'>
      <Transform translate='-50.0 -112.5 -50.0'>
        <Shape>
          <Appearance>
            <Material/>
          </Appearance>
          <IndexedFaceSet solid='true' colorPerVertex='false' coordIndex='
              0 1 4 -1
              1 2 4 -1
              2 3 4 -1
              3 0 4 -1'>
            <Coordinate point='
                0.0 0.0 100.0
                100.0 0.0 100.0
                100.0 100.0 100.0
                0.0 100.0 100.0
                50.0 50.0 125.0'/>
            <Color color='
                0 0 1
                0 1 0
                0 1 1
                1 0 0'/>
          </IndexedFaceSet>
        </Shape>
      </Transform>
    </Transform>
  </Scene>
</X3D>
```

Figure 5.6. Sample file of X3D format

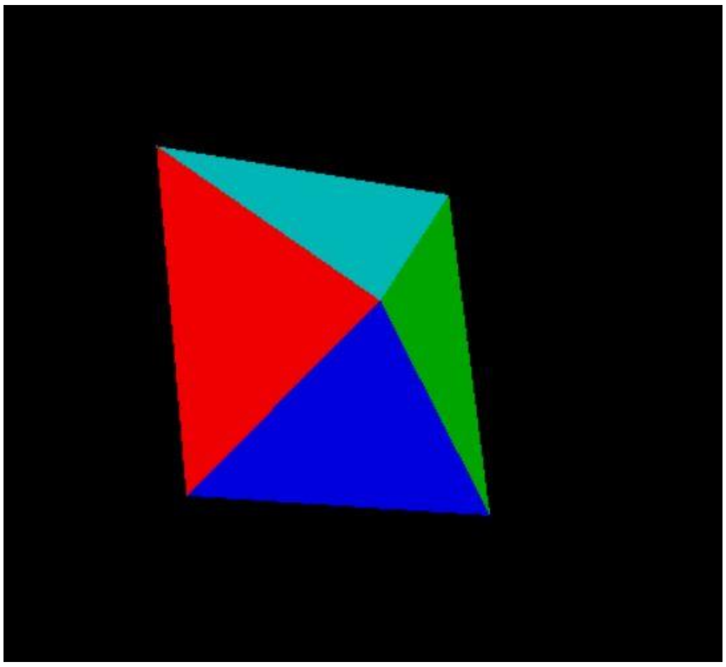


Figure 5.7. File representation

Figure 5.7. shows four triangles forming a TIN with each other. Each of the triangles has its color.

5.3 City Geography Markup Language (CityGML)

The City Geography Markup Language (CityGML) is a concept for the modelling and exchange of 3D city and landscape models and it is quickly being adopted on an international level. CityGML is a common informative model for presenting 3D urban objects. This model defines classes and relationships for the most relevant topographic objects in cities and regional models concerning their geometric, topological, semantic, and appearance properties. Hierarchies of generalization between thematic classes, aggregations, relationships between objects and spatial properties are included. Unlike other 3D vector formats, CityGML is based on a rich, general-purpose information model, with geometry and graphical content that allows the use of virtual 3D city models for sophisticated analysis tasks in various application domains such as simulation, urban data mining, facility management and thematic inquiries. Areas of application explicitly contain urban and landscape planning, architectural design, tourist activities, 3D cadastres, mobile telecommunications, disaster management, vehicle and pedestrian navigation and mobile robotics (URL 17).

CityGML is an open data model and it is an XML-based format for the storage and exchange of virtual 3D city models. The goal of CityGML development is to achieve a common definition of basic entities, attributes and relations of the 3D model of the city. This is especially important for the cost-effective sustainable maintenance of 3D city models because it allows the reuse of the same data in different application fields (Gröger et al., 2012).

CityGML is an official OGC and ISO TC211 standard and can be used free of charge.

Software systems that provide CityGML support are CityEditor, Bentley Map, BS Contact Geo, CityGRID and CityServer3D (URL 19).

CityGML can be applied to large areas and small regions and can represent terrain and 3D objects simultaneously with different levels of detail. Models can be represented as simple models without topology and with few semantics or can be very complex multi-scale models with full topology and fine-grained semantical differentiations. Because of it, CityGML allows lossless information exchange between different GI systems and users.

Features of CityGML are:

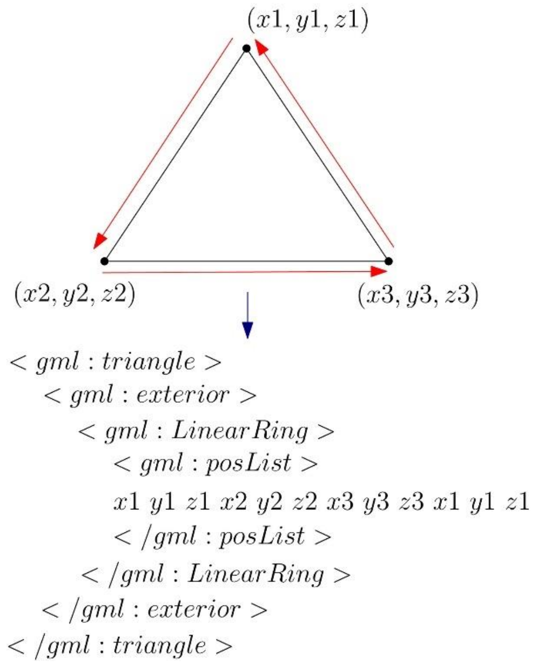
- Geospatial information model for urban landscape,
- GML3 representation of 3D geometries, based on the ISO 19107 model,
- Representation of characteristics of object surface such as textures, materials,
- Taxonomies and aggregations:
    1. Digital Terrain Models as a combination of TINs, regular raster, break and skeleton lines, mass points,
    2. Sites (buildings, bridges, tunnels),
    3. Vegetation,
    4. Water bodies,
    5. Transportation facilities,
    6. Land use,
    7. City furniture,
    8. Generic city objects and attributes,
    9. User-defined (recursive) grouping,
- Five well-defined consecutive levels of detail (LOD): LOD0 (regional, landscape), LOD1 (city, region), LOD2 (city districts, projects), LOD3 (external architectural models), LOD4 (interior architectural models),
- Multiple displays in different LODs simultaneously, generalization of relations between objects in different LODs,
- Optional topological relationships between (sub) geometry features.

CityGML represents the graphical appearance of the city model and also semantic and thematic properties, taxonomies and aggregations. It includes geometry and a thematic model. The model of geometry admits consistent and homogeneous definitions of geometry and topological properties of spatial objects within 3D city models. CityObject is the basic class of all objects and it is a subclass of a GML class Feature. All objects inherit the properties from CityObject.

Although surfaces in CityGML have to be planar and 2-manifold, this format offers some support for non-manifold topology in the form of Cell Complex.

5.3.1 TIN representations in CityGML

The simplest way to represent a TIN is to store each of its triangles as a list of vertex coordinates. *Simple Feature* (Figure 5.8.) is one example of such a data structure. Each triangle is stored as a closed linear ring with its vertex coordinates. The disadvantage of this data structure is the redundancy of the data, ie the first vertex of each ring is repeated as the last vertex of the linear ring. Second, it has very limited topology and does not explicitly store the adjacency relationships between the triangles which are necessary for spatial analysis. Another problem is vertical triangles representation in TIN. CityGML is a 2.75D model, but there is no procedure to explicitly handle these vertical triangles (Kumar et al., 2016).



```
< gml : triangle >
   < gml : exterior >
      < gml : LinearRing >
         < gml : posList >
         x1 y1 z1 x2 y2 z2 x3 y3 z3 x1 y1 z1
         < /gml : posList >
      < /gml : LinearRing >
   < /gml : exterior >
< /gml : triangle >
```

Figure 5.8. Simple Feature representation of a triangle (URL 18)

5.3.2 Storing terrains in CityGML

The CityGML data model consists of a core model and several thematic models for describing urban features such as Building, Relief, LandUse, Transportation and WaterBody. In this data model terrains are defined within the thematic module Relief and represented by the class ReliefFeature, which have 5 different levels of detail (LOD 0-4). With ReliefFeature, the terrain can be displayed as a TIN (TINRelief), a grid (RasterRelief), mass points (MasspointRelief) and as break lines (BreaklineRelief). Terrain can also be represented as a combination of different terrain types in one CityGML data set.

TINReflief (Figure 5.9) represents terrains as TINs using either GML geometry class gml:TriangulatedSurface or gml:Tin. The triangles of a TIN are in gml:TriangulatedSurface explicitly specified with Simple Feature geometry. In gml:TIN, only 3D points are represented, where the triangulation can be reconstructed by standard methods (Delaunay triangulation). However, the disadvantages of Simple Features are visible in the implementation of CityGML when working with massive field datasets.

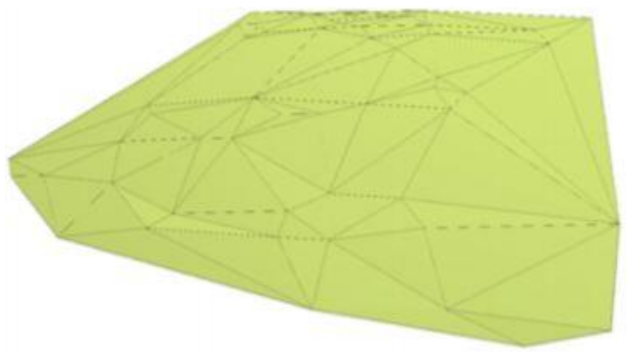


Figure 5.9. CityGML terrain as TINRelief (Kumar et al., 2019)

5.3.3 LOD and support of tiling concept

CityGML specifies five levels of detail (LOD), reflecting different accuracies or resolutions. But in practice, the problem is that it does not exist exact specifications for terrain LODs. There is no difference between different LODs of terrain at geometrical and semantic levels.

Also, there is no support for tiling. The main role in deciding the maximum size of the dataset that can be processed is played by the main memory. If the size of the datasets exceeds the available memory limit, then it is split into small parts called tiles. The concept of TIN tiling cannot be extended to CityGML because there may be triangles that include several tiles. Such triangles are repeated in the spanning tiles to complete the closed linear ring structure, which causes redundancy of information in the CityGML datasets (Kumar et al, 2016).

5.3.4 Attributive thematic data and metadata

Many attributes which are covered in the data model of CityGML. However, if someone wants to model attributes that are not covered in CityGML data model, then there are two possible solutions. First option is to use Generic city attributes and the second is to use Application Domain Extensions (ADE).

Generic CityGML module is a semi-structured extension mechanism in which the city objects are extended with additional objects and attributes, but without making any changes in the CityGML schema. The problem with using generic attributes is that it is not possible to verify them against the schema because their names and data types are not formally defined in the schema. Consequently, the use of Generics has very limited semantic and syntactic interoperability.

CityGML has a mechanism called Application Domain Extensions (ADE). This mechanism specifies additions to the CityGML data model. Such additions include introducing new properties into existing CityGML classes. The ADE has to be defined in an additional XML schema definitional file with its namespace. This file must explicitly import the XML schema definition of extended CityGML modules. Existing classes and objects in CityGML can be supplemented with thematic attributes using ADE. The type and quantity of these attributes are selectable (Groger et al., 2012).

The Figure 5.10. shows a flowchart from which it can be read that the attributes can be assigned to both the triangles and the TIN itself.

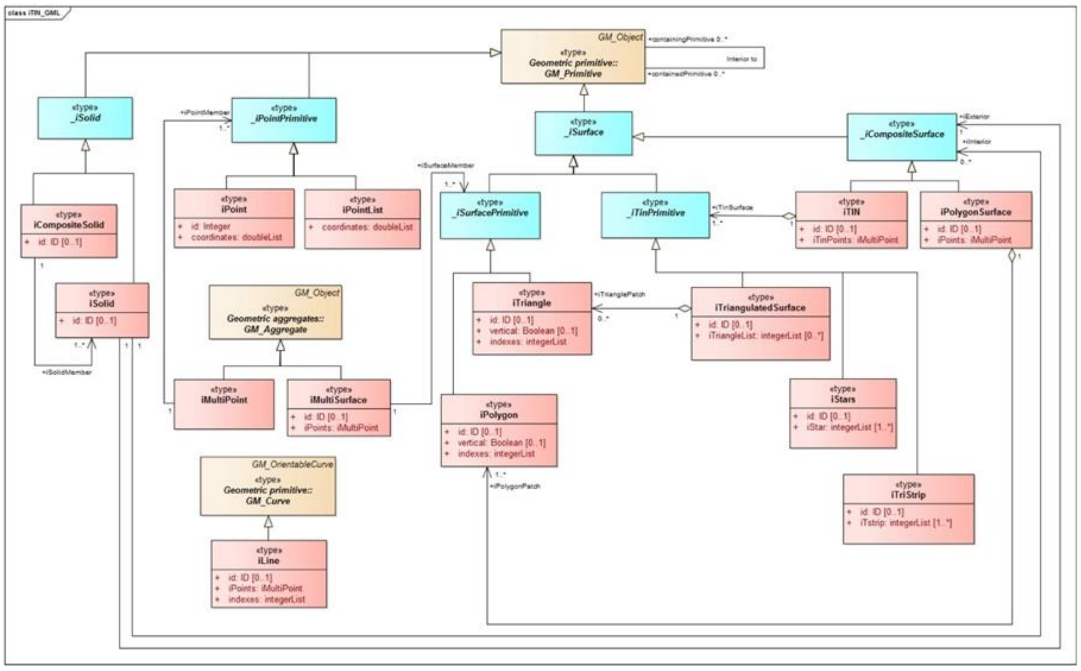
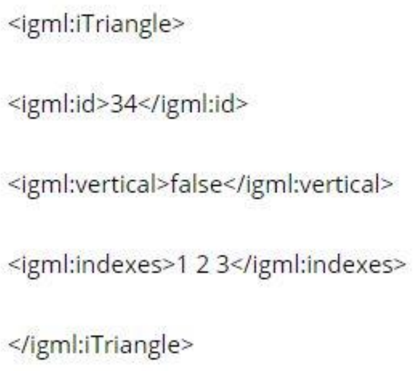


Figure 5.10. Flowchart (URL 18)

iTriangle (Figure 5.11.) represents the geometry of an individual triangle. It has an optional element igml:vertical to specify if the triangle is a vertical because sometimes it is easier to use a single triangulated surface containing vertical triangles instead of using a volumetric model. This means that the model is more than 2.5D, but less than 3D. The geometry is 3D, but the underlying topology in 2D.



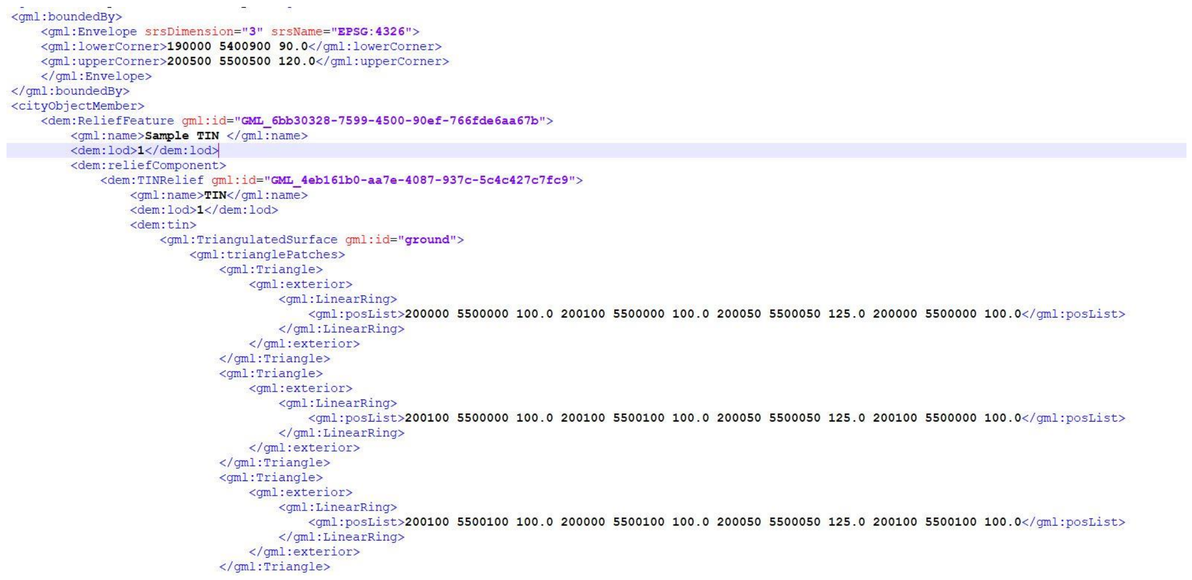
```
<igml:iTriangle>

<igml:id>34</igml:id>

<igml:vertical>false</igml:vertical>

<igml:indexes>1 2 3</igml:indexes>

</igml:iTriangle>
```

Figure 5.11. iTriangle

In CityGML, metadata are rarely stored. The reason is that CityGML does not offer mechanisms to store metadata in a structured way. Practitioners often need to define their own methodology for storing them (URL 20).

5.3.5 Sample file

In CityGML file format TINRelief represents terrain as TIN using gml:TriangulatedSurface. Gml:TrianglePatches close triangles with a triangular surface. Gml:LinearRing creates a triangle of 3 points with coordinates of which the first point must be also the last. CityGML supports geodetic coordinate reference systems. Figure 5.12 shows part of the CityGML file.



```xml
<gml:boundedBy>
    <gml:Envelope srsDimension="3" srsName="EPSG:4326">
    <gml:lowerCorner>190000 5400900 90.0</gml:lowerCorner>
    <gml:upperCorner>200500 5500500 120.0</gml:upperCorner>
    </gml:Envelope>
</gml:boundedBy>
<cityObjectMember>
    <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b">
        <gml:name>Sample TIN </gml:name>
        <dem:lod>1</dem:lod>
        <dem:reliefComponent>
            <dem:TINRelief gml:id="GML_4eb161b0-aa7e-4087-937c-5c4c427c7fc9">
                <gml:name>TIN</gml:name>
                <dem:lod>1</dem:lod>
                <dem:tin>
                    <gml:TriangulatedSurface gml:id="ground">
                        <gml:trianglePatches>
                            <gml:Triangle>
                                <gml:exterior>
                                    <gml:LinearRing>
                                        <gml:posList>200000 5500000 100.0 200100 5500000 100.0 200050 5500050 125.0 200000 5500000 100.0</gml:posList>
                                    </gml:LinearRing>
                                </gml:exterior>
                            </gml:Triangle>
                            <gml:Triangle>
                                <gml:exterior>
                                    <gml:LinearRing>
                                        <gml:posList>200100 5500000 100.0 200100 5500100 100.0 200050 5500050 125.0 200100 5500000 100.0</gml:posList>
                                    </gml:LinearRing>
                                </gml:exterior>
                            </gml:Triangle>
                            <gml:Triangle>
                                <gml:exterior>
                                    <gml:LinearRing>
                                        <gml:posList>200100 5500100 100.0 200000 5500100 100.0 200050 5500050 125.0 200100 5500100 100.0</gml:posList>
                                    </gml:LinearRing>
                                </gml:exterior>
                            </gml:Triangle>
```

Figure 5.12. Part of the xml-based CityGML file

Color is added in the program called eveBIM. It is possible to add it to Properties. One color is added to the whole TIN, not to each triangle separately as seen in Figure 5.13.

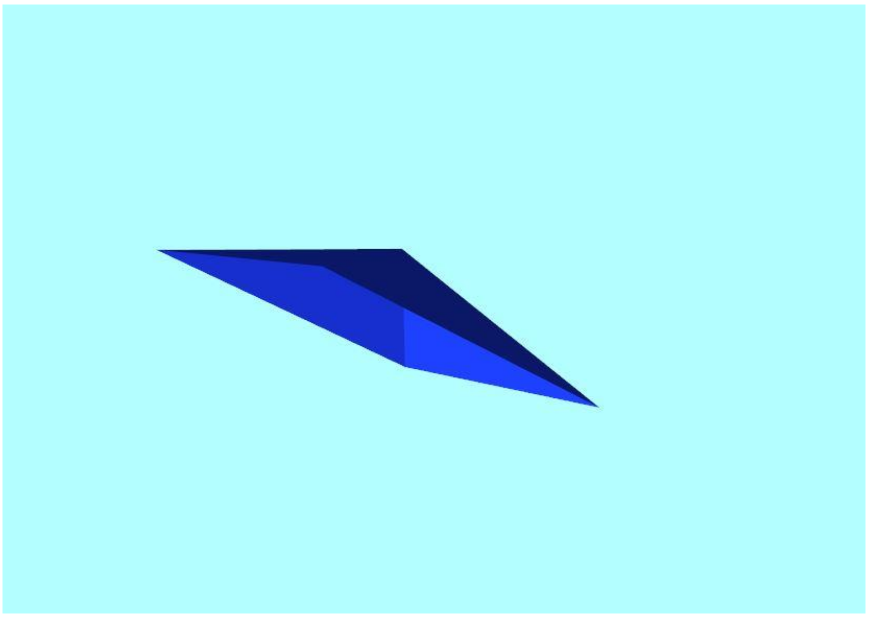


Figure 5.13. TIN visualization in eveBIM

5.4 Geography Markup Language 3 (GML3)

Geography Markup Language is based on XML and it stores, transports and represents geographic information such as attributes, geometries and relationships. GML was developed by the Open Geospatial Consortium (OGC).

GML 3 defines some complex entities such as topologies, dynamic and nested features. These entities have many optional aspects. It includes support for complex geometries, spatial and temporal reference systems, topology, units of measure, metadata, gridded data, and default styles for feature and coverage visualization (Portele, 2012).

GML 3 allows encoding several types of geographic features, including (Portele, 2012):

- Geographic features with their geometry, topology and temporal evolution,
- Geographic coverage, including geometry and attribute values,
- Geographical observations,
- Coordinate reference system (CRS),
- Abstract values (numerical quantities with units of measurement, categorization).

GML specifies XML encodings, accordingly to ISO 19118 standard, of several of the conceptual classes defined in the ISO 19100 series of International Standards and the OpenGIS Abstract Specification. These conceptual models include those defined in (Portele, 2012):

- ISO/TS 19103 — Conceptual schema language (units of measure, basic types),
- ISO 19107 — Spatial schema (geometry and topology objects),
- ISO 19108 — Temporal schema (temporal geometry and topology objects, temporal reference systems),
- ISO 19109 — Rules for application schemas (features),
- ISO 19111 — Spatial referencing by coordinates (coordinate reference systems),
- ISO 19123 — Schema for coverage geometry and functions,
- ISO 19148 — Linear Referencing.

GML provides XML Schema syntax, mechanisms and conventions which:

- have an open framework for description of a geospatial application for storage and transport geographic information in XML,
- allow profiles that support appropriate subsets of the descriptive capabilities of the GML framework,
- support geospatial application schemas description for specialized domains and information communities,
- enable the creation and maintenance of related geographic schemes of applications and datasets,
- support of application schemas storage and transport,
- increase the ability of organizations for sharing geographic application schemas and the information they describe.

GML3 has a temporal reference system for mesuring time.

The Web Feature Service (WFS) is a change in the way geographic data is created, modified and shared on the Internet. WFS 2.0.0 request has to return GML 3.2 as the default format. WFS 1.1.0 requests return GML 3 as the default format (URL 25).

5.4.1 TIN representation in GML3

SimpleTriangle is a spacial type of simple polygon and has 3 control points. Unlike the CityGML format, in this format is not necessary to repeat the first point as the last point.

Triangulated Irregular Network (gmltin:TIN) uses Delaunay or a similar algorithm supplemented with limitations defined by TIN elements (gmltin:tinElements). TIN elements specify elements connected with TIN such as vertices (points), limitations (boundary, break line, hole, stop line, etc.) and user-defined elements. The gmltin:elementType defines the type of TIN element.

Gmltin:elementGeometry specifies the geometry of the TIN element. Gmltin:elementType defines the geometry type and it could be:

- Gml:MultiPoint – random points TIN elements with 3D geometry,
- Gml:Polygon – boundary, break void and void TIN elements with 3D geometry,
- Gml:Polygon – drape void and hole TIN elements with 2D or 3D geometry,
- Gml:LineString – break line, soft break and control contour TIN elements with 3D geometry,
- Gml:LineString – stop line TIN elements with 2D or 3D geometry,
- Gml:GemetryPropertyType – user-defined TIN element.

Gmltin: TINElementTypeType is a code list that is an aggregation of enumerations of predefined TIN element types and a form for specifying a user-defined TIN type.

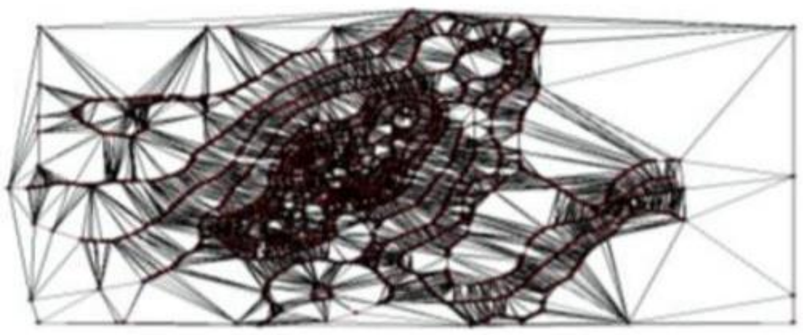


Figure 5.14. TIN created with mass points (Bhargava et al, 2013)

Figure 5.14. shows creation of TIN from the mass points using Delauney's Triangulation method.

5.4.2 Attributive thematic data and metadata

An attribute is a named property of a class that describes a range of values that property cases can hold. Thematic properties of objects are modeled as attributes of the corresponding class.

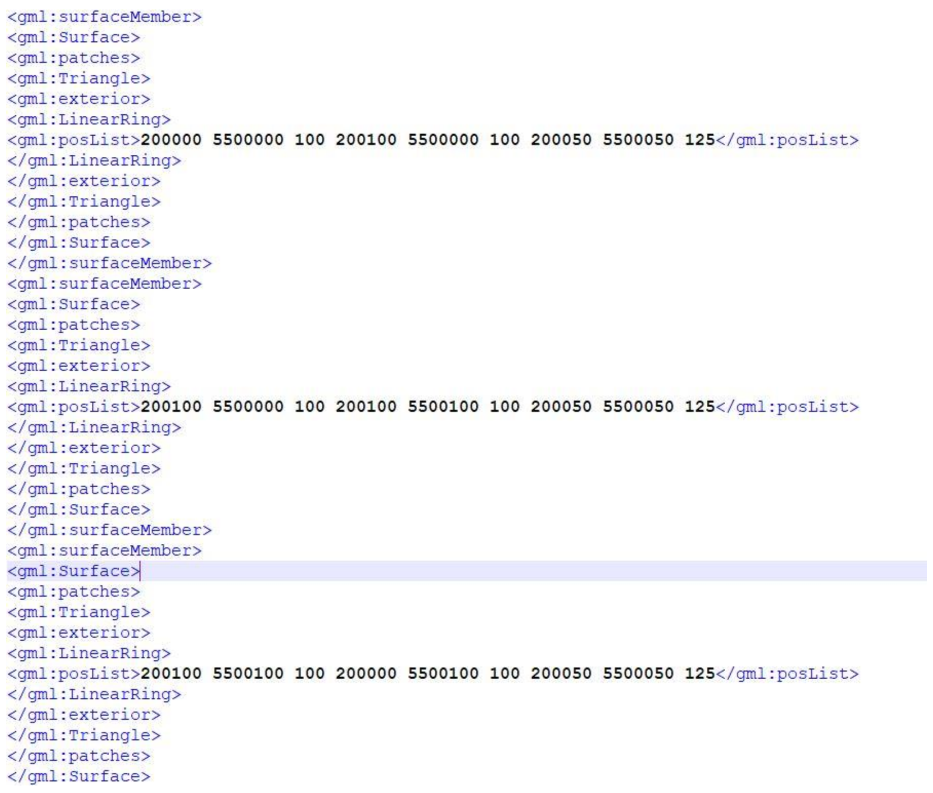
Each element has a type, identified by name (generic identifier) and can have a set of attribute specifications. Each attribute specification has a name and a value. An XML schema connects attributes and elements of an XML object.

The metaDataProperty follows a property form and is used to contain or refer to metadata for GML objects. It is used in one of the following ways:

- A single root element must be declared in the metadata schema. This element must be replaceable for gml_MetaData.
- Attached to a GML object and without the „gml:about“ attribute it provides metadata for the GML object to which it is linked.
- Attached to a collection of GML objects and with the gml:about attribute it provides metadata for GML objects within the collection of objects referenced by the gml:about attribute, the value of which is an XPointer expression.

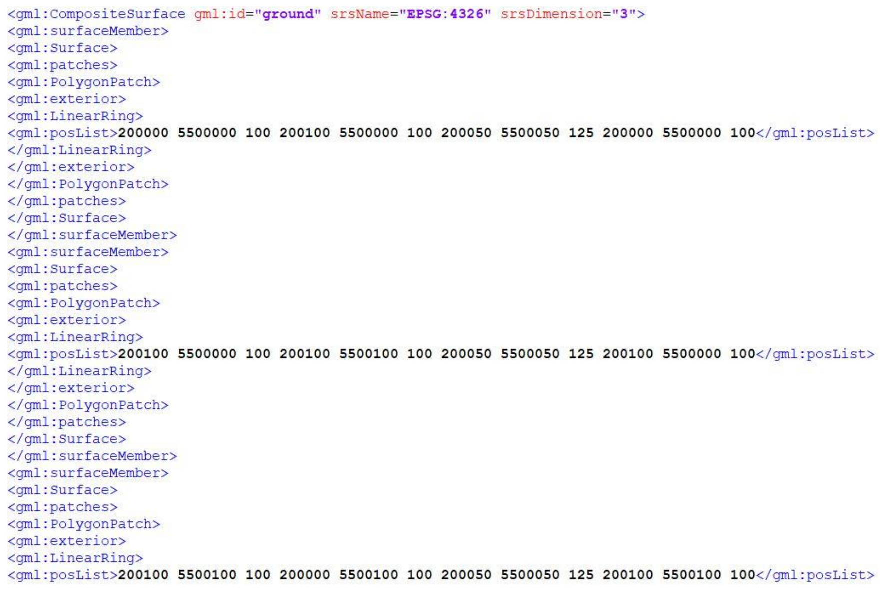
5.4.3 Sample file

In GML file format triangles can be represented using gml:Triangle (Figure 5.15) or gml:PolygonPatch (Figure 5.16). In gml:PolygonPatch the last coordinate has to be the same as the first, while in gml:Triangle the last coordinate does not repeat the first. GML has support for a geodetic coordinate reference system.



```
<gml:surfaceMember>
<gml:Surface>
<gml:patches>
<gml:Triangle>
<gml:exterior>
<gml:LinearRing>
<gml:posList>200000 5500000 100 200100 5500000 100 200050 5500050 125</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:Triangle>
</gml:patches>
</gml:Surface>
</gml:surfaceMember>
<gml:surfaceMember>
<gml:Surface>
<gml:patches>
<gml:Triangle>
<gml:exterior>
<gml:LinearRing>
<gml:posList>200100 5500000 100 200100 5500100 100 200050 5500050 125</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:Triangle>
</gml:patches>
</gml:Surface>
</gml:surfaceMember>
<gml:surfaceMember>
<gml:Surface>
<gml:patches>
<gml:Triangle>
<gml:exterior>
<gml:LinearRing>
<gml:posList>200100 5500100 100 200000 5500100 100 200050 5500050 125</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:Triangle>
</gml:patches>
</gml:Surface>
```

Figure 5.15. Part of the GML file with gml:Triangle



```
<gml:CompositeSurface gml:id="ground" srsName="EPSG:4326" srsDimension="3">
<gml:surfaceMember>
<gml:Surface>
<gml:patches>
<gml:PolygonPatch>
<gml:exterior>
<gml:LinearRing>
<gml:posList>200000 5500000 100 200100 5500000 100 200050 5500050 125 200000 5500000 100</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:PolygonPatch>
</gml:patches>
</gml:Surface>
</gml:surfaceMember>
<gml:surfaceMember>
<gml:Surface>
<gml:patches>
<gml:PolygonPatch>
<gml:exterior>
<gml:LinearRing>
<gml:posList>200100 5500000 100 200100 5500100 100 200050 5500050 125 200100 5500000 100</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:PolygonPatch>
</gml:patches>
</gml:Surface>
</gml:surfaceMember>
<gml:surfaceMember>
<gml:Surface>
<gml:patches>
<gml:PolygonPatch>
<gml:exterior>
<gml:LinearRing>
<gml:posList>200100 5500100 100 200000 5500100 100 200050 5500050 125 200100 5500100 100</gml:posList>
```

Figure 5.16. Part of the GML file with gml:PolygonPatch

GML file was first uploaded in QGIS 2.18., the color was added to the TIN (Figure 5.16). After that, the file is saved again in gml format and uploaded in FME Data Inspector to be displayed in 3D (Figure 5.18).

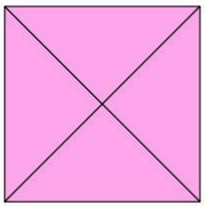


Figure 5.17. TIN representation in 2D

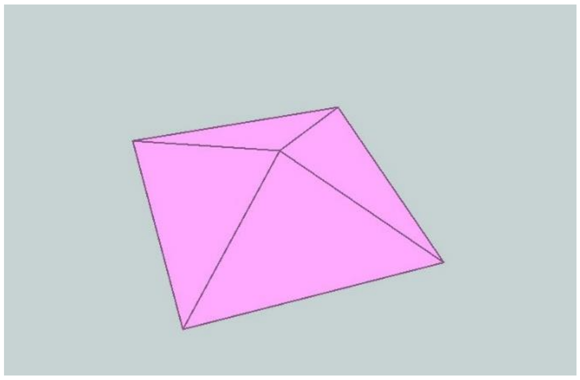


Figure 5.18. TIN representation in 3D

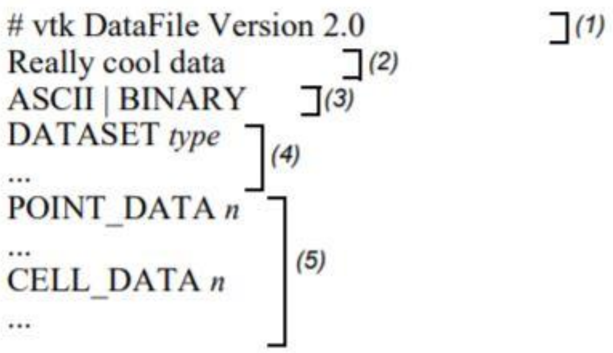
5.5 VTK file format

VTK is a data directory that contains examples of the VTK file format used by the Visualization Toolkit. The Visualisation Toolkit is a software system for 3D computer graphics, image processing and visualization on desktop, mobile and web. VTK is written in C++. It can render data in a web browser (URL 23).

In VTK two different styles of file formats are available. The simplest are the legacy, serial formats that are easy to read and write by hand or programmatically. However, these formats are less flexible than the XML based file formats. The advantages of the XML formats are that they support random access, parallel I/O and portable data compression.

The legacy VTK file formats have five basic parts (Figure 5.19.):

1. Identifier and file version,
2. Header
3. File format – ASCII or binary,
4. Dataset structure – geometry and topology datasets,
5. Dataset attributes.



```
# vtk DataFile Version 2.0
Really cool data
ASCII | BINARY
DATASET type
...
POINT_DATA n
...
CELL_DATA n
...
```

Figure 5.19. Overview of five parts of VTK data file formats (URL 21)

The Visualization Toolkit supports five dataset formats, and those are structured points, structured grid, rectilinear grid, unstructured grid and polygonal data. Triangles and their combinations belong to the unstructured grid and polygonal data. The unstructured grid dataset has an arbitrary combination of any possible cell type. These grids are defined by points, cells and cell type.

Also, there are some dataset attributes as scalars, vectors, normals, texture coordinates, tensors and field data.

Another set of data formats uses XML syntax. They are more complicated than the original VTK, but support many more features. The main reason for their development was to facilitate data flow and parallel I/O. Some format features include support for compression, portable binary encoding, random access, a new extension for different VTK data types. Two different types of VTK XML data files are parallel and serial. Serial data file type is designed to read and write in single-action applications. That means that all data is in one file. Parallel data file is designed to read and write in programs with multiple parallel processes. The data set is divided into parts.

In XML format, VTK datasets are divided into two categories. The first category is structured datasets. Here belongs topologically regular array of cells such as pixels and voxels or quadrilaterals and hexahedra. The second datasets are called unstructured datasets. These datasets form a topologically irregular set of points and cells. TIN belongs to this category.

PARAVIEW is a 3D graphics program that can read VTK files and display data. Also, applications such as Molekel, VisIt, VisTrails, MOOSE, 3D Slicer, MayaVi, and OsiriX use VTK.

5.5.1 Coordinate System Support

VTK can represent a position in a variety of coordinate systems and convert the position to other coordinate systems, but it does not support a geodetic coordinate system (URL 22).

5.5.2 TIN representation of VTK format

In VTK format, vertices can be classified into five classifications (Figure 5.20.):

- Simple vertex (A) – every edge is shared by two triangles,
- Complex vertex (B) – edges can be shared by more than two triangles,

- Boundary vertex (C) – vertex is surrounded by a semi-circle of a triangle,
- Interior vertex (D) – vertex which is shared by exactly two edges, classified as feature edges,
- Corner vertex (E) – shared by more than two feature edges.

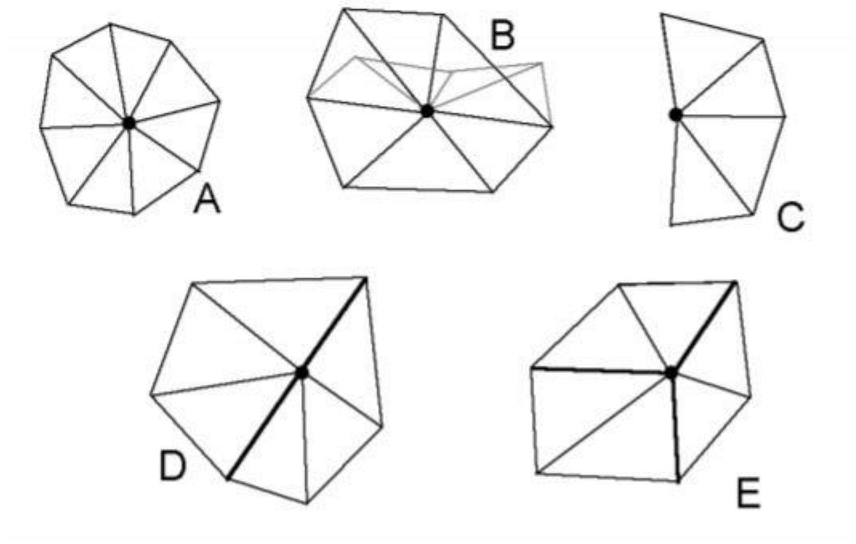


Figure 5.20. Vertex classifications (Knapp, 2002)

Delaunay triangulation is used to construct a topology from unstructured point data. Triangles (i.e. an unstructured grid or polygonal datasets) are generated in two dimensions, while tetrahedra (i.e. an unstructured grid) are generated in three dimensions. If elevated points (z value) are added to the 2D triangulation, a 2.5D Delaunay triangulation will be obtained.

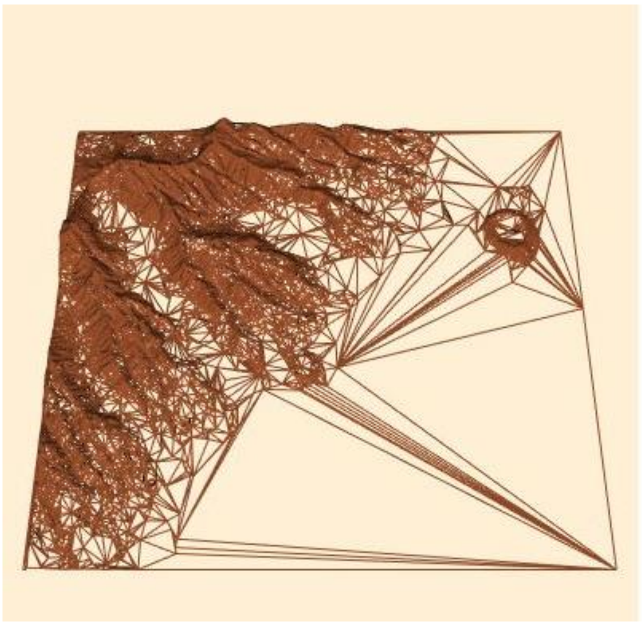


Figure 5.21. Presentation of a terrain model as a triangle mesh (URL 38)

Figure 5.21. shows a terrain model of Honolulu, Hawaii as a triangle mesh.

5.5.3 LOD and support of tiling concept

To display different levels of details is called vtkLODActor. It is storing multiple levels of detail (LOD) and can automatically switch between them. The highest level is just the normal data. The second level is a cloud of points of a certain number of points that are randomly sampled from the map input. Attributes of points are copied over to the point cloud. The lowest level of detail is a simple bounding box outline of the actor. These two lower levels of detail are achieved by creating instances of vtkOutlineFilter (low resolution) and vtkMaskPoints (medium resolution). Additional levels of detail can be added using the AddLODMapper () method (URL 23).

VTK does not support the tiling concept.

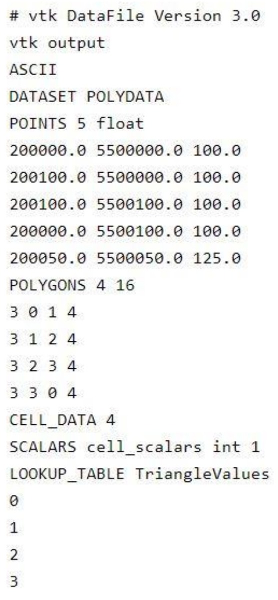
5.5.4 Attributive thematic data and metadata

VtkDataSetAttributes is a class used to represent and manipulate attribute data, such as scalars, vectors, normals, texture coordinates, global IDs and field data. This adds to vtkFieldData the ability to select one of the fields from the field as the currently active string for each attribute type. In addition, vtkDataSetAttributes provides methods that filters call for data forwarding, data copying, and field interpolation. Datasets attributes are information associated with the structure of dataset. In VTK, datasets are supported for both points (point attribute data) and cells (cell attribute data). Attribute data, together with the data set structure, is processed by the many VTK filters to generate new structures and attributes.

Metadata include queries such as number of columns, their names and their data types.

5.6 Sample file

Figure 5.22. shows sample file which is created in Notepad++. VTK format can display large coordinates, although there is no geodetic coordinate reference system (CRS) support.



```
# vtk DataFile Version 3.0
vtk output
ASCII
DATASET POLYDATA
POINTS 5 float
200000.0 5500000.0 100.0
200100.0 5500000.0 100.0
200100.0 5500100.0 100.0
200000.0 5500100.0 100.0
200050.0 5500050.0 125.0
POLYGONS 4 16
3 0 1 4
3 1 2 4
3 2 3 4
3 3 0 4
CELL_DATA 4
SCALARS cell_scalars int 1
LOOKUP_TABLE TriangleValues
0
1
2
3
```

Figure 5.22. Sample file of VTK format

Figure 5.23. shows a visualization of this sample file in Paraview. Four triangles are shown in different colors, together forming a TIN.

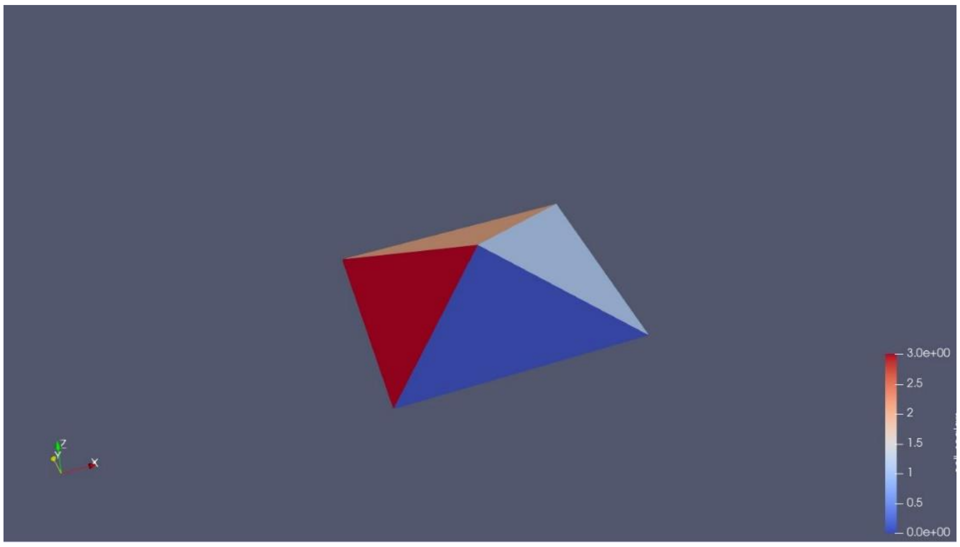


Figure 5.23. File visualization

5.7 Indexed 3D Scene Layers (I3S)

The Indexed 3D Scene Layers is an Open Geospatial Standard (OGC) for storage and transmission of large, heterogeneous 3D geospatial datasets. It is designed to be cloud, web and mobile-friendly. I3S supports various coordinate systems along with a lot of layer types (Reed and Belayneh, 2020).

This format can be used for the representation of different types of 3D data. The following layer types have been specifying and those are 3D Objects, Integrated Meshes, Point Features and Point Clouds.

I3S and the corresponding Scene Layer Package format (*.slpk) fulfil the following features:

- User Experience first – high interactivity and fast display,
- Scalability – support large layers of the scene, of global proportions and many detailed features,
- Reusability – it is a delivery, storage and exchange format,
- Level of detail – multiple levels of detail,
- Distribution – enable distribution of very large data sets,
- User-controllable symbology – effective rendering of symbology on the client-side,
- Extensibility – supporting new layers, geometry types and new platforms,
- Web friendliness – use JSON and other web standards
- Compatibility – structure is compatible across web, mobile and desktop clients.

When it comes to selecting the Coordinate Reference Systems (CRS), I3S have to fulfil some requirements:

- Reduce the need for client-side re-projection,
- Support datasets globally,

- Easy rendering in coordinate reference systems for projected CRSs and geographic CRSs as well
- Support local and global data with very high positioning accuracy.

I3S uses a node-based hierarchal spatial index structure for organizing information, in which each node can contain features with associated geometry, textures and attributes. The role of the index is to enable fast access to blocks of relevant data. Node data is stored in multiple individually available resources. The node index document captures the Bounding Volume Hierarchy (BVH) tree topology for the node. It includes information such as the node's bounding volume information, metadata for LOD and parent-child relationships. Each node includes a set of information that covers the nodes below it and is part of the path of the leaf nodes below it. Internal nodes can have a reduced representation of the information contained in the descendant nodes.

Node information is modelled using a set of resources including NodeIndexDocument, FeatureData, Geometry, Attributes, Texture and SharedResource. All of this together represents a set of features or data elements for a given node. These resources are always attached to the node.

The node structure may be:

- 'expanded' – with complete meta-information about node's position and topology,
- 'fixed-size' – in support of a „paged“ access pattern (many nodes together).

All types of Scene Layer use the same basic set of geometry types: points, lines and triangles. Array Buffer View geometry property declarations control geometries storage and consumption representation. I3S provides complete control over these properties, such as the arrangement of components by vertex (positions, normals, texture coordinates).

5.7.1 TIN representation in I3S format

I3S supports triangle mesh storage via geometric triangle type. 3D Object and Integrated Mesh layer type represent geometry as triangle meshes using the mesh-pyramids profile. The mesh-pyramids profile uses the triangles geometry type to store triangle meshes with a reduced level of detail representations of the mesh, segmented by features, available in the interior nodes.

Triangular Irregular Network (Figure 5.24.) is suitable for engineering applications and provides interactive capabilities. It can present a terrain with multi-resolution and has robust support for handling large amounts of data. Mass points are measurements that are used for triangulation.

TIN may be constructed from point clouds.

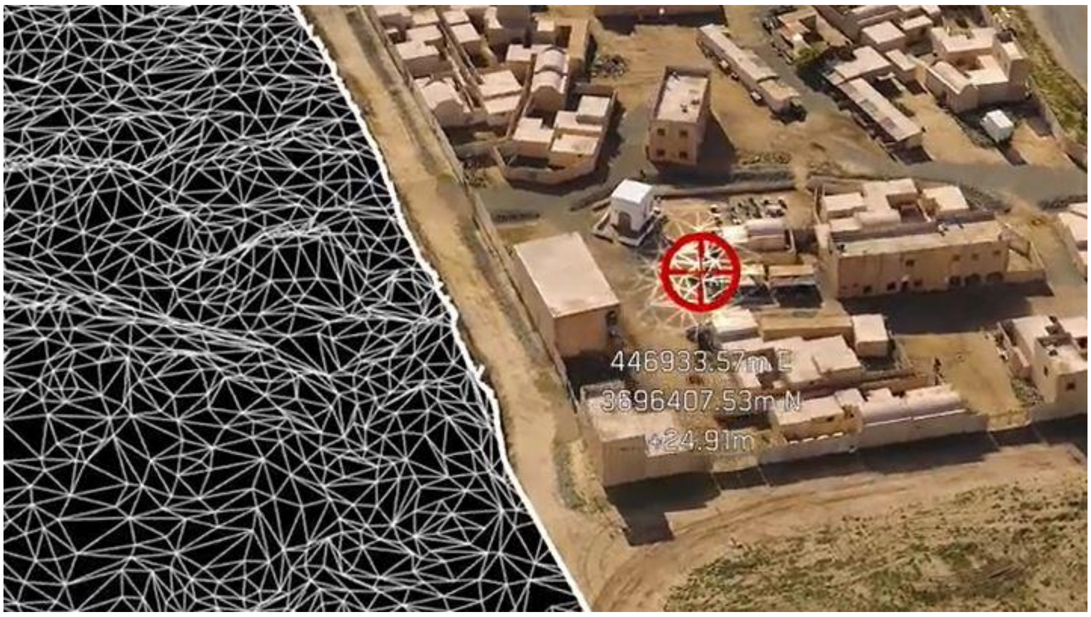


Figure 5.24. TIN representation in I3S format *(URL 33)*

5.7.2 LOD and support of tiling concept

Level of Detail concept is important to the I3S standard. Levels of detail can be selected based on different metricTypes. LOD can be switched based on the screen size of the node, based on the screen scale of the node, based on normalized distance of the node from the camera or based on an estimate of point density covered by the node. In GIS, LODs usually refer to maps defined in given scales and resolutions. Greater fidelity is provided by higher levels of detail. Scene layers support detail levels in a way that preserves the identity of individual features that are retained within any level of detail. Levels of detail can be used to separate heavy features, thin or cluster for better visualization, and to integrate files with external authorship.

I3S has a discrete LOD approach. This means that different levels of detail are connected with different levels of the index tree. The level of detail will be lower as the node are closer to the root. For each subsequent lower level, the amount of data is usually reduced by applying methods such as texture sampling, feature reduction, generalization, grid reduction generalization, grouping, or thinning, so that all internal nodes also have a balanced weight. Generalization refers to the scene layer as a whole, and the number of discrete detail levels for the layer corresponds to the number of levels in the index tree for the scene layer (URL 32).

The Level of detail is here analogous to the standard raster and vector tiling schemes. A node in the I3S scene layer tree could be considered the analogue of a tile in a raster or vector tiling scheme.

5.7.3 Attributive thematic data and metadata

AttributeData are tabular data associated with features that are used for attributed-based display or symbol representation. They are presented in a structured binary form defined in the I3S/SLPK standard. Attributes are used for displaying data and for scaling the displayed model.

I3S supports the two types of access to attribute data:

1. from optional paired services that expose RESTful endpoints that can query and provide direct access to dynamic source data, including attributes,
2. from fully cached attribute data in binary form within the I3S store, attribute values are stored as aligned geometries per field.

Metadata about each attribute resource is available to clients through the scene service layer. When attributes are present in the scene cache, the resourcePattern string in the layer store will contain a value called Attributes, indicating that the attributes are a required resource used for symbolization and display. This metadata allows clients to initialize and allocate all necessary resources on the client side before accessing any interesting attributes.

5.8 Industry Foundation Classes (IFC)

IFC is an ISO open standard format for importing and exporting building objects and their properties. It is also a common data model. This file format provides interoperability between different software applications. BuildingSMART International develops specifications for IFC as its Data Standard. IFC files can be read and edited by any BIM software. IFC files are compatible with Autodesk's Revit, Tekla's BIMsight software, Adobe Acrobat, FME Desktop, Constructivity Model Viewer, CYPECAD, SketchUp, GRAPHISOFT's ARCHICAD, and more (URL 26).

IFC models contain a structure that is a combination of geometric and non-geometric (semantic) data. This data can be displayed, analysed and modified in different ways in multiple software applications.

There are different versions of the scheme. Currently, the most used version is IFC2X3 but expect version IFC4 to gain adoption in the coming years.

Geometry of IFC file format can be defined as a network of polygons (triangles). IFC describes not only individual components but also their interrelationship. For each object, there is the possibility of adding user parameters (attributes) as well as user-defined geometry, attached documents and other references to external information sources (URL 27).

5.8.1 Coordinate System Support

IFC provides entity called IfcCoordinateReferenceSystem. This entity allows defining the name of the coordinate reference system, geodetic datum and vertical datum. While IfcCoordinateReference System is an abstract class, its subclass IfcProjectedCRS is used to define a concrete projected coordinate reference system.

This standard does not provide the calculations required for proper mapping to and from a Cartesian coordinate system into a geodetic coordinate system. In infrastructure design and surveying are mostly used map grid coordinates (easting, northing). BIM tools might have problems with these large coordinates. Therefore, the IFC standard allows referencing of the locally defined coordinate system to the global coordinate system. The advantage of using a local coordinate system is that it allows using shorter specified coordinates.

IfcMapConversion entity is used to define a local coordinate system concerning the global coordinate system. It provides atributes SourceCRS and TargetCRS.

Elements are typically modelled in local coordinate systems, which are defined by a hierarchical set of transformations based on entities defining local systems (IfcLocalPlacement), axes (IfcAxis2Placement), and 2D / 3D vectors (IfcDirection). However, global coordinates can be obtained using the georeferencing information that is sometimes included in IFC files, such as information with the latitude, longitude and elevation in IfcSite.

It is possible to define more than one geodesic reference system within an IFC model, but it is not recommended.

5.8.2 TIN representation in IFC

Terrain surface can be described using an entity called IfcTriangulatedIrregularNetwork. The way triangles are generated in TIN differs between software products, so the final recorded triangle is the only way to ensure that the TIN can be accurately transmitted between the software product. IfcTriangulatedIrregularNetwork efficiently provides enough data for the reconstruction of equivalent surfaces in a variety of software products, including each 3D point and triangular surface. Triangles are represented at IFC as an indexed set, where the indices refer to existing points. Consequently, the point instances can be referenced by various other entities. In IfcTriangularIrregularNetwork all the 3D points must be unique in the XY plane. All faces have to have the same direction. It could be counter-clockwise. In the XY plane faces should not overlap. Boundaries of continuous faces may touch at one or more common points, but they may not cross (Borrmann et al, 2017).

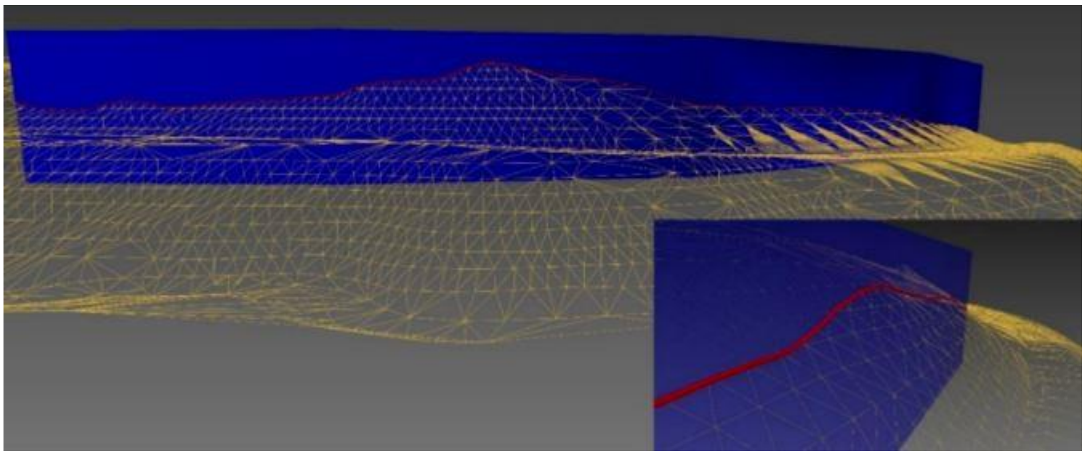


Figure 5.25. TIN representation of the terrain (brown), (URL 34)

Figure 5.25. illustrates the terrain using Triangular Irregular Network.

5.8.3 LOD and support of tiling concept

IFC file format does not support Level of Detail. It only supports Level of Development (LODt), but it is not included in the IFC standard. LODt does not have a standard definition. Unlike LOD, LODt is local while LOD is global. That is, LODt is specific to each object and LOD is specific to the project (Tolmer et al., 2013).

This file format also does not support the tiling concept.

5.8.4 Attributive thematic data and metadata

Each IFC model consists of IFC entities built in a hierarchical order. Each IFC entity has a fixed number of IFC attributes and any number of additional IFCs properties. The main identifiers of the entities are the IFC attributes. The names of these attributes are fixed and defined by buildingSMART as part of the IFC standard code.

IFC classes have associated direct attributes that can indicate a connection to some other object or could simply be attached as a simple attribute of the data type, such as integer, string or logical. Therefore, IFC models make a difference between attributes that are directly connected to the object as entity attributes and attributes assigned to indicate a relationship to other objects.

Figure 5.26. shows that each IFC class can have simple data attributes with referenced object attributes as in the case of IfcRoot, or a referenced object with relationship attributes as in the case of IfcProduct, or entity attributes and relationship attributes together as in the case of IfcObject.

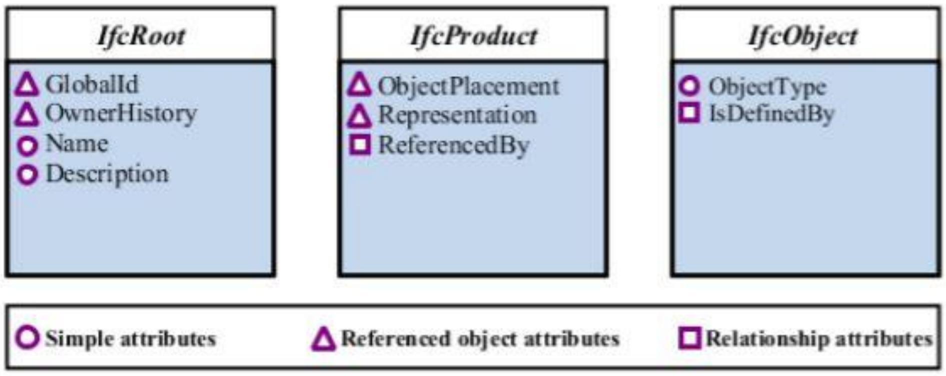


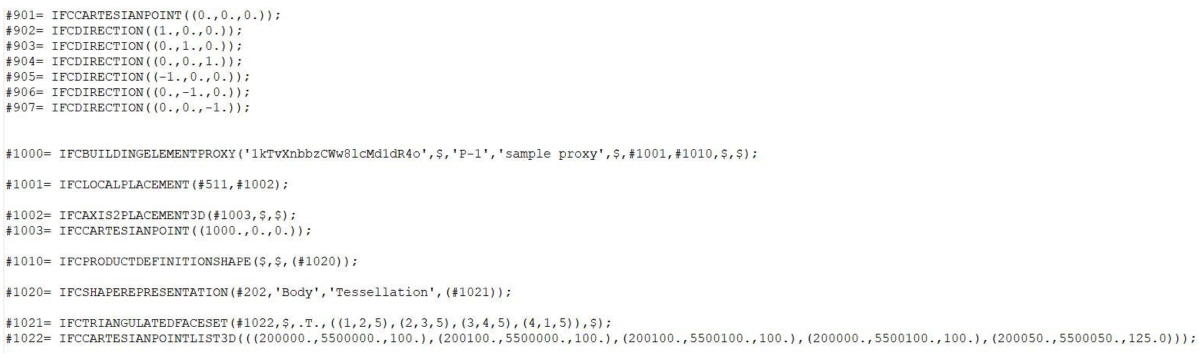
Figure 5.26. IFC attributes (Ismail et al, 2017)

Metadata is used extensively in IFC, but different software does not use it consistently. As basic information, IFC provides special entities for the IFC file header (FILE_DESCRIPTION, FILE_NAME, FILE_SCHEMA), as well as certain entities in the body files, such as IfcOrganization, IfcPerson, etc. Additional information is usually added via a reference to an external document. Reference records external file metadata, and IFC file metadata is located in an external document.

5.8.5 Sample file

In IFC format TIN is created using entity type IFCTRIANGULATEDFACESET. Although IFC has limited support for a geodetic coordinate system, it can support large coordinates.

Figure 5.27 shows part of the written code.



```
#901= IFCCARTESIANPOINT((0.,0.,0.));
#902= IFCDIRECTION((1.,0.,0.));
#903= IFCDIRECTION((0.,1.,0.));
#904= IFCDIRECTION((0.,0.,1.));
#905= IFCDIRECTION((-1.,0.,0.));
#906= IFCDIRECTION((0.,-1.,0.));
#907= IFCDIRECTION((0.,0.,-1.));


#1000= IFCBUILDINGELEMENTPROXY('1kTvXnbbzCWw8lcMd1dR4o',$,'P-1','sample proxy',$,#1001,#1010,$,$);

#1001= IFCLOCALPLACEMENT(#511,#1002);

#1002= IFCAXIS2PLACEMENT3D(#1003,$,$);
#1003= IFCCARTESIANPOINT((1000.,0.,0.));

#1010= IFCPRODUCTDEFINITIONSHAPE($,$,(#1020));

#1020= IFCSHAPEREPRESENTATION(#202,'Body','Tessellation',(#1021));

#1021= IFCTRIANGULATEDFACESET(#1022,$,.T.,((1,2,5),(2,3,5),(3,4,5),(4,1,5)),$);
#1022= IFCCARTESIANPOINTLIST3D(((200000.,5500000.,100.),(200100.,5500000.,100.),(200100.,5500100.,100.),(200000.,5500100.,100.),(200050.,5500050.,125.0)));
```

Figure 5.27. Part of the code

Figure 5.28 shows a visualization of a simple TIN in a program called usBIM.viewer+. One color is added to the whole TIN and it is made in this program.

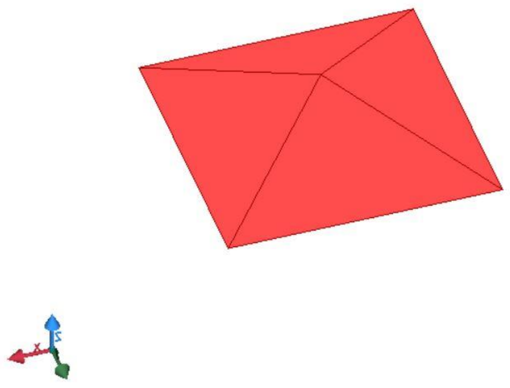


Figure 5.28. TIN visualization

5.9 AutoCAD Drawing Exchange File (DXF)

The DXF file format is an open-source, uniquely structured format created by Autodesk for 2D and 3D models and drawings. This file format is mostly used to export modelling data between various CAD programs. It can be processed as a raster or vector file type (URL 28).

It is a text file consisting of a string of ASCII characters and codes. The DXF file is a complete language that describes how the CAD program reads and displays drawings. These files are highly compressed, device and software-independent and can include 3D models (URL 29).

The DXF file has sections and they are (Figure 5.29.):

- Header: It always comes first in the DXF file. It has code to identify the DWG format version. Other variables include default units and styles for the drawing, drawing size limits, the coordinate system for spatial positioning, a project name, etc.

- Classes: It contains information related to application-defined or custom classes whose instances (objects) appear in other sections.
- Tables: it contains various tables that support the functionality of the CAD application and displays the contents of the drawings.
- Blocks: It contains an entry for each block in the drawing.
- Entities (graphical objects): Includes entity types such as line, circle, vertex, shape, polyline, etc. Entity objects may contain references to related objects.
- Objects (non-graphical): Contains dictionaries, settings, etc. which support the drawing and rendering functions in CAD application.
- Thumbnail Image: An additional section is used if the preview image is saved for use in directory lists or similar application menus.

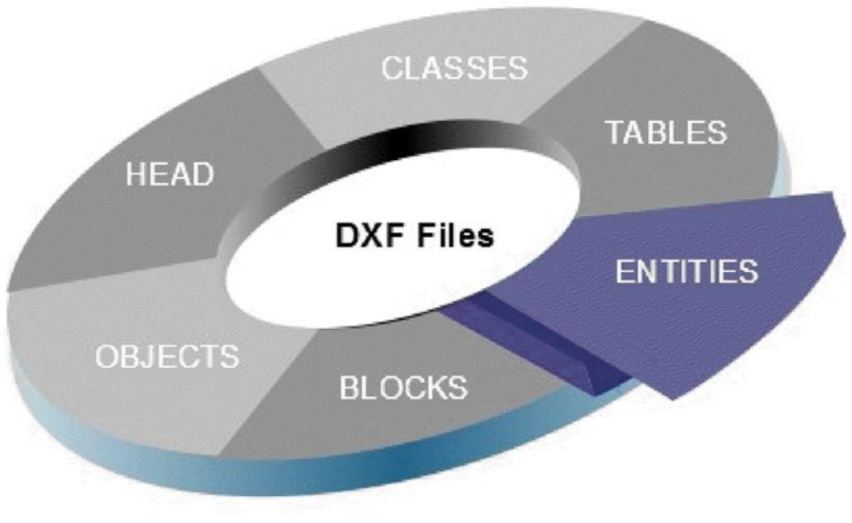


Figure 5.29. Components of DXF file (URL 35)

This file format can provide an exact copy of DWG drawings and, unlike DWG files, it can also be used in third-party software. DXF was made for AutoCAD with multi-platform compatibility. It also works with GIS programs. However it is not designed for the web, but some vendors have provided visualization tools over the Internet.

The DXF is difficult to interpret because it contains many different types of data. However, it is structured so that developers and programs can read the sections they need and skip elements they do not need.

Since DXF file format has been designed as a drawing Interchange format, it does not support thematic attributes. Also, this format has no support for semantic, topology, texture, LOD and coordinate reference systems. It only supports projected coordinates.

5.9.1 TIN representation in DXF

In AutoCAD software TIN can be made using the command TINSurface. The TIN surface can be saved in a DXF file as TIN 3D Faces. DXF files support two types of three-dimensional triangle objects, 3D Faces and Polylines.

The command Export DXF 3D Face File/TIN File allows the user to export any loaded elevation data sets to a DXF 3D Face file. After selecting this command, the user may choose options such as Tiling panel, Export Bounds panel, etc (URL 36).

This file can be used to store 3D triangulation and it can be stored using ASCII or a binary code, but the problem is that it is not effective, rarely georeferenced and does not support a clean topology of shared edges.

Figure 5.30. shows Triangular Irregular Network in AutoCAD Civil 3D sofware.

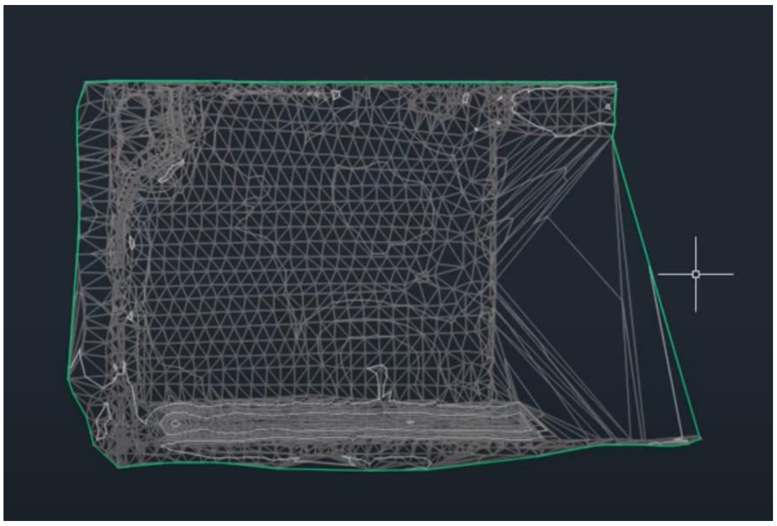


Figure 5.30. TIN representation in AutoCAD Civil 3D (URL 37)

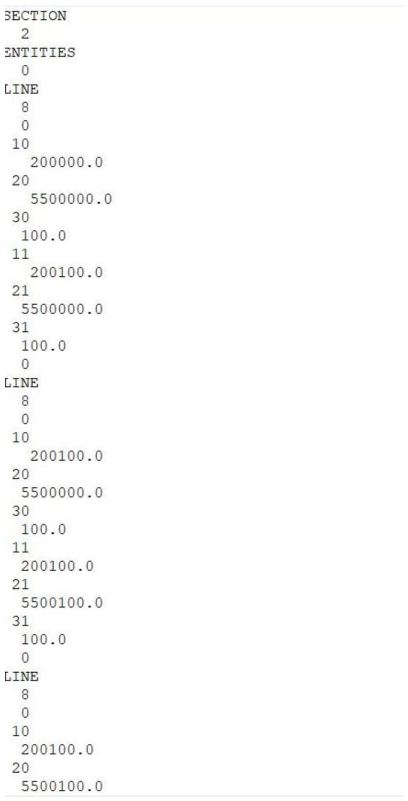
5.9.2 LOD and support of tiling concept

The DXF does not support Level of Detail.

This format supports the tiling concept. This process makes it easy to work with large amounts of data. It splits the model into multiple tiles, and a combination of tiles represents the whole project.

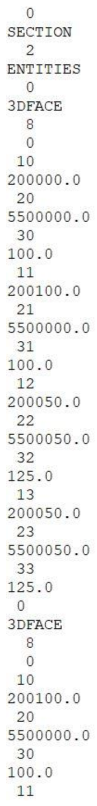
5.9.3 Sample file

DXF-based TIN model (Figure 5.33) was obtained in the way it was first written in ASCII code in two ways, as lines (LINE) of which part is shown in Figure 5.31 and as surface (3DFACE) which part is shown in Figure 5.32. The second one is imported into the SketchUp program, color is added and then DXF is exported (Figure 5.34). Color can be added to each triangle individually (Figure 5.33). DXF format can display large coordinates, although there is no geodetic coordinate reference system (CRS) support.



```
ECTION
  2
NTITIES
  0
LINE
  8
  0
 10
   200000.0
 20
   5500000.0
 30
  100.0
 11
   200100.0
 21
  5500000.0
 31
  100.0
  0
LINE
  8
  0
 10
   200100.0
 20
  5500000.0
 30
  100.0
 11
  200100.0
 21
  5500100.0
 31
  100.0
  0
LINE
  8
  0
 10
  200100.0
 20
  5500100.0
```

Figure 5.31. Part of the written code using the LINE function



```
  0
SECTION
  2
ENTITIES
  0
3DFACE
  8
  0
 10
200000.0
 20
5500000.0
 30
100.0
 11
200100.0
 21
5500000.0
 31
100.0
 12
200050.0
 22
5500050.0
 32
125.0
 13
200050.0
 23
5500050.0
 33
125.0
 0
3DFACE
  8
  0
 10
200100.0
 20
5500000.0
 30
100.0
 11
```

Figure 5.32. Part of the written code using the 3DFACE function

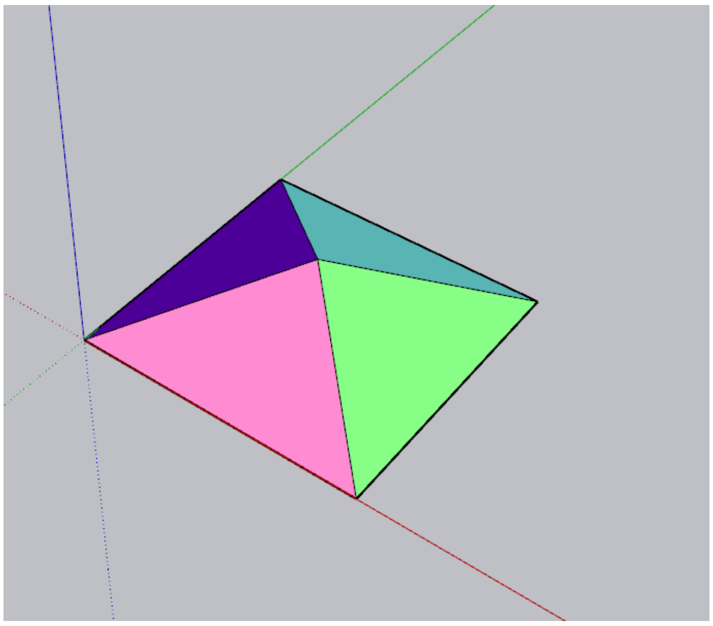
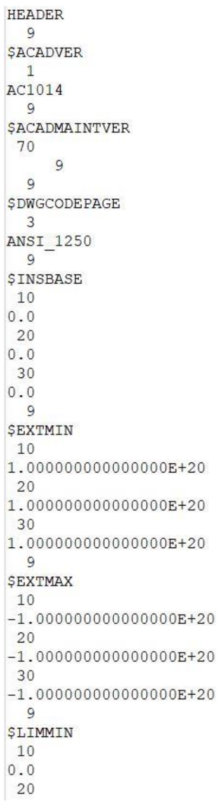


Figure 5.33. Visualization of the TIN in SketchUp



```
HEADER
  9
$ACADVER
  1
AC1014
  9
$ACADMAINTVER
 70
     9
  9
$DWGCODEPAGE
  3
ANSI_1250
  9
$INSBASE
 10
0.0
 20
0.0
 30
0.0
  9
$EXTMIN
 10
1.000000000000000E+20
 20
1.000000000000000E+20
 30
1.000000000000000E+20
  9
$EXTMAX
 10
-1.000000000000000E+20
 20
-1.000000000000000E+20
 30
-1.000000000000000E+20
  9
$LIMMIN
 10
0.0
 20
```

Figure 5.34. Part of the exported code

5.10 Wavefront object files (.obj)

Wavefront OBJ is an open file format used for representing 3D geometry. This format has wide software support in 3D modelling and computer graphics. It was originally developed by

Wavefront Technologies for its software, and has been widely accepted by 3D graphics applications such as Autodesk Maya, Blender, MeshLab and SketchUp (URL 30). This file is usually an end product of the 3D modeling process generated by the CAD.

The OBJ file format consists of multiple lines, each of which contains a key and various values. The key on each line indicates the type of information to be followed. Therefore, the OBJ file format does not require a header. Table 3. shows some keys that can be used.

Table 3. OBJ file keys (McHenry and Bahcsy, 2008):

Key	Description
#	Comment
v	Vertex
l	Line
f	Face
vt	Texture coordinate
vn	Normal
g	Group

Object files can be in ASCII format or binary format. This file format supports polygonal objects and free-form objects. Polygonal objects are points, lines and faces, and free-form geometry uses curves and surfaces.

Counter-clockwise is the default order to store vertices, avoiding explicit declaration of face normals. To encode the model surface geometry, the file stores vertices and normal to each polygon. Although OBJ files declare scale data in the comment line, units for OBJ coordinates have not yet been declared.

Although CRS is not defined in OBJ file format, due to its ASCII nature, the coordinates of the vertices can receive a very large number, and the global coordinates are not a problem.

5.10.1 TIN representation in OBJ

OBJ format contains a common structure of vertex data, storing a table of vertices and an indexed list of faces that encode faces as index triples to vertices. This type of data storage is very effective and allows easy rendering.

DEM as a TIN model can be provided in Wavefront OBJ format (Figure 5.35.).

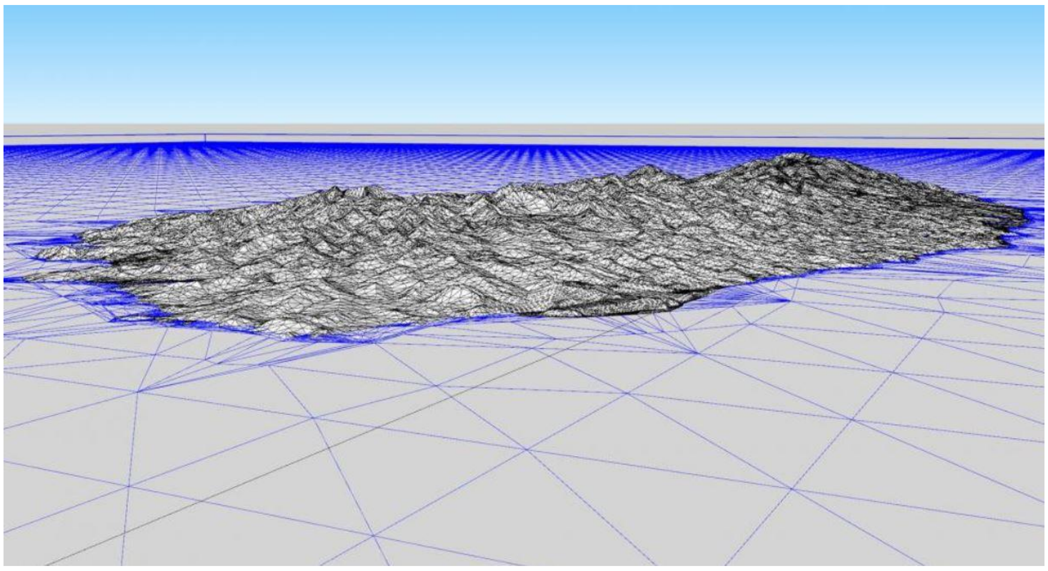


Figure 5.35. Terrain model (URL 39)

5.10.2 LOD and support of tiling concept

The level of detail belongs to display and render attributes. These attributes describe how an object looks when it is displayed in Model and PreView.

The level of detail gives the control of which elements of an object will be displayed while working in PreView. The level is the level of detail to display. When the level of detail is set to 0 or omitted then all the elements are displayed. Specifying an integer between 1 and 100 sets the level of detail that will be displayed when reading the OBJ file.

The OBJ file format allows the user to tile the surface of the 3D model.

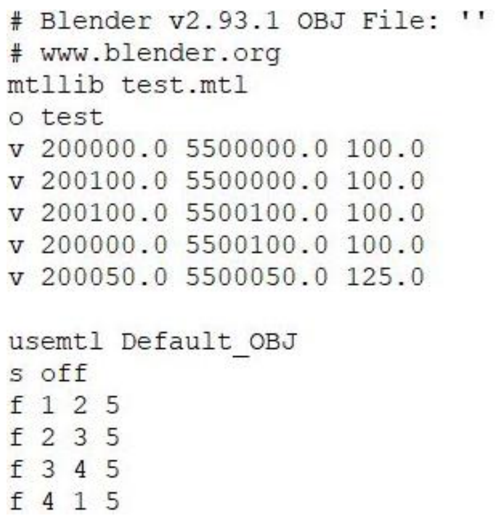
5.10.3 Attributive thematic data and metadata

Storing attributive data such as color and texture to each face is possible in a companion file format called Material Template Library (.mtl). When rendering a 3D model, each surface point is assigned a coordinate from a 2D image that has attributes. First are mapped vertices of the mesh and then other points are assigned coordinates by interpolating between the coordinate of the vertices.

Attribute support is limited in OBJ and its semantics are very weak. Therefore, it is best to easily transfer static network geometry between applications.

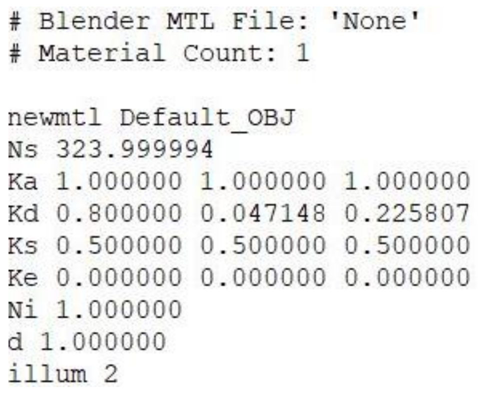
5.10.4 Sample file

Wavefront OBJ format consists of two files, .obj and .mtl. As can be seen from Figure 5.36., the OBJ format although not supported by CRS can display large coordinates. Color data is stored in the second file (Figure 5.37).



```
# Blender v2.93.1 OBJ File: ''
# www.blender.org
mtllib test.mtl
o test
v 200000.0 5500000.0 100.0
v 200100.0 5500000.0 100.0
v 200100.0 5500100.0 100.0
v 200000.0 5500100.0 100.0
v 200050.0 5500050.0 125.0

usemtl Default_OBJ
s off
f 1 2 5
f 2 3 5
f 3 4 5
f 4 1 5
```

Figure 5.36. .obj file



```
# Blender MTL File: 'None'
# Material Count: 1

newmtl Default_OBJ
Ns 323.999994
Ka 1.000000 1.000000 1.000000
Kd 0.800000 0.047148 0.225807
Ks 0.500000 0.500000 0.500000
Ke 0.000000 0.000000 0.000000
Ni 1.000000
d 1.000000
illum 2
```

Figure 5.37. .mtl file

The visualization is made in a program called Blender (Figure 5.38). In this format is not possible to add color to each triangle separately, but the whole TIN together.

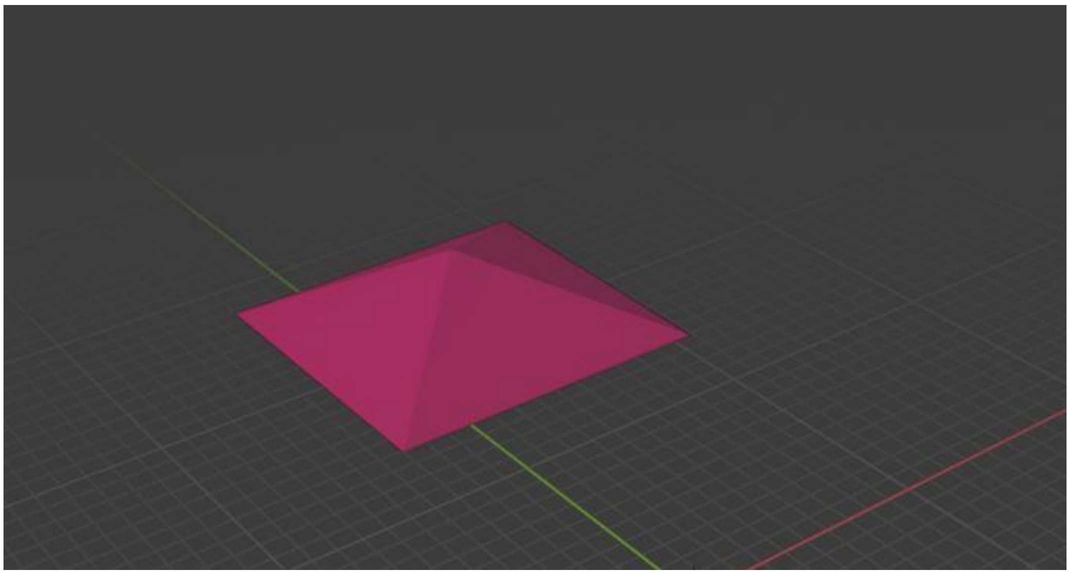


Figure 5.38. File visualization in Blender

6. COMPARISON OF 3D FILE FORMATS

The comparison of 3D file formats is performed based on fifteen criteria, which are relevant for the best possible representation of a 3D terrain model using an irregular triangle network.

Table 4. Comparison of 3D file formats

	VRML	**X3D**	**CityGML**	**GML3**	**VTK**	**I3S**	**IFC**	**DXF**	**OBJ**
Ease of use	easy	quite easy	relatively easy	relatively easy	easy	relatively easy	relatively easy	easy	quite easy
Popularity of usage	not popular	not popular	quite popular	not very popular	not supported by many tools	quite popular	popular	popular	widely-used
Support from common software products	CAD (Autodesk 3DS), Cortona3D-Viewer, FreeWRL, OpenVRML, Microsoft 3D Builder	Blender, Project Wonder-land	CityEditor, Bentley Map, BS Contact Geo, CityGRID, CityServer, GIS	FME Desktop, Mekaartor, TatukGIS Viewer, yED Graph Editor, Garden Planner, Gaia3, Falcon View, AutoCAD, ArcGIS	ParaView, Unity, Molekel, VisIt, VisTrails, MOOSE, 3D Slicer, MayaVi, and OsiriX	Vricon, Pix4Dm Bentley, ArcGIS, Cesium	Revit, BIMsight software, Adobe Acrobat, FME Desktop, Construct-ivity Model Viewer, CYPE-CAD, SketchUp, GRAPHI-SOFT's ARCHI-CAD, usBIM. viewer+	AutoCAD, Blender, CINEMA 4D, Maya, Autodesk Viewer, DWG TrueView, eDrawings Viewer, ShareCAD, Turbo-CAD, CorelCAD, Adobe Illustrator, SketchUp, Revit, Tekla	Autodesk Maya, Blender, MeshLab, SketchUp, 3ds Max, Cesium
Standardization	ISO	ISO, Web3D. org	OGC	OGC	-	OGC	ISO	Commer-cial (Autodesk)	-
„Web-enabling“	supported	supported	supported	supported	not supported	supported	not supported	not supported	supported
Possibility to add attributive thematic data and metadata (to triangles or to the TIN itself)	to triangles	to triangles	to TIN	to TIN	to triangles	to triangles	to TIN	to triangles	to TIN
Support of tiling concepts	supported	supported	not supported	not supported	not supported	supported	not supported	supported	supported
LOD	basic support	basic support	LOD 0-4 (5)	not supported	multiple LODs	multiple LODs	not supported	not supported	basic support
3D or 2.5D model	3D	3D	3D	3D	3D	3D	3D	3D	3D
Geodetic coordinate reference system (CRS) support	not supported (need extension for support)	support limited set of CRS	supported	supported	not supported	supported	supported (limited)	not supported	not supported
Ability to share vertices in TINs	has ability	has ability	has ability	has ability	has ability	has ability	has ability	has ability	has ability
Vertex ordering	clockwise and counter-clockwise	counter-clockwise	counter-clockwise	counter-clockwise	counter-clockwise	counter-clockwise	counter-clockwise	counter-clockwise	counter-clockwise

	VRML	X3D	CityGML	GML3	VTK	I3S	IFC	DXF	OBJ
Kind of encoding	ASCII	XML	XML	XML	ASCII and XML	JSON	ASCII, XML, JSON, STEP	ASCII	ASCII
Possibility to model non-manifold geometries	has possibility	has possibility	has possibility	has possibility	has possibility	no possibility	has possibility	has possibility	has possibility
ASCII/binary format respectively "compressibility"	ASCII and binary	ASCII and binary	ASCII	ASCII	ASCII and binary	binary	ASCII	ASCII and binary	ASCII and binary

The comparison in Table 4. clearly shows that every 3D standard is designed for specific purposes. All formats are easy or at least relatively easy to use. The most popular standards are DXF, IFC, CityGML and they are followed by others. VRML, X3D and IFC are ISO standards, CityGML, GML3 and I3S are OGC standards, and VTK, OBJ and DXF are not standardized. The „Web-enabling“ criterion gives an indication of which standards are designed and optimized for Web use. The table shows that almost all formats, except GML3, IFC and DXF, have web support. CityGML, IFC and GML3 have powerful methods for storing attributes and the first two are rich in semantics, but they can only add attributes to the whole TIN. VRML, X3D and DXF have basic support for adding thematic attributes to triangles. Support of tiling concept have only VRML, X3D, DXF and I3S formats.

VRML, X3D and OBJ formats have basic support of LOD. Multiple LODs have I3S and VTK. In the case of these formats, the LOD represents the level of complexity by which a particular object will be visualized, relatively number of triangles to be displayed. In the case of CityGML, LODs are used to represent how detailed the terrain or building is given, but the problem is there is no geometric and semantic difference between different LODs of a terrain. Also, the browsers do not use CityGML's LODs in the visualization.

All formats are real 3D because for the same x and y it is possible to add multiple values of z. They also have ability to share vertices in the TIN and to model non-manifold geometry beacuse one egde is shared by more than two faces or several faces share a common vertex but no edge. To represent the terrain as an irregular triangular network, the 2.5D model is mainly used, which means that for each x and y only one z value is added. Only CityGML and GML3 are 2.75D models. They are still 2-manifold, i.e. each edge of the TIN belongs to only one or two triangles. When the 2.5D+ model is projected onto a 2D surface, the vertial surfaces are leveled which distorts the geometry of the model. There is no mechanism to remove vertical surfaces while transforming from 3D to 2D.

The geodetic coordinate reference system is not supported by VRML, VTK, DXF and OBJ. The remaining formats either support the full or limited number of CRSs. All formats have vertex ordering in a counter-clockwise direction, but VRML can also have in a clockwise direction. X3D, CityGML and GML3 are XML-based formats, VRML, DXF and OBJ are ASCII formats, VTK can be both, I3S has JSON encoding and IFC can be all of the above.

VRML and X3D formats are formats for web-based visualization and representation. They represent 3D interactive vector graphics. DXF and OBJ formats are used for geometrical modeling and visualization. VTK can handle large data sets, to visualize and analyze them. I3S is perfect for storing and representing massive TINs.

DXF, VRML, X3D and IFC support the largest variety of geometries, VRML and X3D are also the best in supporting realistic textures, but with DXF they are the poorest in semantic and attributes storing.

CityGML and GML3 are characteristic for 3D GIS, have powerful methods for storing attributes and georeferenced datasets and are rich in semantics. The disadvantage is that they are not effective for analysis and visualization, i.e. they would only show rough 3D visualization.

IFC file format is characteristic of BIM, primarly used for exchanging building and construction data. It is, as CityGML, rich in semantics. It can be used for GIS, although it is widely accepted for many software.

IFC and CityGML also have very good support of objects, attributes and relationships between the objects.

DXF and IFC are designed to maximize compatibility and minimize conflicts between various software programs, but data loss can still occur during format conversion.

7. FORMAT CONVERSION

Each system has its own proprietary data representation and formats are stored in multiple, incompatible formats. Because of that conversion of 3D file formats exchanging data is complicated. In file formats conversions, geometry and semantic information are lost.

Several common problems occur during the format conversion and those are (McHenry and Bahcsy, 2008):

- missing or inverted faces,
- surfaces and edges do not join (models do not form closed solids),
- incorrect feature orientation in models,
- lines that do not meet at the corners,
- lines that intersect at the corners,
- curves or lines drawn in as many segments of short lines as possible,
- the same feature occurs multiple times in the same place,
- lines or surfaces that coincide with other lines or surfaces,
- surfaces not encountered on the line,
- some of the geometry is not translated,
- geometry, dimensions and notes are not properly separated into different layers,
- geometry of features that are not drawn to scale.

Another problem with data conversation is that each of the formats has its advantages and disadvantages. Each format is made for a specific role, and none of them is made for all. Also, no format is universal for all softwares.

Table 5. shows which specific losses occur during format conversion.

Table 5. Data loss in conversion between different file formats

Target format / **Source format**	**VRML/ X3D**	**CityGML**	**GML3**	**VTK**	**I3S**	**IFC**	**DXF**	**OBJ**
VRML/X3D	-	loss of some geometry and objects	loss-free	loss-free	not support conversion	loss-free	lossfeee	loss of color and material information
CityGML	loss of geographical information, most of semantic and geometries	-	loss of surface materials	not support conversion	loss of LOD, geometry	loss of geographical information, a small number of attributes and semantics	loss of semantic, geographical information	attributes, semantic and thematic support are partialy lost. Geodetic datum is lost.
GML3	loss of geographical information	loss-free	-	not support conversion	not support conversion	not support conversion	loss of attribute dana and geographical information	loss of geographical and some semantic information
VTK	loss of color data	not support conversion	not support conversion	-	not support conversion	not support conversion	not support conversion	loss of color data
I3S	not support conversion	not support conversion	not support conversion	not support conversion	-	not support conversion	not support conversion	not support conversion

Target format / Source format	VRML/ X3D	CityGML	GML3	VTK	I3S	IFC	DXF	OBJ
IFC	loss of some geometric and semantic information	a small loss of semantic	loss-free	loss-free	loss of attribution and metadata	-	loss of attribute data	loss of many geometric and all semantic information
DXF	loss-free	loss-free	loss-free	not support conversion	not support conversion	loss-free	-	loss-free
OBJ	loss-free	loss-free	loss-free	loss-free	not support conversion	loss-free	loss-free	-

7.1 Conversion between VRML/X3D and CityGML

Conversion from CityGML to VRML/X3D lead to a loss of geographical information, most of semantic and geometries, which restricts the aspect of geospatial analysis (Tiwari, 2015).

The physics engine, shapes such as arc, circle, sphere, moving objects, animations and movies are out of range of CityGML. This is the main reason why in the process of converting VRML to CityGML may occur loss of geometry or objects. There should be no data loss when converting TIN.

7.2 Conversion between VRML/X3D and GML3

Conversion from VRML to GML3 is loss-free.

Although VRML has many performance limitations, it has also many advatanges: it is available on almost all platforms, has a wide audience and is cost-effective. For these reasons, GML is convertiable to VRML, although there are some losses such as geographical information (Coltekin and Haggren, 2000).

7.3 Conversion between VRML/X3D and VTK

VTK has the ability to export VRML/X3D file formats using the functions vtkX3DExporter and vtkVRMLExporter. In this conversion, all data is preserved, except that color loss may sometimes occur (URL 42).

The vtkVRMLExporter filter converts the polygonal mesh from VTK to VRML format. This conversion is lossfree. It is important because the VTK file format is not platform-independent. Unlike VRML where browsers run under many different operating systems can display VRML-based graphical representations, this gives the platform an independent way of displaying complex 3D visualizations.

7.4 Conversion between VRML/X3D and I3S

Conversion of these two formats in both directions is not supported.

7.5 Conversion between VRML/X3D and IFC

Conversion of VRML/X3D format to IFC format is loss-free.

Conversion from IFC to VRML/X3D file format leads to loss of some geometric and semantic data because VRML format is very poor in semantic support (Zhu et al., 2021). In this process colors and materials are preserved. Also, metadata are preserved and any related data that cannot be mapped directly to VRML/X3D can be exported as metadata set embedded into the X3D file for further processing. VRML is used to represent the IFC model in the 3D world of virtual reality. The advantage of VRML is that can allow users to interactively navigate 3D models and implements some actions on any objects in the 3D VRML model.

7.6 Conversion between VRML/X3D and DXF

Conversion from VRML/X3D to DXF format is loss-free.

The CAD system can export the graphical model to many different format and one of them is VRML. Seamless conversion is achieved by converting CAD graphics to VRML files when both VRML and DXF files are saved in text format based on the capture of VRML and DXF files, which creates a practical and effective tool and method for mutual conversion of engineering works (drawings) (Zheng et al., 2014).

7.7 Conversion between VRML/X3D and OBJ

OBJ files are generally more limited than VRML, because there is no object's hierarchy and they have no support for primitives as spheres. As a result, all spheres are converted to meshes and all meshes are listed in a flat array of "groups". OBJ files do not store material properties, but allow reference to the materials in a separate „Material Template Library“ (MTL) file. Although OBJ format is widely used, not all software can recognize .mtl files. Therefore, color and material information is likely to be lost. The created model will generally have similar attributes to those that existed in the original model (URL 43).

Conversion from OBJ to VRML file format is loss-free.

7.8 Conversion between CityGML and GML3

GML3 has no built-in concept for the representation of surface materials such as colors, shininess and textures. Unlike CityGML which has a class called TexturedSurface. For this reason there is a loss of texture data when converting from CityGML format to GML3. The reverse conversion is lossless.

7.9 Conversion between CityGML and VTK

Conversion of these two formats in both directions is not supported.

7.10 Conversion between CityGML and I3S

I3S allows only one view of geometry per object while CityGML allows a view of multiple geometries. Also, I3S does not have support for generating LOD supported by CityGML so it is possible to display only one level of detail (URL 40).

Conversion from I3S to CityGML is not supported.

7.11 Conversion between CityGML and IFC

Conversion from IFC to CityGML is problematic in terms of geometry conversion and semantic transfer due to inconsistencies in semantics and modelling methods. IFC uses only entities that do not exist in CityGML format which causes clear information loss when converting from the IFC model to the CityGML model. Most CityGML objects can be mapped from the IFC model by partial matching, which implies that additional information or complex processing is required in mapping. Also, IFC and CityGML have been developed for different purposes. Therefore they cannot coincide completely in semantics. (Stouffs et al., 2018).

In the conversion from CityGML to IFC there is a loss of geographical coordinates, i.e. the local coordinate system must be used because IFC does not support the geodetic coordinate system. Some attributes are transferred to an IFC file from CityGML. However, the attributes depend on the data set and the program should be edited accordingly for each different data set. The IFC data model is expected to provide enough opportunities to retain most of the attributes from CityGML in an immediate way, but this requires checking which attributes should be retained. Also, the conversion of these formats depends on the software in which it is implemented which can lead to different results in terms of geometry and semantics (Salheb et al., 2020).

7.12 Conversion between CityGML and DXF

CityGML format is converted to DXF because the DXF format is used to display the massing models. The disadvantage is that massing models do not have semantic information (Chen et al, 2017). Also, DXF format do not have support for geodetic coordinate reference system. There is no data loss in the reverse conversion.

7.13 Conversion between CityGML and OBJ

CityGML and OBJ are two very different formats. In the conversion of CityGML to OBJ, there is a loss of information. Information loss occurs due to limited support for attributes and semantics in OBJ format. The semantic problem can be partially solved by storing the same semantic surfaces in separate OBJ files. As OBJ has limited use of attributes, materials are used as a solution to preserve attributes. The quantitative attributes of an object are converted into the material of a certain color assigned to all faces of that object. As OBJ is not geospatial format, the geodetic data is generally not stored along with the data (Biljecki and Ohori, 2015).

Conversion of OBJ to CityGML file format is without loss of the data.

7.14 Conversion between GML3 and VTK

Conversion of these two formats in both directions is not supported.

7.15 Conversion between GML3 and I3S

Conversion of these two formats in both directions is not supported.

7.16 Conversion between GML3 and IFC

Conversion of IFC format to GML format is possible without data loss. In this process local coordinates are converted to real world coordinates and also contains spatial relationship information between objects (Liu et al., 2014). However, reverse conversion is not possible.

7.17 Conversion between GML3 and DXF

Conversion from DXF to GML format does not lead to data loss. Objects from the local coordinate system are translated into a global coordinate system. Also, it is possible to display spatial data in GML format. The purpose of this conversion lies in the application of the GML format, i.e. its direct applicability via the Internet, the wide range of use and the ease of creating XML documents (Križaić et al., 2017).

In the conversion from GML to DXF format there is a loss of attributes because DXF has no semantic support. Also, DXF has no support for CRS, so there is a loss.

7.18 Conversion between GML3 and OBJ

Conversion from GML to OBJ leads to loss of geographical and some semantic information because OBJ has no CRS support and its semantics are very weak. There is no data loss in the reverse conversion.

7.19 Conversion between VTK and I3S

The conversion of these two formats in either direction is not supported.

7.20 Conversion between VTK and IFC

In the conversion from IFC to VTK format there is no data loss (Kivell and Commend, 2016).

The reverse conversion is not possible.

7.21 Conversion between VTK and DXF

The conversion of these two formats in either direction is not supported.

7.22 Conversion between VTK and OBJ

VTK can export OBJ file formats using the function vtkOBJExporter. In this conversion, all data is preserved, except that color loss may sometimes occur (URL 42). Conversion from OBJ to VTK does not lead to data loss.

7.23 Conversion between I3S and IFC

Converting rich information stored in IFC to I3S will result in loss of attributes and metadata (URL 41). The reverse conversion is not supported

7.24 Conversion between I3S and DXF

The conversion of these two formats in either direction is not supported.

7.25 Conversion between I3S and OBJ

Conversion of these two formats in both directions is not supported.

7.26 Conversion between IFC and DXF

During the conversion from DXF format to IFC format semantic data is added to geometric data. There is no data loss in this process (Häfele and Benner, 2015). In the conversion from IFC to DXF format there is a loss of semantics because DXF has no support for it.

7.27 Conversion between IFC and OBJ

Conversion from IFC to OBJ file format leads to the loss of many geometric and all semantic data (Zhu et al., 2021). There are no data losses in the reverse conversion.

7.28 Conversion between DXF and OBJ

In the conversion from DXF format to OBJ format and vice versa there is no data loss. The OBJ format has some advantages over the DXF format. OBJ has larger software support than DXF, so it is a more versatile file type than DXF (URL 44).

8. CONCLUSION

The formats covered in this paper and their ability to display a triangular irregular network are important in the fields dealing with the terrain and its as accurate and faithful a representation as possible. In order for this to be possible, the format needs to meet as many criteria as possible that are relevant for displaying the terrain using the TIN.

Comparing the formats, it was concluded that none of this formats fully meets the required criteria, each of them has its advantages and disadvantages and is designed for a specific purpose. In the future, efforts should be made to add or improve format elements that do not meet required requirements.

As there is no universal standard, a format conversion is often required. This paper also deals with the theoretical conversion of the format and it was concluded that the format that least supports the conversion is I3S, followed by VTK. VRML, DXF and OBJ are formats that can be converted to almost all other formats and whose conversion does not cause or causes very small losses. But the problem is that they already have some drawbacks in themselves such as weak attribute and no CRS support. On the other hand IFC and CityGML can also be converted to almost all other formats, but there are losses in conversion because other format often do not support everything they do. Improving conversions or creating a singe format that will meet all the required criteria should be addressed in the future.

Although each of the formats is well developed for its specific purposes and the applications that use it, for their mutual integration, conversion, to become univeral and enabled to be used by as many application as possible, my opinion is that much more effort is needed.

REFERENCES

Al-Salami A.M. (2009): TIN support in an open spatial database, International Institute for Geo-information Science and Earth Observation, Netherlands

Bell G., Carey R., Marrin C., (1996): The Virtual Reality Modeling Language Specification

Bhargava N., Bhargava R., Tanwar P.S., Triangulated Irregular Network Model from Mass Points, International Journal of Advanced Computer Research

Biljecki F., Ohori K.A., (2015): Automatic semantic-preserving conversion between OBJ and CityGML, 3D geoinformation, Delft University of Technology, Netherlands

Borrmann A., Amann J., Chipman T., Hyvarinen J., Liebich T., Muhič S., Mol L., Plume J., Scarponcini P., (2017): IFC Infra Overall Architecture Project Documentation and Guidelines

Campos R., Quintana J., Garcia R., Schmitt T., Spoelstra G., Schaap D. (2020): 3D Simplification Methods and Large Scale Terrain Tiling

Chen K.W., Janssen P., Norford L., (2017): Automatic Generation of Semantic 3D City Models from Conceptual Massing Models, National University of Singapore, Singapore

Coltekin A., Haggren H., (2000): VRML as a Tool for Web-based, 3D, Photo-realistic GIS, Helsinki University of Technology, Finland

Gomez Lahoz J., Gonzalez Aquilera D. (2006): Teaching Cartography Through VRML: Including Breaklines and Vertical Walls in TINs to Render Archaeological Sites, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science, Tokyo, Japan

Goodchild, M. F., Kimerling, A. J., (2002): Discrete Global Grids: A Web-Book, University of California, California

Gröger G., Kolbe T., Nagel C., Häfele K. H., (2012): OGC City Geography Markup Language (CityGML) Encoding Standard

Gröger G., Plumer L., (2012): CityGML – Interoperable semantic 3D city models, ISPRS Journal of Photogrammetry and Remote Sensing, 12-33

Häfele K. H., Benner J., (2015): Advance Mapping Structures and Standards, European research on energy-efficient healthcare districts

Ismail A., Nahar A., Scherer R., (2017): Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC standard, TU Dresden, Germany

Kivell S., Commend S., (2016): Large 3D model to estimate the impact of a construction project on groundwater flow, GeoMod Ing. SA, Lausanne

Knapp M., (2002): Mesh Decimation Using VTK, Institute of Computer Graphics and Algorithms Vienna University of Technology, Vienna, Austria

Križaić V., Varga M., Gradišer L., Buč S., (2017): Legalization Applications Integration, Creative Construction Conference 2017, Primošten, Croatia

Krtalić A., Gajski D., Maltarski M., (2019.): Digitalni trodimenzionalni prikaz scene i satelitska stereofotogrametrija, Geodetski list, 147-164

Kumar K., Labetski A., Ledoux H., Stoter J. (2019): An Improved LOD Framework for the Terrains in 3D City Models, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Singapore

Kumar K., Ledoux H., Stoter J., (2016): A CITYGML EXTENSION FOR HANDLING VERY LARGE TINS, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Athens, Greece

Leung A., Coddington P. (1998): Interactive viewing of 3D terrain models using VRML, School of Computer and Information Science, Syracuse University, Syracuse NY 13244, U.S.A.

Liu H., Shi R., Zhu L., Jing C., (2014): Conversion of model file information from IFC to GML, Geoscience and Remote Sensing Symposium, Quebec City, Canada

McHenry K., Bahcsy P., (2008): An Overview of 3D Data Content, File Formats and Viewers, University of Illinois at Urbana-Champaign, Urbana, IL

Pajarola R. (2002): Overview of Quadtree-based Terrain Triangulation and Visualization, Department of Information & Computer Science University of California, Irvine

Portele C., (2012): OGC Geography Markup Language (GML) — Extended schemas and encoding rules

Reddy M., Leclerc Y.G., Iverson L., Bletter N, Vidimce K., (1999.): Modeling the Digital Earth in VRML, SRI International, Menlo Park, California

Reed C., Belayneh T., (2020): OGC Indexed 3d Scene Layer (I3S) and Scene Layer Package Format Specification

Salheb N., Ohori K., Stoter J., (2020): Automatic Conversion of CityGML to IFC, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, London, UK

Stouffs R., Tauscher H., Biljecki F., (2018): Achieving Complete and Near-Lossless Conversion from IFC to CityGML, Department of Architecture, School of Design and Environment, National University of Singapore, Singapore

Taubin G., Horn W., Lazarus F., Rossignac J., (1998): Geometry Coding and VRML

Tiwari V., (2015): 3D Visualization of City GML Building Data on World Wind, International Institute of Information Technology Hyderabad Gachibowli, Hyderabad, India

Tolmer C-E., Castaing C., Diab Y., Morand D., (2013): CityGML and IFC: going further than LOD, International Congress Digital Heritage, Marseille, France

Vivoni E. R., (2003): Hydrologic Modeling using Triangulated Irregular Networks: Terrain Representation, Flood Forecasting and Catchment Response, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, USA

Zheng C., Liu S., Zhang S., (2014): Three-dimensional CAD graphics and virtual reality file conversion, University of Chinese Academy of Sciences, Beijing

Zhu J., Wang P., Wang X., (2021): An Assessment of Paths for Transforming IFC to Shapefile for Integration of BIM and GIS, Ruhr Universitat Bochum, Bochum

URL LIST:

URL 1. What is VRML file?,
https://docs.fileformat.com/3d/vrml/, (13.05.2021.)

URL 2. Triangulated Irregular Network,
http://wiki.gis.com/wiki/index.php/Triangulated_irregular_network, (15.05.2021.)

URL 3. What is a TIN surface?,
https://desktop.arcgis.com/en/arcmap/latest/manage-data/tin/fundamentals-of-tin-surfaces.htm, (15.05.2021.)

URL 4. Delaunay Triangulation,
https://towardsdatascience.com/delaunay-triangulation-228a86d1ddad, (15.05.2021.)

URL 5. Delaunay Triangulation,
https://en.wikipedia.org/wiki/Delaunay_triangulation, (15.05.2021.)

URL 6. How to create a TIN from a GENERATE input file?,
https://www.caee.utexas.edu/prof/maidment/grad/azagra/Research/tin.htm, (16.05.2021.)

URL 7. 10 best file formats used in various industries,
https://professional3dservices.com/blog/3D-file-formats.html, (16.05.2021.)

URL 8. The most popular 3D file formats for 3D commerce,
https://www.marxentlabs.com/3d-file-formats/, (16.05.2021.)

URL 9. The Most Common 3D File Formats,
https://all3dp.com/3d-file-format-3d-files-3d-printer-3d-cad-vrml-stl-obj/#vrml, (16.05.2021.)

URL 10. The Virtual Reality Modeling Language,
http://paulbourke.net/dataformats/vrml1/, (18.05.2021.)

URL 11. What Is VRML? We Uncover How This 3D Programming Language Works,
https://websitebuilders.com/how-to/glossary/vrml/, (31.05.2021.)

URL 12. FreeWRL is an X3D/VRML open source viewer for Windows and Android,
http://freewrl.sourceforge.net/, (31.05.2021.)

URL 13. Level of detail,
https://de.wikipedia.org/wiki/Level_of_Detail, (01.06.2021.)

URL 14. What is The Difference Between 2d, 2.5d & 3d Animation?, https://www.explainervideoly.com/blog/what-is-the-difference-between-2d-2-5d-3d-animation/, (02.06.2021.)

URL 15. Content Standard for Digital Geospatial Metadata (CSDGM),
https://www.fgdc.gov/metadata/csdgm-standard, (04.06.2021.)

URL 16. X3D Metadata Information,
https://x3dgraphics.com/examples/X3dForWebAuthors/Chapter15-Metadata/Chapter15-MetadataInformation.html, (04.06.2021.)

URL 17. CityGML - City Geography Markup Language,
https://www.citygmlwiki.org/index.php?title=Citygml_Wiki, (04.06.2021.)

URL18. Compactly representing massive terrain models as TINs in CityGML,
https://onlinelibrary.wiley.com/doi/full/10.1111/tgis.12456, (08.06.2021.)

URL19. Commercial Software,
https://www.citygmlwiki.org/index.php/Commercial_Software, (08.06.2021.)

URL 20. A metadata ADE for CityGML,
https://opengeospatialdata.springeropen.com/articles/10.1186/s40965-018-0057-4, (11.06.2021.)

URL 21. VTK File Formats,
https://vtk.org/wp-content/uploads/2015/04/file-formats.pdf, (14.06.2021.)

URL 22. Coordinate,
https://kitware.github.io/vtk-js/api/Rendering_Core_Coordinate.html, (14.06.2021.)

URL 23. VTK,
https://www.kitware.com/platforms/, (14.06.2021.)

URL 24. vtkLODActor Class Reference,
https://vtk.org/doc/release/5.0/html/a01691.html, (15.06.2021.)

URL 25. WFS settings,
https://docs.geoserver.org/stable/en/user/services/wfs/webadmin.html, (16.06.2021.)

URL 26. What is IFC and what do you need to know about it?,
https://blog.areo.io/what-is-ifc/, (30.06.2021.)

URL 27. IFC format,
https://bim-hrvatska.hr/ifc-format/, (30.06.2021.)

URL 28. DXF (File Format): Simply Explained,
https://all3dp.com/2/dxf-file-format-simply-explained/, (01.07.2021.)

URL 29. DXF (AutoCAD Drawing Interchange Format) Family, ASCII variant,
https://www.loc.gov/preservation/digital/formats/fdd/fdd000446.shtml, (01.07.2021.)

URL 30. Integrate Wavefront OBJ Using FME,

https://www.safe.com/integrate/obj/, (01.07.2021.)

URL 31. What is X3D?

https://www.web3d.org/x3d/what-x3d, (06.07.2021.)

URL 32. OGC Indexed 3d Scene Layer (I3S) and Scene Layer Package Format Specification,

https://docs.opengeospatial.org/cs/17-014r5/17-014r5.html, (06.07.2021.)

URL 33. 3D data suite,

https://www.maxar.com/products/3d-data-suite, (14.07.2021.)

URL 34. Sustainability and Interoperability: An Economic Study on BIM Implementation by a Small Civil Engineering Firm,

https://www.mdpi.com/2071-1050/12/22/9581/htm, (14.07.2021.)

URL 35. Off-line programming of a robotic system based on DXF files of 3D models,

https://ieeexplore.ieee.org/abstract/document/6720494, (14.07.2021.)

URL 36. DXF,

https://www.bluemarblegeo.com/knowledgebase/global-mapper-20/Formats/DXF.htm, (14.07.2021.)

URL 37. Civil 3D – Create TIN surface,

https://www.youtube.com/watch?v=pkUwbz0QxWQ, (14.07.2021.)

URL 38. Chapter 9 – Advanced Algorithms,

https://kitware.github.io/vtk-examples/site/VTKBook/09Chapter9/#decimation, (16.07.2021.)

URL 39. OBJ2XPMesh (MeshRemexe alternative),

https://forums.x-plane.org/index.php?/forums/topic/232795-obj2xpmesh-meshremexe-alternative/, (16.07.2021.)

URL 40. CityGML to I3S Toolbox,

https://www.arcgis.com/home/item.html?id=585b25bad7b94c09aa7fb1833db1f318, (22.07.2021.)

URL 41. Common Patterns for BIM and GIS Integration,

https://www.esri.com/arcgis-blog/products/arcgis-pro/transportation/common-patterns-for-bim-and-gis-integration/, (22.07.2021.)

URL 42. Scene import and export in ParaView and VTK,

https://www.programmersought.com/article/54174068739/, (23.07.2021.)

URL 43. General Commands Manual,

https://bio3d.colorado.edu/imod/doc/man/imod2obj.html, (23.07.2021.)

URL 44. DXF to OBJ: How to Convert DXF files to OBJ, https://all3dp.com/2/dxf-to-obj-convert-files/, (24.07.2021.)

URL 45. Diving Deep: Data-Management and Visualization Strategies for Adaptive Mesh Refinement Simulations,

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=774839, (25.07.2021.)

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

2.5D two and a half dimensional

2D two-dimensional

3D three-dimensional

ADE Application Domain Extensions

ASCII American Standard Code for Information Interchange

CAD Computer-aided design

CRS Coordinate Reference System

DEM Digital Elevation Model

DSM Digital Surface Model

DTM Digital Terrain Model

DWG AutoCAD Drawing Format

DXF AutoCAD Exchange Format

GIS Geographic Information System

GML Geography Markup Language

GNSS Global Navigation Satellite System

I3S Indexed 3D Scene Layers

IFC Industry Foundation Classes

LiDAR Light Detection and Ranging

LOD Level of Detail

nDSM normalized Digital Surface Model

NURBS Non-Uniform Rational B-Splines

TIN Triangular Irregular Network

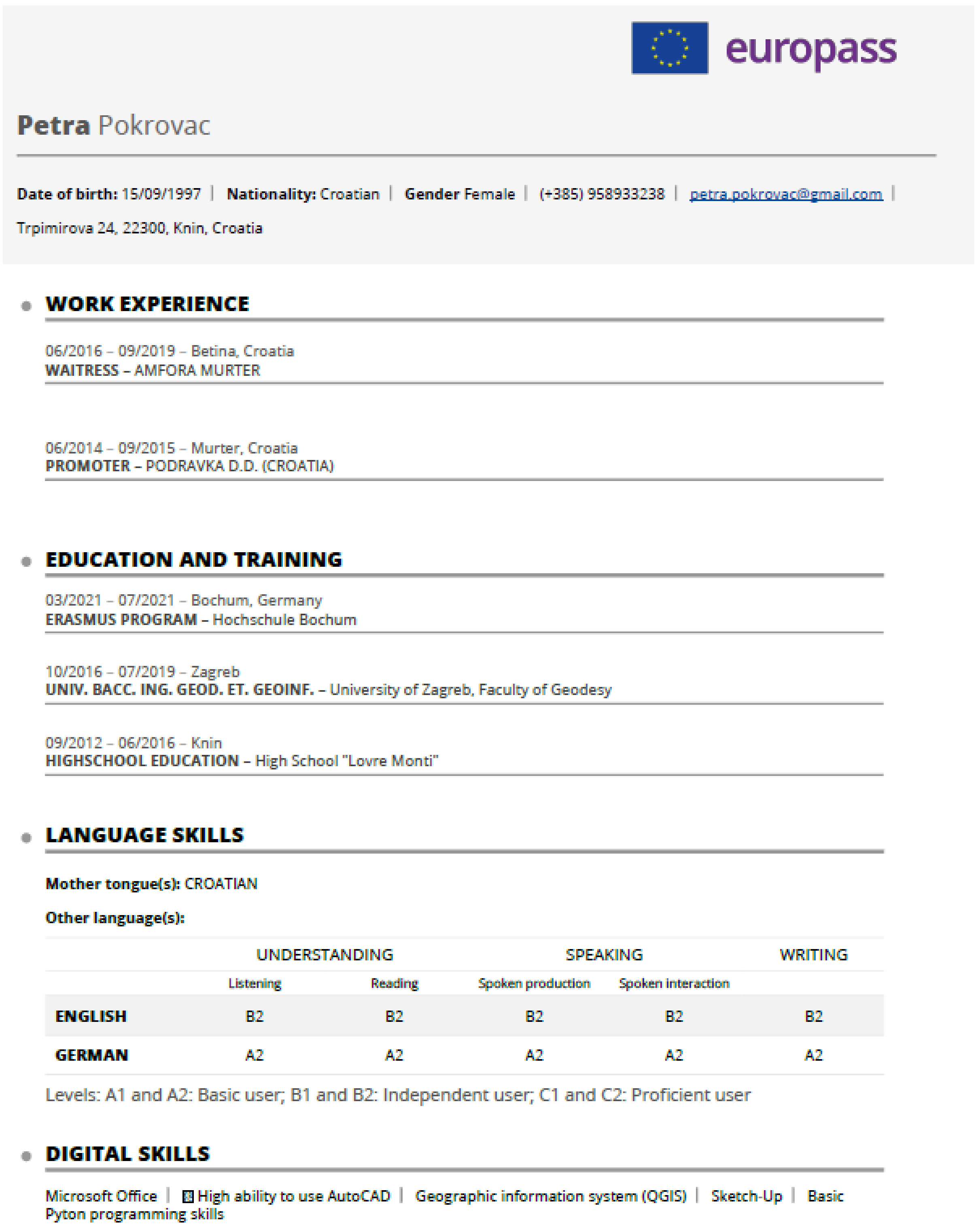
VRML Virtual Reality Modeling Language

VTK The Visualization Toolkit

X3D Extensible 3D Graphics

XML Extensible Markup Language

BIOGRAPHY



europass

Petra Pokrovac

Date of birth: 15/09/1997 | **Nationality:** Croatian | **Gender** Female | (+385) 958933238 | petra.pokrovac@gmail.com |

Trpimirova 24, 22300, Knin, Croatia

WORK EXPERIENCE

06/2016 – 09/2019 – Betina, Croatia
WAITRESS – AMFORA MURTER

06/2014 – 09/2015 – Murter, Croatia
PROMOTER – PODRAVKA D.D. (CROATIA)

EDUCATION AND TRAINING

03/2021 – 07/2021 – Bochum, Germany
ERASMUS PROGRAM – Hochschule Bochum

10/2016 – 07/2019 – Zagreb
UNIV. BACC. ING. GEOD. ET. GEOINF. – University of Zagreb, Faculty of Geodesy

09/2012 – 06/2016 – Knin
HIGHSCHOOL EDUCATION – High School "Lovre Monti"

LANGUAGE SKILLS

Mother tongue(s): CROATIAN

Other language(s):

	UNDERSTANDING		SPEAKING		WRITING
	Listening	Reading	Spoken production	Spoken interaction	
ENGLISH	B2	B2	B2	B2	B2
GERMAN	A2	A2	A2	A2	A2

Levels: A1 and A2: Basic user; B1 and B2: Independent user; C1 and C2: Proficient user

DIGITAL SKILLS

Microsoft Office | ⊠ High ability to use AutoCAD | Geographic information system (QGIS) | Sketch-Up | Basic Pyton programming skills