Open Geospatial Consortium, Inc.

Date: 2009-08-05

Reference number of this document: OGC 09-075r1

Version: **0.3.0**

Category: Public Engineering Report

Editor: Arne Schilling

OGC® OWS-6 3D Flythrough (W3DS) Engineering Report

Warning

Document type:	OpenGIS® Public Engineering Report
Document subtype:	NA
Document stage:	Approved for Public Release
Document language:	English

Preface

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

Forward

OWS-6 Testbed

OWS testbeds are part of OGC's Interoperability Program, a global, hands-on and collaborative prototyping program designed to rapidly develop, test and deliver Engineering Reports and Chnage Requests into the OGC Specification Program, where they are formalized for public release. In OGC's Interoperability Initiatives, international teams of technology providers work together to solve specific geoprocessing interoperability problems posed by the Initiative's sponsoring organizations. OGC Interoperability Initiatives include test beds, pilot projects, interoperability experiments and interoperability support services - all designed to encourage rapid development, testing, validation and adoption of OGC standards.

In April 2008, the OGC issued a call for sponsors for an OGC Web Services, Phase 6 (OWS-6) Testbed activity. The activity completed in June 2009. There is a series of on-line demonstrations available here: http://www.opengeospatial.org/pub/www/ows6/index.html
The OWS-6 sponsors are organizations seeking open standards for their interoperability requirements. After analyzing their requirements, the OGC Interoperability Team recommended to the sponsors that the content of the OWS-6 initiative be organized around the following threads:

1. Sensor Web Enablement (SWE)
2. Geo Processing Workflow (GPW)
3. Aeronautical Information Management (AIM)
4. Decision Support Services (DSS)
5. Compliance Testing (CITE)

The OWS-6 sponsoring organizations were:

- U.S. National Geospatial-Intelligence Agency (NGA)
- Joint Program Executive Office for Chemical and Biological Defense (JPEO-CBD)
- GeoConnections - Natural Resources Canada
- U.S. Federal Aviation Agency (FAA)
- EUROCONTROL
- EADS Defence and Communications Systems
- US Geological Survey
- Lockheed Martin

- BAE Systems
- ERDAS, Inc.

The OWS-6 participating organizations were:
52North, AM Consult, Carbon Project, Charles Roswell, Compusult, con terra, CubeWerx, ESRI, FedEx, Galdos, Geomatys, GIS.FCU, Taiwan, GMU CSISS, Hitachi Ltd., Hitachi Advanced Systems Corp, Hitachi Software Engineering Co., Ltd., iGSI, GmbH, interactive instruments, lat/lon, GmbH, LISAsoft, Luciad, Lufthansa, NOAA MDL, Northrop Grumman TASC, OSS Nokalva, PCAvionics, Snowflake, Spot Image/ESA/Spacebel, STFC, UK, UAB CREAF, Univ Bonn Karto, Univ Bonn IGG, Univ Bunderswehr, Univ Muenster IfGI, Vightel, Yumetech.

Contents

Page

Figures

Page

OGC® OWS-6 3D Flythrough (W3DS) Engineering Report

1 Introduction

1.1 Scope

This document describes the 3D portrayal server components which were used in the OGC OWS-6 Decision Support Systems (DSS) thread. The objective pf this activity was to efficiently stream and display GML 3 content in internet or wireless networks with limited bandwidth, especially focusing on the CityGML application profile. The server for delivering landscape and city models is implemented as Web 3D Service (W3DS) that is designed as portrayal service. The W3DS is currently an OGC discussion paper. The interface is described in detail in this document. The concept of how to process CityGML content for efficient streaming and rendering is explained. CityGML is converted into VRML and shape files containing all available attributes and semantics. The W3DS is backed up by a data base containing VRML code for each individual feature and attribute tables.

This OGC® document is applicable to the OWS-6 DSS and GPW thread in order to retrieve the data sets used in the airport scenario, display the content interactively and get additional attribute information of features.

1.2 Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Arne Schilling	University of Bonn

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
2008-12-12	0.1	Arne Schilling		
2009-05-18	1.0	Arne Schilling		

2009-08-03	0.3.0	Carl Reed	Various	Prepare for publication as PER.

1.4 Future work

This is the first draft. The final document will be submitted until 04-17-2009.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

OGC 06-121r3, *OpenGIS® Web Services Common Specification*

NOTE This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

OGC 08-007r1 *OpenGIS® City Geography Markup Language (CityGML) Encoding Standard*

OGC 05-019 *OpenGIS® Web 3D Service Discussion Paper*, *Version 0.3.0*

In addition to this document, this report includes several XML Schema Document files as specified in Annex A.

3 Conventions

3.1 Abbreviated terms

API	Application Program Interface
BIM	Building Information Model
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf
CRS	Coordinate Reference System
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DSS	Decision Support System
GPW	Geo Processing Workflow

IDL	Interface Definition Language
LOD	Level of Detail

3.2 UML notation

Most diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121r3].

4 Overview

The Web3D Service (W3DS, OGC 05-019) was proposed as Portrayal Service for 3D geo data. It has currently the status of a discussion paper. The latest version is 0.3.0. The W3DS creates 3D scenes of landscape and city models that can be explored interactively on the client. It delivers graphical elements for displaying a complete 3D map or parts of it. The client, which is equipped with modern 3D graphics acceleration hardware can decide how to visualize and explore the scene and is not confined to certain viewpoints (like e.g. in panoramic images). The W3DS is suitable for a Medium Server Medium Client concept, which means that the Server collects the necessary geo data, and generates display elements which are streamed to the Client. The Client is responsible for rendering the display elements on the screen using the rendering techniques of his choice.

The W3DS is proposed in the DSS thread as a middle tier between the actual data store containing city and landscape models and the client application, e.g. the proposed Virtual Flythrough Application or Integrated Client which is used as a front end and allows user interaction and rendering at an interactive frame rate. Beyond the DSS thread, W3DS servers have been successfully deployed in numerous research activities, including 3D routing in emergency cases [2], user defined styling for generating thematic cartographic representations [4], providing large user generated content (OpenStreetMap 3D, [3]). Basic principles and components for setting up 3D SDIs are explained in [5] and [7].

5 Web 3D Service

5.1 Introduction

The W3DS is designed as Portrayal Service. It does not provide the raw geo data but a 3D representation of the data. The difference is that the geo data itself is organized in features and object with additional attributes, metadata, and semantics, and the result of a Portrayal Service is just something that can be viewed. There is no guarantee on the internal structure of the resulting scenes and attribute data is generally missing due to lacking support in current 3D internet formats (e.g. COLLADA, X3D). It is even advisable to re-organize the scene graph structure for a more efficient rendering. For retrieving fully GML compatible and attribute rich geo data, an OGC WFS should be used. The advantage of using visualization-centric formats is that they support a wide range of features for controlling the visual appearance (e.g. textures, surface properties, animations, lighting, atmosphere) and that they can be more efficiently transmitted and encoded.

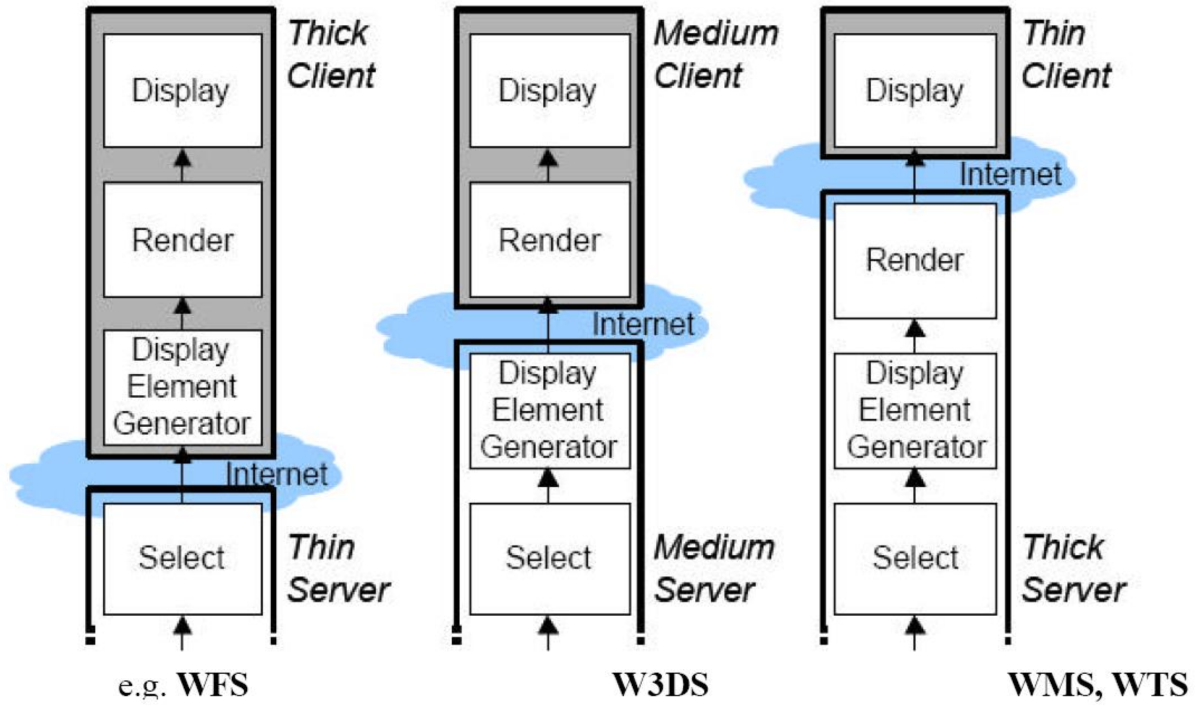


Fig. 1: Portrayal Pipeline Comparison

The W3DS component is usually backed by a WFS provided by a third party which provides content in OGC GML 3.x or OGC CityGML (OGC 08-007r1) format. Since the OGC WFS is maintained by a public authority such as a local municipality, a regional command center for disaster management, or a federal agency, it is always ensured that the data is kept up to date. In the OWS-6 testbed we import CityGML data sets which must be validated using the GML Application Schema for the UTDS data and converted into X3D format plus according attribute tables which can then imported into our data base.

The W3DS has its own data store for the contents that need to be visualized in DSS. The advantage of using a separate database for storing relevant data is that it can be configured and optimized for visualization exploiting the features of state of the art computer graphics. This makes it possible for using CityGML contents for very efficient fly-through visualizations. Otherwise the file structure of GML3 makes it very difficult for graphics hardware to render the contents efficiently and only a small portion of the data can be displayed. The optimizations are part of the conversion process and include mesh reduction for Digital Elevation Models, restructuring the scene graph, combining objects with the same attributes, efficient usage of display lists, generalization, adapting texture resolutions, compression, and other techniques.

5.2 W3DS Scope

As mentioned before, the W3DS is a Portrayal Service and delivers Scenes that are comparable to images. Similar to an image, a Scene is a representation of the geo data and is composed of display elements, which may comprise triangles, polygons, billboards, text, textures, materials, atmospheric elements (fog, lights), also animations are possible. A Scene is encoded in formats that are widely accepted for streaming 3D contents over the internet. Our implementation delivers all content in VRML.

What is a Scene?

1. A Scene is composed of data from one or multiple layers. Different integration strategies are feasible. Simple client implementations may request static and ready to use Scenes. The GetScene request would include all layers that need to displayed at one time (e.g. terrain, buildings, trees, objects). Since combining and overlaying multiple Scenes is technically no problem, GetScene requests can also be sent to different servers and the results integrated in the client. E.g. a 3D map application might download a landscape model from a federal W3DS provider and a city model from the local authorities.

2. A Scene may also contain map elements (title, compass, scale bar, legend etc.). In this case the Scene becomes a real 3D map ready for online publication. Predefined viewpoints can be provided simplifying the navigation in the Scene.

3. A Scene must be provided in a CRS that can be used for visualization. At least it must be kept in mind that geographic coordinates (WGS84) are not suitable for the display. Large coordinates are problematic because of single precision arithmetic used in graphics hardware. Therefore an offset value is usually used shifting the Scene back to the local origin of the coordinate system. The y axis must point upwards which is the standard in most graphics formats. X and z axes define East and South.

4. A Scene is composed of “Display Elements” (geometries, triangles, materials, animations, lights, fog).

5. The structure of a Scene is not defined. CityGML contents can be reconfigured in order to reduce the complexity. Examples are building parts such as multiple WallSurfaces, RoofSurfaces, which can be combined if they share the same material. Also all gml:Polygons of a gml:MultiSurface can be combined into one geometry.

6. Semantics are usually missing since internet formats usually do not support it. Semantics and attribute information must be provided by additional service requests (GetFeatureInfo). However, this depends on the format being used. VRML does not support semantics.

5.3 Interface Specification

The Service interface supports 4 operations:

1. GetCapabilities
2. GetSCene
3. GetFeatureInfo
4. GetLayerInfo

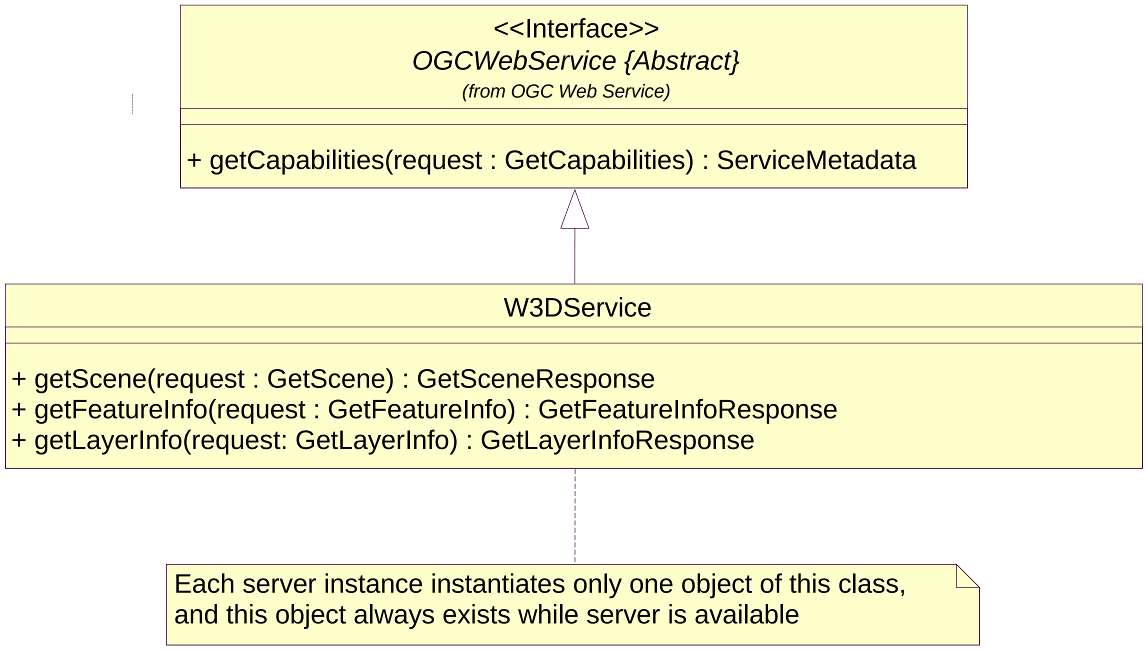

<<Interface>>
OGCWebService {Abstract}
(from OGC Web Service)
+ getCapabilities(request : GetCapabilities) : ServiceMetadata
W3DService
+ getScene(request : GetScene) : GetSceneResponse
+ getFeatureInfo(request : GetFeatureInfo) : GetFeatureInfoResponse
+ getLayerInfo(request: GetLayerInfo) : GetLayerInfoResponse
Each server instance instantiates only one object of this class, and this object always exists while server is available

Fig. 2: W3DS interface UML diagram

5.3.1 GetCapabilites Operation

The W3DS GetCapabilities operation is almost identical to the WMS GetCapabilities operation (OGC 06-042). One additional layer element was defined for providing information on the available LODs.

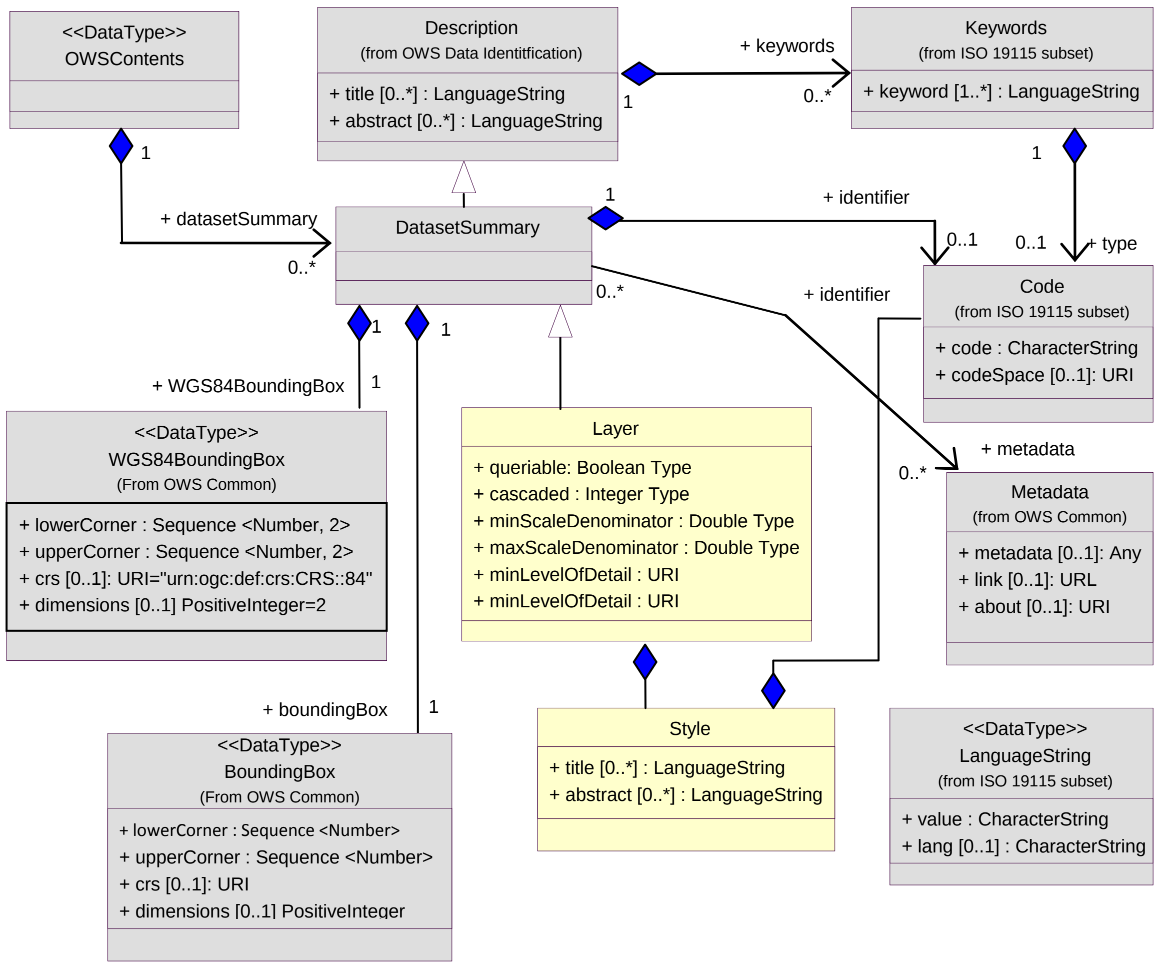

<<DataType>>
OWSContents
Description
(from OWS Data Identitfication)
+ title [0..*] : LanguageString
+ abstract [0..*] : LanguageString
+ keywords
Keywords
(from ISO 19115 subset)
+ keyword [1..*] : LanguageString
+ datasetSummary
DatasetSummary
+ identifier
Code
(from ISO 19115 subset)
+ code : CharacterString
+ codeSpace [0..1]: URI
+ type
+ WGS84BoundingBox
<<DataType>>
WGS84BoundingBox
(From OWS Common)
+ lowerCorner : Sequence <Number, 2>
+ upperCorner : Sequence <Number, 2>
+ crs [0..1]: URI="urn:ogc:def:crs:CRS::84"
+ dimensions [0..1] PositiveInteger=2
Layer
+ queriable: Boolean Type
+ cascaded : Integer Type
+ minScaleDenominator : Double Type
+ maxScaleDenominator : Double Type
+ minLevelOfDetail : URI
+ minLevelOfDetail : URI
+ metadata
Metadata
(from OWS Common)
+ metadata [0..1]: Any
+ link [0..1]: URL
+ about [0..1]: URI
+ boundingBox
<<DataType>>
BoundingBox
(From OWS Common)
+ lowerCorner : Sequence <Number>
+ upperCorner : Sequence <Number>
+ crs [0..1]: URI
+ dimensions [0..1] PositiveInteger
Style
+ title [0..*] : LanguageString
+ abstract [0..*] : LanguageString
<<DataType>>
LanguageString
(from ISO 19115 subset)
+ value : CharacterString
+ lang [0..1] : CharacterString
1
0..*
0..1

Fig. 3: UML model of W3DS contents section

5.3.1.1 Levels of Detail

The optional <MinLevelOfDetail> and <MaxLevelOfDetail> elements describe the range of Levels of Detail that can be provided by the layer. The actual LOD value which is a number, is preceded by a qualifier. The qualifier specifies a namespace or value range which is used for correctly interpreting the value. The most commonly used LOD definition originates from the CityGML standard and ranges from 0 (landscape model to 4 (indoor model).

Example:

```
<MinLevelOfDetail>CityGML:1</MinLevelOfDetail>
<MaxLevelOfDetail>CityGML:4</MaxLevelOfDetail>
```

5.3.2 GetScene Operation

The GetScene request is the main operation of a W3DS. The basic usage is described in OGC's OWS standard and is corresponding to the GetMap request of the ISO/DIS 19128 Web Map Service standard. Table 1 shows the parameters for a GetScene request.

Parameters marked with "R" are mandatory, "O" means they are optional and "C" is conditional, i.e. the usage of the conditional rated parameters depends on the required or optional parameters.

Table 1 — Parameters of the GetScene request

URL parameter	Required/ Optional/ Conditional	annotation
VERSION=*<version>*	R	requested version
REQUEST=GetScene	R	requested operation
CRS=namespace:identifier	R	coordindate reference system
POI=*<point_of_interest>*	C	x,y,z point coordinates according to CRS
PITCH=*<pitch>*	C	angle of inclination [degree]
YAW=*<yaw>*	C	azimuth [degrees]
ROLL=*<roll>*	O	rotation around viewing vector [degree]
DISTANCE=*<distance>*	C	distance POI to POC [meter]
POC=x, y, z	C	x,y,z coordinates of camera according to SRS
AOV=*<angle_of_view>*	C	angle of view [degree]
BBOX=*xmin,ymin,xmax,ymax*	R	2d bounding box
MINHEIGHT=*<lower_limit>*	O	displaying objects with height ≥ *lower_limit* according to SRS
MAXHEIGHT=*<upper_limit>*	O	displaying objects with height ≤ *upper_limit* according to SRS
LAYERS=*<layer list>*	O	comma separated list of 3D object sets
STYLES=*<style list>*	O	comma separated list of styles for each layer
FORMAT=*<format>*	R	MIME type of output
TIME=*<date_and_time>*	O	date and time
EXCEPTIONS=*<excepttype>*	O	exception format
TRANSLATE=*x,y,z*	C	translation vector that is applied to all 3D coordinates
ENVIRONMENT=on / off	O	switch on/off background elements like sky or light source
BGCOLOR=*<color>*	O	background color
BGIMAGE=*<image url>*	O	URL of background image

LOD=<*qualifier:number*>	O	Level of Detail
LOD_SELECTION=<*string*>	O	The method how to select the LODs. *("equals" | "equals_or_smaller")*
SELECTION_METHOD=<string>	O	How to select features *("intersection" | "by_center" | "crop").*
SLD=<*string*>	O	URL reference to SLD document
SLD_BODY=<*string*>	O	inline SLD Document in GET request
StyledLayerDescriptor=<xml>	O	inline SLD Document in POST request

The parameter are described in detail in the W3DS discussion paper which can be downloaded from the OGC portal. Reference number: OGC 05-019.

Example:

```
http://myw3ds.de/W3DS_HD/W3DS?REQUEST=GetScene&VERSION=0.1.0&SRS=EPSG:3
1466&FORMAT=model/vrml&layers=Terrain,Buildings&bbox=3479000.0,5474500.
0,3480000.0,5476000.0&POI=3478633.0,5475125.0,109.0&POC=3479430.0,54749
00.0,250.0
```

5.3.3 GetFeatureInfo Operation

The GetFeatureInfo operation is designed to provide clients of a W3DS with more information about features within a scene that is currently displayed. The canonical use case for GetFeatureInfo is that a user explores the response of a GetScene request and points at an object within the scene for which to obtain more information. The concept of this operation is that the client determines a location in 3D space by clicking on an object and calculates either the intersection point of the object geometry with the picking ray or the center point of the object and submits this location together with additional parameters to the server. The location can be also determined by other 3D input devices or by any other means. Since the W3DS protocol is stateless, also the current CRS needs to be submitted so that the W3DS is able to reconstruct the location within the CRS of its data store. Also the layer(s) must be submitted in order to restrict the search to selected data sets. The current implementation selects features based on the footprint. For each feature a 2D footprint was generated and stored in the database. The coordinates in the GetFeatureInfo request are transformed into a small bounding box (size 0.5 m) which is used for a spatial database query. Features are selected if the bounding box intersects with the footprint. The response is encoded as MIME type text/html and can be easily displayed in a web browser.

Table 2 — Parameters of the GetFeatureInfo request

URL parameter	**Required/ Optional/ Conditional**	**annotation**

VERSION=0.3.1	M	Request version.
REQUEST=GetFeatureInfo	M	Request name.
CRS	M	Value of CRS in 19128 is a text string that identifies a coordinate reference system defined by another authority.
QUERY_LAYERS=layer_list	M	Comma-separated list of one or more layers to be queried.
INFO_FORMAT=output_format	M	Return format of feature information (MIME type).
FEATURE_COUNT=number	O	Number of features about which to return information (default=1).
X=number	M	x coordinate of the location in Scene CS.
Y=number	M	y coordinate of the location in Scene CS.
Z=number	M	z coordinate of the location in Scene CS.
EXCEPTIONS=exception_format	O	The format in which exceptions are to be reported by the W3DS (default= XML).

5.3.3.1 VERSION

The mandatory VERSION parameter is defined in 6.2.1. The value “0.3.1” shall be used for GetFeatureInfo requests that comply with this proposed Standard.

5.3.3.2 REQUEST

The mandatory REQUEST parameter is defined in ?.?.?. For GetFeatureInfo, the value “GetFeatureInfo” shall be used.

5.3.3.3 CRS

The mandatory CRS parameter is defined in 7.3.2.4. The value should be the same as used in the GetScene request, regardless of the internally used coordinate system for the computer graphics, which may be different.

5.3.3.4 QUERY_LAYERS

The mandatory QUERY_LAYERS parameter lists the server layer(s) from which features and their information should be retrieved. The value is a comma-separated list of one or more layers. This parameter shall contain at least one layer name. If any layer in the QUERY_LAYERS parameter is not defined in the service metadata of the W3DS, the server shall issue a service exception (code = LayerNotDefined).

5.3.3.5 INFO_FORMAT

The mandatory INFO_FORMAT parameter indicates what format to use when returning the feature information. Supported values for a GetFeatureInfo request on a W3DS server are listed as MIME types in one or more <Request><FeatureInfo><Format> elements of its service metadata. The entire MIME type string in <Format> is used as the value of the INFO_FORMAT parameter. In an HTTP environment, the MIME type shall be set on the

returned object using the Content-type entity header. If the request specifies a format not supported by the server, the server shall issue a service exception (code = InvalidFormat).

EXAMPLE The parameter INFO_FORMAT=text/xml requests that the feature information be formatted in XML.

5.3.3.6 FEATURE_COUNT

The optional FEATURE_COUNT parameter states the maximum number of features per layer for which feature information shall be returned. Its value is a positive integer. The default value is 1 if this parameter is omitted or is other than a positive integer.

5.3.3.7 X, Y, Z

The mandatory X, Y, and Z request parameters are floating point values that indicate a location in 3D space within the scene from which feature information has to be generated. The location should be within or at the border of a feature geometry, but it does not have to be. The W3DS shall detect the feature(s) which geometry is containing the location or lying nearest to it. Note that x, y, z values are not in computer graphics coordinate system, but they should have the same axis orientation and direction as defined in the CRS parameter.

Example:

```
http://myserver.de/W3DS_HD/W3DS?REQUEST=GetFeatureInfo&VERSION=0.1.0&QU
ERY_Layers=Gebaeude_LOD1&X=3478297.22&Y=5475044.53&SRS=EPSG:31467&Info_
Format=text/html&Feature_Count=10
```

5.3.4 GetLayerInfo Operation

The purpose of the GetLayerInfo request is to collect information on the available attribute names and the values in the attribute table of a specific layer. The attribute table is managed by the W3DS in a database table. The entries in the attribute table are linked to the geometries that can be retrieved using the GetScene request.

The GetLayerInfo request contains a mandatory LAYER parameter for identifying the dataset from which attribute information should be received and an optional COLUMNNAME parameter. If only the LAYER parameter is used, then the response of the request contains only a list of all available attribute or column names in the format specified by the FOMRAT parameter. The received attribute names can then be used to receive additional information on the available values in the attribute table. If additionally to the LAYER parameter also a COLUMNNAME parameter is present, then the attribute table is queried for all available values that the features in the layer may have. The response to such a request contains a full list of unique values. The list has no duplicate values.

Table 3 — Parameters of the GetLayerInfo request

URL parameter	Required/ Optional/ Conditional	annotation
VERSION=0.3.1	M	Request version.
REQUEST=GetLayerInfo	M	Request name.
LAYER=<*layer*>	M	One Layer
COLUMNNAME=<column list>	O	Comma-separated list of one or more column to be queried.
FORMAT=output_format	M	Return format of feature information (MIME type).

5.3.4.1 VERSION

The mandatory VERSION parameter is defined in 6.2.1. The value “0.3.1” shall be used for GetFeatureInfo requests that comply with this proposed Standard.

5.3.4.2 REQUEST

For GetLayerInfo, the value “GetLayerInfo” must be used.

5.3.4.3 LAYER

The mandatory LAYER parameter specifies the server layer from which information should be retrieved. The value should be exactly one layer. If information from several layers needs to be collected, then several requests must be sent to the W3DS. If the layer in the LAYER parameter is not defined in the service metadata of the W3DS, the server shall issue a service exception (code = LayerNotDefined).

5.3.4.4 COLUMNNAME

The optional COLUMNNAME parameter specifies one or several table column(s) or attribute name(s) of the selected layer in the LAYER parameter from which all available unique values should be retrieved. The value is a comma-separated list of one or more attribute names. If all available Attributes of a Layer should be queried, the value of the COLUMNNAME parameter can be set to “ALLINFO”. If any layer in the COLUMNNAME parameter is not defined in the service metadata of the W3DS, the server shall issue a service exception (code = ColumnNameNotDefined).

5.3.4.5 FORMAT

The optional FORMAT parameter indicates what format to use when returning the attribute information. Supported values for a GetLayerInfo request on a W3DS server are listed as MIME types in one or more <Request><LayerInfo><Format> elements of its service metadata. The entire MIME type string in <Format> is used as the value of the FORMAT parameter. In an HTTP environment, the MIME type shall be set on the returned object using the Content-type entity header. If the request specifies a format not supported by the server, the server shall issue a service exception (code = InvalidFormat).

Examples:

GetLayerInfo request:

```
http://www.myserver.de/W3DS?REQUEST=GetLayerInfo&VERSION=0.3
.1&LAYER=Terrain&FORMAT=text/xml
```

GetLayerInfo response:

```
<GetLayerInfo>
      <Layer>
            <Name>Terrain</Name>
            <Attribute>
                  <Name>id</Name>
            </Attribute>
            <Attribute>
                  <Name>landuse</Name>
            </Attribute>
      </Layer>
</GetLayerInfo>
```

GetLayerInfo request:

```
http://www.myserver.de/W3DS?REQUEST=GetLayerInfo&VERSION=0.3
.1&LAYER=Terrain &COLUMNNAME=landuse&FORMAT=text/xml
```

GetLayerInfo response:

```
<GetLayerInfo>
    <Layer>
      <Name>Terrain</Name>
      <Attribute>
        <Name>landuse</Name>
        <Values>
            <Value>Bahn</Value>
            <Value>Baubloecke</Value>
            <Value>Gruenflaechen</Value>
            <Value>null</Value>
            <Value>Strassen</Value>
            <Value>Waldflaechen</Value>
            <Value>Wasserflaechen</Value>
         </Values>
      </Attribute>
    </Layer>
</GetLayerInfo>
```

6 CityGML Adaptor

In DSS the geographic content is provided and stored as GML data sets. This will be the basis for the data exchange between servers. CityGML is the accepted standard for describing 3D data sets for city environments since it enables to store all relevant information in a well defined XML structure which allows extracting specific parts such as geometry, specific properties very easily and in a standardized manner. The standard is implemented as GML application profile and comes with a XSD validation schema which makes it possible to validate any CityGML content before being processed. The XML schema defines elements for all possible objects and properties that are required in order to describe a city model. It is therefore inherently semantic rich. CityGML will be also the basis for setting up the previously described visualization server.

However, the scope of CityGML and W3DS differ. On the one hand, CityGML must be able to store all available information on city object including buildings, vegetation, streets, addresses etc. Information must not be lost when using CityGML as an exchange format automatically generated by converter tools or exporters built in into COTS software. It supports features derived from 3D modeling software (appearances, materials, textures) as well as from GIS software (bboxes, addresses), and from BIM software (semantics for buildings parts). On the other hand, W3DS is designed for supporting server-client architectures with limited bandwidth and for supporting very efficient real-time 3D rendering on the client side (e.g. virtual globes, virtual flythrough). The content maintained and served by the W3Ds must be compact and allow for efficient rendering at a high frame rate, yet it must be possible to retrieve additional information on selected objects.

For these reasons we implemented a CityGML adaptor, which is used when setting up the server and importing CityGML content into the W3DS database. The W3DS database stores tables for each data set containing all available information. Each row in this table represents a GIS feature which is a more traditional way to maintain geographic data sets, compared with the possibly hierarchical nature of XML files. Each row/feature contains:

- 2D footprint for 2D map representations
- Center point for selecting features
- Unique ID
- 3D geometry stored as VRML
- Attributes

The CityGML adaptor separates a GML file into an array of features using a predefined XML tag. In the example below (Listing 1) the `<groupMember>` tag has been used in order to identify the individual features. From the sub-elements in `<groupMember>` the geometry is extracted, in this case a `<lod1MultiSurface>`. How to deal with multi LOD features is left aside in this report. The `<lod1MultiSurface>` is represented as `<gml:MultiSurface>`, containing a set of `<gml:surfaceMember>`, `<gml:Polygon>`, `<gml:exterior>`, `<gml:LinearRing>`, and finally `<gml:posList>`. This leads to a highly redundant representation because each polygon has its own point list and does not share vertices with adjacent polygons. The adaptor merges all polygons and creates a single `VRML IndexedFaceSet` from a `<gml:MultiSurface>` containing a single `Coordinate` array and a `coordIndex` array defining triangles and polygons. The same applies to texture coordinates.

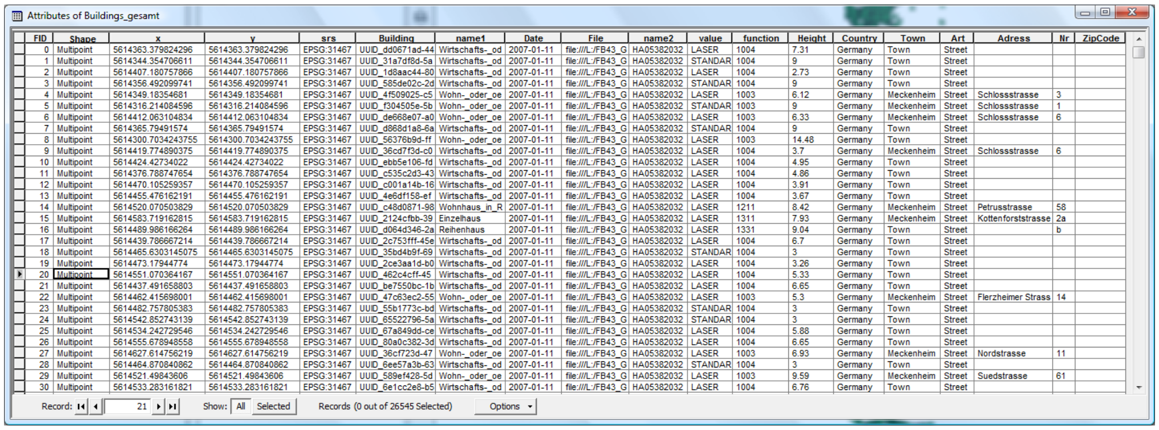
The adaptor converts the remaining XML elements in `<groupMember>` into attributes. It collects attribute information from the tag names (`<Building>, <CityFurniture>, <VegetationObject>, <WaterObject>, <Landuse>,…)` providing the top level semantics, from `gml:id` fields, which usually contain the cadastre unique IDs, from the `<address>` elements, and from additional elements which as defined in the GML application profile, e.g. `<stringAttribute>`.

```
<groupMember>

        <Building gml:id="HA05513000061300056    003">

          <gml:description>HA05513000061300056    003</gml:description>

          <gml:name>HA05513000061300056    003</gml:name>

          <creationDate>2008-10-30</creationDate>

          <gml:boundedBy>

            <gml:Envelope>

              <gml:lowerCorner>2572874.82700002 5715926.04600018
0</gml:lowerCorner>

              <gml:upperCorner>2572883.55300015 5715930.93
4.25</gml:upperCorner>

            </gml:Envelope>

          </gml:boundedBy>

          <lastModificationDate>2008-12-01</lastModificationDate>

          <updatingPerson>01.12.2008</updatingPerson>

          <reasonForUpdate>new calculations</reasonForUpdate>

          <lineage>City of Nimmerlein</lineage>

          <stringAttribute name="OSCHL">

            <value>2366</value>

          </stringAttribute>
```

```
<stringAttribute name="OART">
  <value>1032</value>
</stringAttribute>
<stringAttribute name="OBJNUM">
  <value>D00XBJH</value>
</stringAttribute>
<stringAttribute name="OBJNAME">
  <value>HA05513000061300056    003</value>
</stringAttribute>
<stringAttribute name="STREET">
  <value>06130</value>
</stringAttribute>
<stringAttribute name="HOUSENR">
  <value>56</value>
</stringAttribute>
<stringAttribute name="LFDNR">
  <value>3</value>
</stringAttribute>
<stringAttribute name="ANZHOCH">
  <value>1</value>
</stringAttribute>
<stringAttribute name="ANZDACH">
  <value>1</value>
</stringAttribute>
<doubleAttribute name="Traufe">
  <value>4.25</value>
</doubleAttribute>
<stringAttribute name="Origin">
  <value>LIDAR capture</value>
</stringAttribute>
<stringAttribute name="Cadastre_ID">
  <value>05513000061300056    003</value>
</stringAttribute>
```

```
          <function>1032</function>

          <yearOfConstruction>-1</yearOfConstruction>

          <measuredHeight uom="#m">4.25</measuredHeight>

          <storeysAboveGround>1</storeysAboveGround>

          <storeyHeightsAboveGround uom="#m">0 </storeyHeightsAboveGround>

          <storeyHeightsBelowGround uom="#m">0 </storeyHeightsBelowGround>

          <lod1MultiSurface>

            <gml:MultiSurface gml:id="5114">

              ...

            </gml:MultiSurface>

          </lod1MultiSurface>

          <address>

            <Address>

              <streetName>Backerodstr.</streetName>

              <houseNumber>0056</houseNumber>

              <zipCode>000</zipCode>

              <city>13</city>

            </Address>

          </address>

        </Building>

      </groupMember>
```

Listing 1: Extract from a sample CityGML file.


Attributes of Buildings_gesamt

FID	Shape	x	y	srs	Building	name1	Date	File	name2	value	function	Height	Country	Town	Art	Adress	Nr	ZipCode
0	Multipoint	5614363.379824296	5614363.379824296	EPSG:31467	UUID_dd0671ad-44	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	7.31	Germany	Town	Street			
1	Multipoint	5614344.354706611	5614344.354706611	EPSG:31467	UUID_31a7df8d-5a	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	STANDAR	1004	9	Germany	Town	Street			
2	Multipoint	5614407.180757866	5614407.180757866	EPSG:31467	UUID_1d8aac44-80	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	2.73	Germany	Town	Street			
3	Multipoint	5614356.492099741	5614356.492099741	EPSG:31467	UUID_585de02c-2d	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	STANDAR	1004	9	Germany	Town	Street			
4	Multipoint	5614349.18354681	5614349.18354681	EPSG:31467	UUID_4f509025-c5	Wohn-_oder_oe	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1003	6.12	Germany	Meckenheim	Street	Schlossstrasse	3	
5	Multipoint	5614316.214084596	5614316.214084596	EPSG:31467	UUID_f304505e-5b	Wohn-_oder_oe	2007-01-11	file:///L:/FB43_G	HA05382032	STANDAR	1003	9	Germany	Meckenheim	Street	Schlossstrasse	1	
6	Multipoint	5614412.063104834	5614412.063104834	EPSG:31467	UUID_de668e07-a0	Wohn-_oder_oe	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1003	6.33	Germany	Meckenheim	Street	Schlossstrasse	6	
7	Multipoint	5614365.79491574	5614365.79491574	EPSG:31467	UUID_d868d1a8-6a	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	STANDAR	1004	9	Germany	Town	Street			
8	Multipoint	5614300.7034243755	5614300.7034243755	EPSG:31467	UUID_56376b9d-ff	Wohn-_oder_oe	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1003	14.48	Germany	Town	Street			
9	Multipoint	5614419.774890375	5614419.774890375	EPSG:31467	UUID_36cd7f3d-c0	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	3.7	Germany	Meckenheim	Street	Schlossstrasse	6	
10	Multipoint	5614424.42734022	5614424.42734022	EPSG:31467	UUID_ebb5e106-fd	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	4.95	Germany	Town	Street			
11	Multipoint	5614376.788747654	5614376.788747654	EPSG:31467	UUID_c535c2d3-43	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	4.86	Germany	Town	Street			
12	Multipoint	5614470.105259357	5614470.105259357	EPSG:31467	UUID_c001a14b-16	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	3.91	Germany	Town	Street			
13	Multipoint	5614455.476162191	5614455.476162191	EPSG:31467	UUID_4e6df158-ef	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	3.67	Germany	Town	Street			
14	Multipoint	5614520.070503829	5614520.070503829	EPSG:31467	UUID_c48d0871-98	Wohnhaus_in_R	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1211	8.42	Germany	Meckenheim	Street	Petrusstrasse	58	
15	Multipoint	5614583.719162815	5614583.719162815	EPSG:31467	UUID_2124cfbb-39	Einzelhaus	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1311	7.93	Germany	Meckenheim	Street	Kottenforststrasse	2a	
16	Multipoint	5614489.986166264	5614489.986166264	EPSG:31467	UUID_d064d346-2a	Reihenhaus	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1331	9.04	Germany	Town	Street		b	
17	Multipoint	5614439.786667214	5614439.786667214	EPSG:31467	UUID_2c753fff-45e	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	6.7	Germany	Town	Street			
18	Multipoint	5614465.6303145075	5614465.6303145075	EPSG:31467	UUID_35bd4b9f-69	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	STANDAR	1004	3	Germany	Town	Street			
19	Multipoint	5614473.17944774	5614473.17944774	EPSG:31467	UUID_2ce3aa1d-b0	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	3.26	Germany	Town	Street			
20	Multipoint	5614551.070364167	5614551.070364167	EPSG:31467	UUID_462c4cff-45	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	5.33	Germany	Town	Street			
21	Multipoint	5614437.491658803	5614437.491658803	EPSG:31467	UUID_be7550bc-1b	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	6.65	Germany	Town	Street			
22	Multipoint	5614462.415698001	5614462.415698001	EPSG:31467	UUID_47c63ec2-55	Wohn-_oder_oe	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1003	5.3	Germany	Meckenheim	Street	Flerzheimer Strass	14	
23	Multipoint	5614482.757805383	5614482.757805383	EPSG:31467	UUID_55b1773c-bd	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	STANDAR	1004	3	Germany	Town	Street			
24	Multipoint	5614542.852743139	5614542.852743139	EPSG:31467	UUID_65522796-5a	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	STANDAR	1004	3	Germany	Town	Street			
25	Multipoint	5614534.242729546	5614534.242729546	EPSG:31467	UUID_67a849dd-ce	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	5.88	Germany	Town	Street			
26	Multipoint	5614555.678948558	5614555.678948558	EPSG:31467	UUID_80a0c382-3d	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	6.65	Germany	Town	Street			
27	Multipoint	5614627.614756219	5614627.614756219	EPSG:31467	UUID_36cf723d-47	Wohn-_oder_oe	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1003	6.93	Germany	Meckenheim	Street	Nordstrasse	11	
28	Multipoint	5614464.870840862	5614464.870840862	EPSG:31467	UUID_6ee57a3b-63	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	STANDAR	1004	3	Germany	Town	Street			
29	Multipoint	5614521.49843606	5614521.49843606	EPSG:31467	UUID_589ef428-5d	Wohn-_oder_oe	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1003	9.59	Germany	Meckenheim	Street	Suedstrasse	61	
30	Multipoint	5614533.283161821	5614533.283161821	EPSG:31467	UUID_6e1cc2e8-b5	Wirtschafts-_od	2007-01-11	file:///L:/FB43_G	HA05382032	LASER	1004	6.76	Germany	Town	Street			

Record: 21 Show: All Selected Records (0 out of 26545 Selected) Options

Fig. 4: Feature attribute table (viewed as shape file, dbf) translated from CityGML.

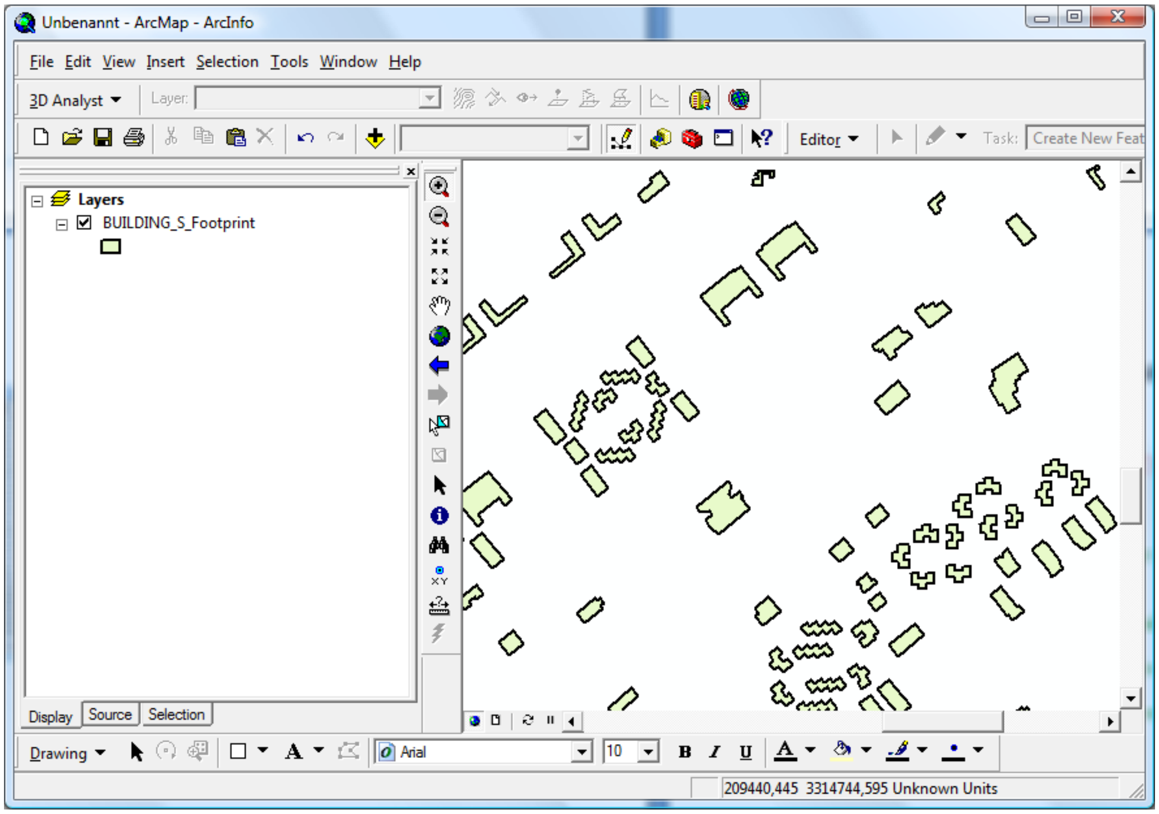

Unbenannt - ArcMap - ArcInfo
File Edit View Insert Selection Tools Window Help
3D Analyst
Layer:
Editor
Task: Create New Feat
Layers
BUILDING_S_Footprint
Display Source Selection
Drawing
Arial
10
209440,445 3314744,595 Unknown Units

Fig. 5: Generated building footprints viewed as shape file.

The CityGML adaptor is implemented using a SAX parser. The parser validates against the GML application schema, which can be provided by third parties. The second step is to create a memory internal DOM document which is then transformed into a Java3D scene graph. Java3D is used as internal representation for 3D geometries and appearances. An optimization step merges polygons that belong to the same `<gml:MultiSurface>` and flattens partly the scene graph structure, which improves the rendering performance significantly. Also appearances are merged, that means that instead of repeating the same appearance for each 3D shape, a single appearance is created which is referenced multiple times by the 3D shapes. The result is an optimized 3D representation for each GIS feature which is then encoded in VRML and stored in the data base along with the attributes.

7 Streaming of GML Content

In order to visualize and analyze the GML content, a special client software was developed, used as integrated client in the DSS and GPW threads. The GML data is streamed via the W3DS interface to the client. This section covers the basic principles of streaming and dynamic scene graph updating. Upon connecting to the W3DS server, the client analyzes the server’s meta data which contains information on supported protocols, formats, available layers, CRSs, styles, spatial extents, and on the service provider. Each layer contains a BoundingBox element, which describes the maximum spatial extent of all included features.

The streaming is based on a block-wise data update schema. This means that the space is divided into parcels or tiles of rectangular shape. For each tile a server request is created and sent to the W3DS server. The response is parsed and loaded into internal scene graph. This corresponds to a tree like scene graph structure stored in the clients memory and modified according to what needs to be displayed. Each tile is connected to a LOD trigger, which measures the distance from the virtual viewpoint to the center of the tile. If the viewer comes closer and the distance becomes smaller than the pre-defines threshold value, then the tile is activated and data for the tile’s extent is requested from the W3DS. A principle issue with this schema is that requests with adjacent bounding boxes may result in overlapping features at the border. That means that duplicate features at the tile borders must be loaded if the spatial selection logic performs an intersection method which selects all features that are inside or overlap with the requested bounding box. In order to avoid such effects, the W3DS server performs a spatial selection based on the feature’s center point. Features are selected if the center point is inside the requested bounding box. Thus adjacent tiles will not contain duplicate features.

For each layer the provided bounding box is translated in a base tile covering all data that may be downloaded from this layer. If the viewpoint comes closer, the LOD node is triggered, upon which 4 child tiles are generated arranged as a 2x2 raster splitting the area of the base node into 4. The child tiles are added to the base tile node. LOD triggers are added to the child nodes which have a threshold value which is generated from the size of the tile. The distance to these child nodes is then checked again and the scene graph is extended accordingly. This update procedure is continued until the leaf tiles with a predefined size have been reached. The leaf tiles will not extended, they contain the actual 3D model downloaded from the W3DS. If the viewer moved forward, then leaf tiles may come out of range and removed from the scene graph. This update procedure generates a tree structure of the scene graph. The spatial layout is a quad tree with small tiles near the viewpoint displaying a part of the GML content and larger tiles farther away from the viewpoint. Fig. 6 illustrates the block wise streaming schema.

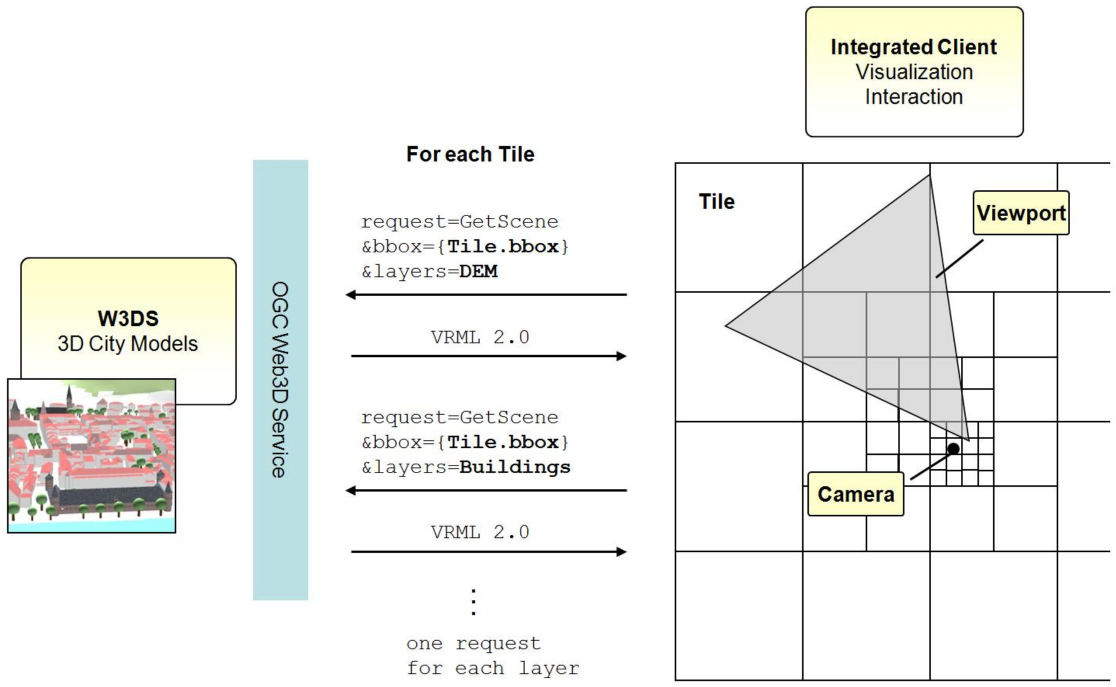


Fig. 6: Streaming concept. Diagram showing the interaction between client and W3DS server

Bibliography

[1] Guidelines for Successful OGC Interface Standards, OGC document 00-014r1

[2] Neis, P., A. Schilling, A. Zipf (2007): 3D Emergency Route Service (3D-ERS) based on OpenLS Specifications. GI4DM 2007. 3rd International Symposium on Geoinformation for Disaster Management. Toronto, Canada.

[3] Neubauer, N., M. Over, A. Schilling, A. Zipf (2009): Virtual Cities 2.0: Generating web-based 3D city models and landscapes based on free and user generated data (OpenStreetMap). GeoViz 2009. Contribution of Geovisualization to the concept of the Digital City. Workshop. Hamburg. Germany.

[4] Neubauer, S., Zipf, A. (2007): Suggestions for Extending the OGC Styled Layer Descriptor (SLD) Specification into 3D – Towards Visualization Rules for 3D City Models, Urban Data Management Symposium. UDMS 2007. Stuttgart. Germany.

[5] SCHILLING, A., S.NEUBAUER, A. ZIPF (2009): Putting GDI-3D into practice: Experiences from developing a 3D spatial data infrastructure based on OpenGIS standards for the sustainable management of urban areas. FIG Commission 3, International Workshop on 'Spatial Information for Sustainable Management of Urban Areas'. Mainz. Germany.

[6] Schilling, A., Basanow, J., Zipf, A. (2007): VECTOR BASED MAPPING OF POLYGONS ON IRREGULAR TERRAIN MESHES FOR WEB 3D MAP SERVICES. 3rd International Conference on Web Information Systems and Technologies (WEBIST). Barcelona, Spain. March 2007.

[7] Zipf, A., J. Basanow, P. Neis, S. Neubauer, A. Schilling (2007): Towards 3D Spatial Data Infrastructures (3D-SDI) based on Open Standards - experiences, results and future issues. In: "3D GeoInfo07". ISPRS WG IV/8 International Workshop on 3D Geo-Information: Requirements, Acquisition, Modelling, Analysis, Visualisation. Delft, NETHERLANDS