A Processing Pipeline for X3D Earth-based Spatial Data View Services

Thorsten Reitz* Michel Krämer† Simon Thum‡

Fraunhofer Institute for Computer Graphics Research IGD
Darmstadt, Germany

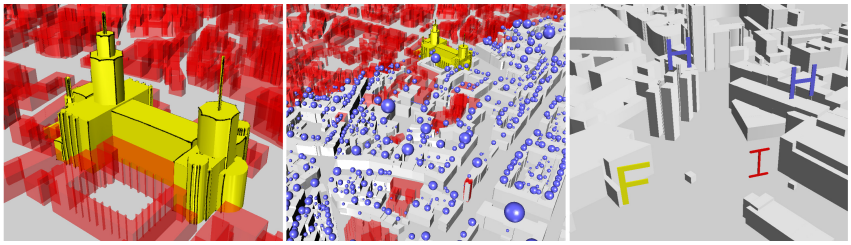

Figure 1: *Views of a 3D city model using our approach for X3D-based web services to highlight points of interest.*

Abstract

Over the last years, a high demand for scenario-specific visualizations of 3D urban models has evolved. At the same time, established service specifications do not yet provide the means to define 3D map products and to deliver them in suitable formats, since they are focused on traditional 2D map products. In this paper, we present an approach for the definition of a 3D urban model view service. This approach consists of a three-step process, in which original geodata is integrated, filtered and then transformed into various scene graph formats such as X3D.

We were able to maintain a high degree of compatibility with existing services and specifications such as Styled Layer Descriptors and the Web Map Service interface. The paper concludes with the experiences gathered from implementing and using this approach and provides an outlook as to how the lessons learned can be used in application and standardization.

CR Categories: I.3.6 [Computing Methodologies]: COMPUTER GRAPHICS—Methodology and Techniques; I.3.7 [Computing Methodologies]: COMPUTER GRAPHICS—Three-Dimensional Graphics and Realism; H.5.2 [INFORMATION SYSTEMS APPLICATIONS]: INFORMATION INTERFACES AND PRESENTATION—User Interfaces

Keywords: X3D Earth, GML, CityGML, SLD, visualization, Spatial Data Infrastructure, Portrayal Harmonisation

1 Introduction

In Spatial Data Infrastructures (SDIs), several classes of services have emerged and together form a stack, going from data publishing over processing to discovery and visual analysis services. One service class are the 'view' services, which transform raw vector or coverage geodata into a final, usable map product. The most popular type of these view services, the Web Map Service (WMS) has been specificed by the Open Geospatial Consortium (OGC), and numerous implementations have become available. In a WMS, the configuration of map products is done via external styling rules called Styled Layer Descriptors (SLDs), which in turn make use of Symbology Encoding (SE) elements for the specification of drawing rules.

This approach is quite robust and has been applied successfully for a wide range of map types. However, WMS and SLD/SE only take into account traditional 2D maps. There have been initiatives to extend the WMS to also provide 3D view rendering capabilities (in the form of the Web Terrain Service discussion paper [Open Geospatial Consortium 2001]) or actual 3D scene graphs (Web 3D Service discussion paper [Open Geospatial Consortium 2005b]), but none of these has been standardized or allows configuration of the view product in a way similar to the WMS.

We propose to fill this gap by defining and implementing a standards-based, 3D view service that can deliver a wide range of different 3D portrayal products. The presented approach is based on the OGC's Symbology Encoding Specification [Open Geospatial Consortium 2005a] and the original Web 3D Service proposal [Open Geospatial Consortium 2005b] to a high degree, to maintain compatibility across product definitions for 2D and 3D visualizations. This can for example become important when rendering a terrain grid in 3D using a texture that has to be created from a WMS with a style specification. In this way, interoperability in spatial applications based on Service Oriented Architecture (SOA) principles and a separate exchange of visualization information across networks and applications is ensured. This enables more integrating efforts like geospatial GRID computing infrastructures [Zhao et al. 2004], which our system could be part of.

We focus on delivering scene-graph oriented structures with object behavior and detailed appearance configurations such as VRML, X3D or JSR-178/M3G. These scene graphs are created in a three-step processing pipeline which includes

- the filtering of original geodata to the level required by a certain visualization,
- the application of 3D styling to determine geometry modifications, appearance and other parameters and

*e-mail: thorsten.reitz@igd.fraunhofer.de
†e-mail: michel.kraemer@igd.fraunhofer.de
‡e-mail: simon.thum@igd.fraunhofer.de

Web3D 2009, Darmstadt, Germany, June 16 – 17, 2009.
© 2009 ACM 978-1-60558-432-4/09/0006 $10.00

- the encoding into one of the aforementioned scenegraph formats.

This paper is organizsed as follows: Section 2 outlines core requirements and benefits expected of 3D view services. Section 3 provides an overview of relevant related work, focusing on the definition and exchange of 3D view products. Section 4 explains the overall concept behind our processing pipeline and explains the rules driving it. Section 5 outlines our portrayal rules definition approach. Sections 6 and 7 finally provide a discussion and conclusion.

2 Added Value of 3D View Services

Services producing application-specific 3D views have generated quite a big response with large user groups. Quite often, the visualisations and geodata preprocessing that is employed are specifically tailored to individual applications, which makes development and deployment of 3D view services comparatively expensive.

Especially cities such as Berlin, Hamburg or Stuttgart were early adopters of 3D view services for various purposes, ranging from urban planning over noise and microclimate simulation to touristic and marketing purposes. These have often generated high interest because of a certain novelty facor, but also because of much better opportunities to communicate complex interrelationships. Studies about different aspect of human perception have provided optimistic results for the usage of 3D visualizations compared to 2D ones (A list of the studies with a short explanation can be found in [Marcus et al. 2003]) and have also provided an impression of the range of different visualization options. Some of the studies suggest that the understanding of 3D structures is directly connected to the possibility of manipulation [Hubona et al. 1997]. Furthermore representations of structures in a three-dimensional space are said to be better memorized than 2D representations [Tavanti and Lind 2001], which would mean that the experiences from a 3D view service are more succinct than those from a 2D view service. This seems to be especially important in complex environments, such as presented by a 3D urban city model visualization.

When using appropriate data visualisation, human orientation can also be supported in a 3D view service. Individuals orient themselves in their environment by using the position of objects in relation to each other and to themselves [Hunt and Waller 1999]. Other test series deal with this egocentric orientation of people and support the assumption that an exact orientation from the particular position is stored and adapted while moving [Wang and Spelke 2000]. These reflections allow a better understanding of the associated mental representations and affirm the assumption of advantages of a 3D view services, in relation to a 2D view service, for at least some application areas (e.g. navigation).

3 Portrayal and Delivery of Urban Models

Our approach is based on the separation of visualization and data aspects in urban model management, and on delivering visualizations based on that separation. For 2D maps and geodata, approaches for the separation of geometric data and visualization descriptions can be found in OGC standards. Since the standards have to provide solutions not just for visualization, both data formats and web services are discussed and published. Regarding 3D city models, the Web Feature Service (WFS) and the Web 3D Service (W3DS) are the most relevant developments.

While a WFS allows the provision and manipulation of spatial data, a W3DS as a portrayal service is visualization-centred and can be used directly in application scenarios [Haist et al. 2006]. The W3DS is derived from the Web Map Service (WMS), which also offers a solution for rendering spatial data. The WMS is able to provide 2D images and uses simple geometry types (such as points, lines, polygons and raster), which are processed together with so-called Styled Layer Descriptors (SLD). In SLDs, the connection between features of geodata sets, grouped by so-called FeatureTypes, and rendering instructions is maintained. The actual rendering instructions are described in so-called Symbology Encoding (SE) documents. One important aspect of this newer specification is that it was always intended to be used with more services than the WMS. Consequently, SE describes general structures about how features can be visualized and was the base of the development presented in the following sections.

This direction was also chosen by another approach that complements ours quite well by Neubauer and Zipf [2007]. This approach focuses on the extension of the SLD towards more detailed surface descriptions, for example with the goal to specify how to render a DEM's inclination. It also includes parameters for 3D legend description, the inclusion of external 3D elements and geometry-generating or -alternating functionality. Finally, it also has means of controlling the display of volumetric data.

Another interesting approach is presented by [Quillet et al. 2006]. They derived vector approximations of the edges in textures, and provided models incorporating those vectors approximations. This preserves critical visual information while being much cheaper in terms of bandwidth, as compared to textures.

Dynamic 3D maps, mainly based on textures and the terrain model were presented by [Döllner and Hinrichs 2000]. This approach utilizes 2.5D terrain and 2D raster maps to generate 3D maps. The generation can be extended by icons or symbols, labels and shapes. Unfortunately, the architecture for the feature-based visualization is not described.

3.1 Exchange of View Product Specifications and Portrayed Data

Taking into account typical system environments in which 3D view services are being used, several relevant formats and standards for describing and displaying 3D city models are available. A general purpose format used to exchange spatial data is the Geographic Markup Language (GML) [Open Geospatial Consortium 2007a]. There are several profiles based on GML which extend the specification to store more domain-specific data. One of these is the CityGML exchange format [Open Geospatial Consortium 2008]. It models geometric as well as high-level semantic characteristics of 3D city models. As a GML profile, CityGML is data-driven and not visualization-based, even though it includes some elements from the W3C X3D standard [International Organization for Standardization 2008], such as textures. The main purpose of CityGML is to provide spatial data for various applications which need a well-defined topology and a high degree of semantic information, such as building usage, geometry types (wall, roof or ground) and so on. Applications using this information include urban planning or risk management. This modeling approach may also be useful for visualization (for example if all roofs shall be colored red), but in most cases it is not suitable for this specific purpose. Apart from that, CityGML does not represent a scene graph. It does not support transformation nodes and is not able to describe behavior that objects should show, e.g. reacting to proximity of a user.

In the visualization-oriented side of 3D geoinformation, KML [Open Geospatial Consortium 2007b], the Keyhole Markup Language, has been accepted quite widely. This is mostly due to its use in Google Earth and many Web 2.0 applications. Originally developed by Keyhole/Google, it has been accpted as a specification

by the OGC in April 2008. The language offers basic geometries and allows the integration of more detailed models via inclusion of COLLADA [Khronos Group 2008] structures. Therefore, KML has comprehensive means for describing the display of spatial data, but does not provide a way to store detailed semantic information like it is possible in CityGML. COLLADA is an open standard driven by the Khronos Group, an industry consortium whose members are leading companies working with multimedia data.[1] It is an intermediate format [Arnaud and Parisi 2007] used to transfer data from one Digital Content Creation tool to another and consequently contains alot of information that is not needed for a portrayed product. COLLADA is more comparable to GML as a data exchange format, rather than being a portrayal format.

Another exchange format which has been explicitly designed for use in the Web is the X3D, a standard developed by the Web3D Consortium. It has been declared International Standard by the ISO in 2004. X3D provides means to store the visualization of 3D data, including the ability to describe behaviors and interaction and thus enables more interactive applications. There is even support for streaming web delivery and it is therefore considered ideal for the use in web services [Arnaud and Parisi 2007], [Lin et al. 2007].

Originally, X3D has been designed to store 3D data in general and not explicitly spatial data. The X3D-Earth Working Group of the Web3D Consortium, founded in 2006 as a follow-on effort to the Web3D Geospatial Working Group, aims to define open standards to visualize real-world objects in a geospatial context. A special profile in X3D has already been implemented for that purpose. It is derived from GeoVRML, a spatial extension for the VRML format, and therefore allows associating real-world locations to objects in the X3D scene graph. Combining the comprehensive means to describe the visualization of 3D data with the possibility to a add spatial reference, X3D is not only suitable for web services visualizing simple scenes, but also for those delivering geospatial data. One of these is the aforementioned Web 3D Service, on which our approach is based.

4 A Scenario-oriented Web 3D Service

In current applications, clients are often developed specifically for a single scenario which then exchanges data with a scenario-specific service. This circumvents possible issue sources, but also limits the reusability of both clients and services.

Directly serving geospatial data to heterogeneous clients would require a broad range of processing capabilities on the side of the client, which would have to perform data and services integration. These often tend to be very specific to the environment, including factors such as past and current data aquisition, the way people are working, which data is being integrated and other factors.

A possible solution for this is a multiple-step processing chain that separates integration and visualisation that uses standard interfaces between the individual processing steps where available. In this way, generic clients can be used that work with one of the standardized 3D scene representation formats described in Section 3.1. Many service-side components can then also be made more generic, so that from scenario to scenario only declarative changes are necessary. Our concept for these generic 3D view service components is presented in the following sections.

4.1 The Web 3D Service

We have developed a novel approach to scenario-oriented portrayal as an extension to the OGC Web 3D Service (W3DS). In this approach, the service has detailed knowledge about a set of target scenarios. Each scenario is expected to match to a particular application area or workflow. It consists of simple name, a set of portrayal rules and a set of parameters detailing specific aspects of rule execution, such as the order in which rule are to be evaluated. The scenario has to be predefined by a user as part of a system installation.

In addition to any scenario-specific parameters, two parameters have been added to the W3DS implementation: 'scenario' and 'globalStyles'. The scenario parameter selects a scenario by its name, the 'globalStyles' parameter provides a list of SLD expressions. The name is chosen to avoid conflicts with the layer-specific styles parameter of the W3DS.

When answering a W3DS request, the service derives a view of data accessible to it using the present knowledge on the requested target scenario. The Web 3D Service is based on HTTP. Requests like the following can be used to obtain 3D scenes:

```
http://www.server.com/W3DS?version=0.3.0
    &request=GetScene&srs=EPSG:31467
    &scenario=municipal_planning
    &globalStyles=outlineStreets,noOrthophoto
    &format=model/x3d
    &bbox=3475398,5526431,3475684,5526593
```

This request calls the Web 3D Service's `GetScene` operation, which will load all objects in the given area (`bbox`) from the dataset, transforms them to the spatial reference system 'Gauß-Krüger Zone 3 (`EPSG:31467`)' if necessary and encodes it to the scene graph format X3D. The specifics of this encoding are defined in the scenario 'municipal_planning' (which could strip irrelevant details and favor simple models) and two styles which detail additional portrayal aspects.

These steps are also shown in Figure 2, which outlines the architecture of the scenario-oriented Web 3D Service: the client requests a 3D scene through this service. If it is the first request the control will be handed over to the Preparation Executor. This agent integrates and filters the original geodata according to formally defined business rules (see Section 4.2). The processed data is stored in the so-called Data Description (see Section 4.3). This is a structure containing meta-information about the objects in the dataset (e.g. an object's spatial location or its level of detail). After that, the Data Filter uses scenario-dependent filter rules to combine the information in the Data Description with the original geodata requested from the Preparation Executor. The Portrayal Service will then create the view, which consists of the filtered objects in their representation defined by Styled Layer Descriptors (SLD rules, see Section 5). The Web 3D service will then render this view into the exchange format requested by the client (for example VRML or X3D).

Consequently, only required information is passed from one agent to the next in the process. This enables us to save bandwidth and other resources by not processing any information irrelevamnt for the current scenario. Also, this makes scenario definitions viable as means to enforce security measures.

In particular, the extension is designed to address:

- filter issues, such as level of detail or layer selection
- spatial, thematical or security-motivated access constraits
- bandwidth use, e.g. by clipping of features

The scenarios we have analysed so far differ widely and require a broad range of integration, filtering and visualisation configuration options. Thus, to create a reuseable implementation, the actual

[1] `http://www.khronos.org/`

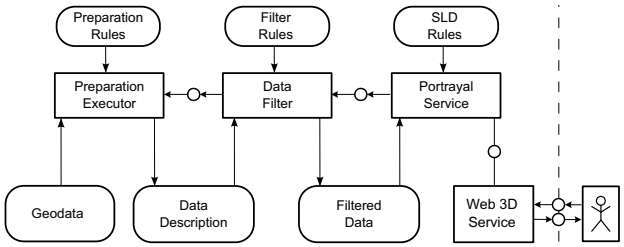

Figure 2: *The architecture of the scenario-oriented Web 3D Service: geospatial data will be filtered in two steps. The view will then be created according to SLD rules (Style Layer Descriptors) and encoded into a scene graph format like X3D before it is rendered to the client.*

details are stored in business rules. This technique is common in modern business applications, and has successfully been applied in geospatial data integration [Krämer 2008].

4.2 Business Rules in the Geospatial Domain

Business rules are especially useful when data and logic need to be decoupled. For our approach, this provides important benefits: The implementation merely needs to ensure that basic geospatial propositions can be decided, and corresponding actions be performed upon subjects.

A rule takes the general form:

```
WHEN proposition
THEN action(s)
```

For example,

```
WHEN item is-a Geoimage
     AND precision < 10 m
     AND user is Guest
THEN deny
```

would cause an action 'deny' if a detailed geoimage was requested by a guest account. Of course, this is rather simple. The rules processing makes sure complex conjugations can be formulated, evaluted and the respective actions executed.

Since the rules are maintained in the server's database, they are fairly independent of programmatic changes to the server itself. To enable the rules to be written in a natural way, their syntax has to match the geospatial domain, and should be adaptable to specific subdomains. Our rule processing allows not only to alter the syntax to match common geospatial propositions, but is also customizable to match the specific problems at target sites. This is not only aimed at complexity reduction, it also opens up the possibility that users may themselves alter rules to adapt to changing requirements.

4.3 The Data Description

When serving geospatial data from different sources to different users (in different scenarios), it can become a complex task to ensure consistency. Moreover, even slightly varying data quality can cause simple assessments to lead to overly complex propositions, which are hard to maintain. Finally, it is important to be able to provide results in a timely manner, since some of our scenarios include the requirement for highly interactive and performant visualizations.

To remedy those problems, our implementation establishes a central, runtime Data Description structure. This description is derived from source data using preparation rules which effectively build several indexes. It contains meta-information about the original dataset, including the spatial location of objects and the availability of different Levels of Detail (LOD). Afterwards, a data subset is derived using the filter rules associated with the target scenario.

Since the Data Description is derived from source data using preparation rules, it is possible to adapt to data quality changes without even touching portrayal or filter logic. On the other hand, one may change protrayal logic irrespective of actual data quality. This separation thus helps to keep rule complexity at bay. It is comparable to the way modern compilers handle multiple source languages and multiple target architectures. Ideally, this way we can reduce rules growth from $O(i * o)$ to $O(i + o)$, where i is the number of input data sources and o the number of scenarios we aim to provide.

The Data Description is designed to hold all the information needed to derive a scenario-specific view. In other words, it needs to cover the variability of the anticipated scenarios. It is also canonical, i.e. it contains exactly the information neccessary to derive views. For example, if the scenario description defines that the user may want request objects in different levels of detail in the vicinity of the city center, the Data Description will contain the information about which objects meet these requirements and which Levels of Detail are available for them. If the user then requests the Web 3D service to view all objects in their highest Level of Detail, the system will only have to load the requested geometries and skip those with a lower one. On the other hand, if the scenario description does not define such a constraint, the Data Description will not need to contain this information at all.

Given suitable preparation rules, the Data Description alleviates differences in source data quality or constitution. This enables filter rules to be simpler, focused on scenario-specific issues and therefore reduces complexity. Preparation rules, on the other hand, are also more focused towards specific issues, such as data quality or determination of non-trivial meta information needed when deriving views.

The Data Description, and therefore its validity, depends solely on the input data and the set of preparation rules. This ensures that, if required, the Data Description can be regenerated to take new or changed source data into account. Usually the Data Description will be cached, so it does not need to be regenerated on each request.

Example The Data Description also helps to integrate spatial information. As an example, suppose we have data with models of varying Levels of Detail (LOD) which are located in different data sets from different sources. It becomes hard to judge which models belong to the same feature, and users should not see different, intersecting models of the same feature.

In this situation, filter rules can be used to judge instantly which objects should pass the filtering stage to be included in the final view product. For these, simply keeping the LOD in the Data Description of a model does not suffice, as there might be spatially conflicting models. These conflicts can be determined using spatial indexes, but these would have to be built.

A solution in this case is to create preparation rules which attach to models not only what their respective LOD is, but also which other LODs are available for the same feature. For example, consider the following simple heuristic:

```
WHEN
     Model a intersects Model b
```

```
     AND volume (a union b) * 0.2 >
         volume (a difference b)
THEN
     attach LOD(b) to a.otherLOD
     attach LOD(a) to b.otherLOD
```

This approach infers enough information for the filtering rules to decide which models to select based on LOD criteria, without doing spatial lookups again:

```
WHEN
    is-preferred(Model a.LOD)
    OR NOT is-preferred(a.otherLOD)
THEN
    select-model(a)
```

Due to the general nature of rules, this technique is adaptable to a wide range of problems. The key here is to find an appropriate Data Description, which carries 'just enough' information. When this is given, costly processing can be done in the preparation step.

5 Defining a 3D View Product using SE3D

After analysing the service-side integration and filtering of objects applicable to a given scenario context, the third component used for transforming the filtered data into a fully portrayed product is presented in this section.

Rules on how to transform geographic data into an understandable map have been around for centuries. Such rules are now described in a formal way by using so-called Styled Layer Descriptors (SLD) and Symbology Encoding (SE). These allow a clear separation of concerns between data collection and management and the visualization of data. Traditionally, SLDs are used to transform 2D vector and raster geodata into a 2 vector or raster image, usually delivered as SVG, PNG or JPEG. However, the same principles can be applied, with only very minor additions and modifications, to 3D+ source data. Just as with WMS and normals SLD/SE, this means that a client can use the W3DS interface and can either let the service decide on the portrayal, or he can selected predefined styles, or he can even submit user-defined styles (using HTTP-POST), if there are no security concerns.

5.1 The structure of SLDs and SE

The basic structure of a SLD/SE XML document is centered around references to layers, which are defined as a collection of features. The linkage between the styling information and features is realized via `FeatureTypes`, i.e. classifications of objects. An important aspect to consider is that a feature can be assigned to more than one `FeatureType`, for example from different `FeatureType` Catalogues (conceptual schemas), so that classification according to multiple domains of usage is possible. At the same time, this means that styling rules can be conflicting on a single feature.

SLD/SE defines two document types: The 'style library' and the 'dynamic configuration'. The style library consists of the assignment of styling rules and features via `FeatureTypes`. The layers in the style library can be referenced within the dynamic configuration whose main elements are `NamedLayers` and `NamedStyles` (see figure 3). Due to this separation, a high flexibility is achieved. In the style library different style configurations for certain layers are persisted which are activated by references within the dynamic configuration. Also, it is possible to define new layers and thematic groups within the dynamic configuration so that adaptations can be easily integrated by changing the dynamic configuration.

The specific styling is defined by using `UserStyle` elements. These visual styles can be accessed by the identifying name, e.g. 'streets'. They contain rule statements for binding symbolizers to filters. `Symbolizers` are central elements and define the appearance of features using Scalable Vector Graphics (SVG) vocabulary. `Filters` define conditions based on attribute definitions of `FeatureTypes` so that features can be drawn according to their attribute values. As an example, a classification attribute for a 'road' type can be used to draw a single line or a more complex pattern.

SE defines different types of `Symbolizers`, each one describing how a feature is visualized on a map. It determines the shape and properties like color and opacity. Currently there are five types of Symbolizers defined in SE: The `LineSymbolizer` styles a 'stroke'-like geometry along a line feature. A `PolygonSymbolizer` draws a polygon. The `TextSymbolizer` is used for text labels and their formatting. A

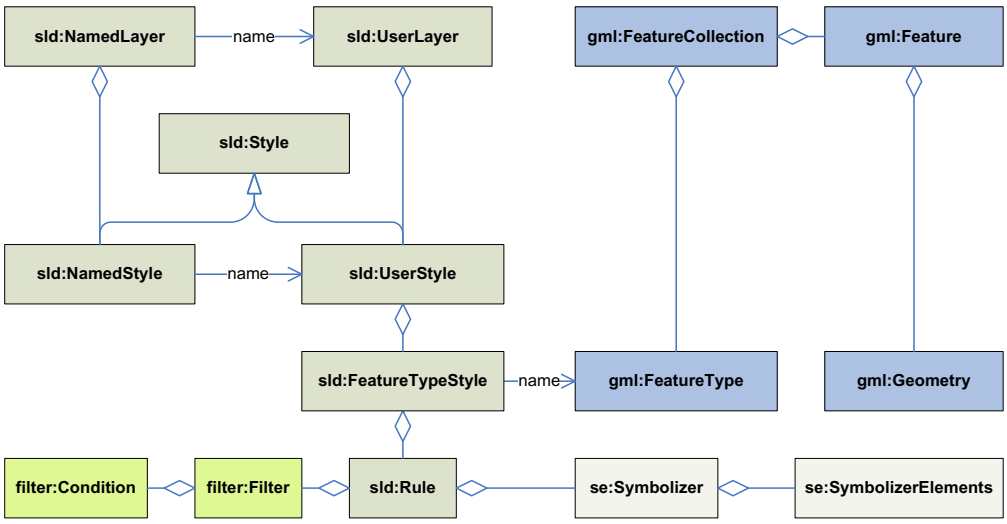

Figure 3: *Class diagram depicting the SLD/SE structure and the assignment to* `Feature` *via* `FeatureTypes`.

symbol at a given point can be drawn by the `PointSymbolizer`. For describing how to render raster/matrix-coverage data, the `RasterSymbolizer` is used.

5.2 Symbolizer extensions for 3D

For the use of SE with 3D urban models, a subset of the SE specification was extracted on which extensions were designed. It should be noted that in this approach, all 2D elements can still be used where applicable, such as for styling a map used as terrain texture. The meaning of the standard elements is not changed.

An important aspect was to have a single visualization definition structure that could be used on all levels, from the persisted geodata up to dynamically changing states in interaction with the data. This approach requires a precedence ordering on available visualization descriptions. So, as shown in Figure 7, if no representation definitions for the current state (*Object Status Appearance*) or a client-provided definition (*Style Library Appearance*) could be found, the *persisted appearance* is used. In case of no existence of a data model appearance, a *default appearance* takes effect.

The main extension point in the specification is the `Symbolizer`. When creating subtypes of this class, it was decided to categorize the Symbolizers according to their purpose with the following two general types:

- `AppearanceSymbolizers` change the material (aspects such as color, specularity, transparency) and texture representation
- `GeometrySymbolizers` change geometrical characteristics of features.

These added Symbolizers reuse existing SymbolizerElements such as `Texture`, `Fill`, `Dimension` and `Geometry` to ensure flexibility. The following 3D-specific Symbolizers were ntegrated into SE and implemented so far:

AppearanceSymbolizer With the definition of `AppearanceSymbolizers`, color, transparency/opacity and object-oriented effects can be controlled. Figure 4 shows a building which uses the fill `SymbolizerElement` as well as an effect element which defines a textured line stroke to be applied on the edges.

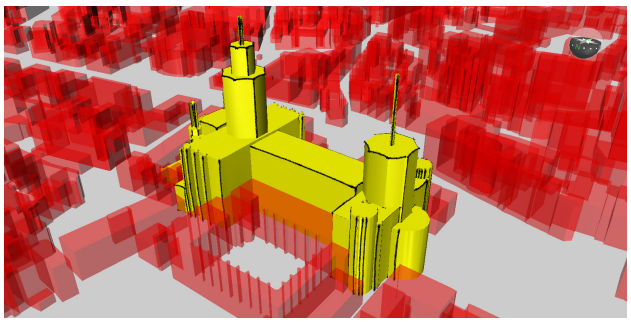

Figure 4: *A building which has been assigned an `AppearanceSymbolizer` defining a color and enhanced edges. Another `AppearanceSymbolizer` is used for all other buildings which are displayed in a half-transparent red color (opacity = 0.5).*

AlternateGeometrySymbolizer This Symbolizer defines a prototype geometry which is displayed instead of the feature's explicit geometry. It can be used for several visualization purposes especially for navigation. Figure 5 is a screenshot showing an important location in a city where points of interest have been replaced by respective symbols. Other buildings are using the `AppearanceSymbolizer` (gray color).

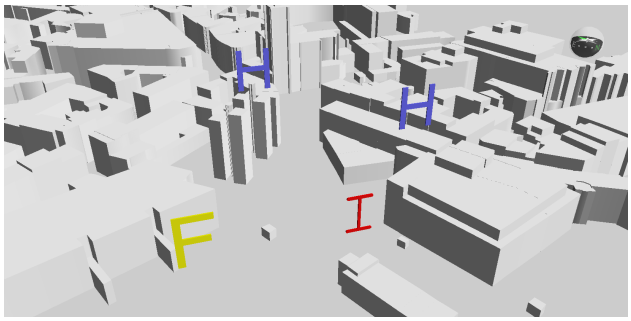

Figure 5: *Using the `AlternateGeometrySymbolizer` points of interest can be replaced by respective symbols according to their feature type. "H" represents hotels, "I" is a tourist information point and "F" is used for a restaurant (food).*

AdditionalGeometrySymbolizer This Symbolizer is used to display additional prototype geometries to a feature. These geometries are located in the universe with the `DimensionSymbolizerElement`. Figure 6 shows an example where the size of the additional geometry is representing the inhabitants of the building. Both, for this Symbolizer and for the `AlternateGeometrySymbolizer`, either GML elements can be used, or external geometries referenced. Supported external structures are currently GML, VRML, X3D shape and several others, with the recent addition of the Generative Markup Language. This latter one was added as optional element at least in parts to the SE3D schema, so that instead of referencing literal geometry descriptions, functional descriptions and complex transformations can be used.

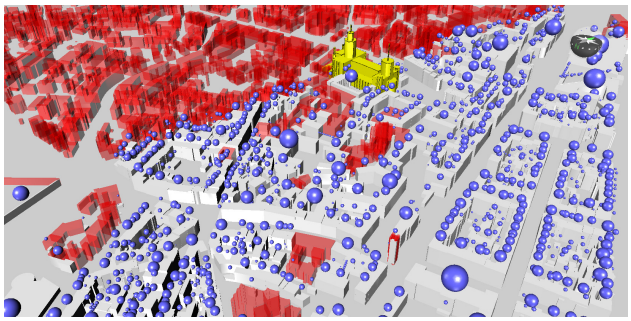

Figure 6: *The `AdditionalGeometrySymbolizer` allows displaying spheres whose volumes are proportional to the inhabitants of the respective buildings.*

TextureSymbolizer Compared to 2D ones, 3D urban models tend to have a more complex geometry model, with topology being of high importance. In most cases, boundary representations are used to persist models. In higher levels of detail, textures are added and geometries usually consist of several parts with different colors and material attributes. This leads to a conflict that our approach had to resolve: so-called geodata textures and appearance textures. Geodata textures are generated in term of spatial data acquisition

and can be treated as spatial data. As an example, CityGML as a data format allows the defintion of texture images. By comparison, appearance textures are only used to support or to realize visualization effects and can include far more than just a diffuse, statically-mapped image map.

The `TextureSymbolizer` was integrated to control the visualization of textures. Textures can be switched on, also the opacity can be set. It is also possible to add an `EffectSymbolizerElement` to apply effects like posterization, color depth reduction, contrast manipulation and other image manipulations. Finally, an important option that has been added to the current version is to include a WMS call, including full 2D SLD and SE capabilities, whose result can then be mapped as a texture.

RasterSymbolizer The `RasterSymbolizer` is taken from the SE specification and is used to define texturing and coloring of 2.5D elevation grids. Since the geo-reference of this data is known, the `RasterSymbolizer` (equivalent to the `TextureSymbolizer`) does not have to define the texture mapping. As an example, the coloring of a digital elevation model is accomplished by defining a color map with threshold values.

TextSymbolizer With the help of the `TextSymbolizer` text elements can be placed into the virtual universe. This Symbolizer is already defined within SE and only gets an additional position element for positioning in 3D.

5.3 Processing of SE data

In a first step, the SE documents are converted into instruction objects to be used at runtime. Both the predefined style library and the dynamic configuration are imported and consolidated. This comprises resolving the references declared in the dynamic configuration, as described before. Furthermore, new user-defined styles may be introduced in the dynamic configuration.

An important step now is to decide which styling has to be applied to which feature. More often than not, there is more than one styling that can be applied - one based on persisted parameters or one based on a client-defined style, one based on the current state of the object (e.g. when it is currently actively selected). Since a single feature can have more than one `FeatureType`, it also has to be decided at the same level which of the styles to apply. For this intention a special data structure called `FeatureStyleInstructions` (FSI) was added. This instruction contains all the information needed to alter the visual appearance of features at runtime.

`FeatureStyleInstructions` are the implementation specific complementary form of rule elements. They act on the same level and contain a set of `FeatureTypeNames` as a filter for the feature assignment. Furthermore, `FeatureStyleInstructions` contain instructions which are on the same semantic level as Symbolizers.

With this extension, runtime object state (e.g. a selection) can be visualized as well. These state-based representation definitions are stored in a separated XML file. Its structure is mostly equal to the other SE files, but containing layers are not referenced to any `FeatureTypes`. Another aspect to take into account is reclassification: the `FeatureType` of features within a scene may be changed. In that case features are updated and assigned instructions are possibly changed in order to avoid inconsistency in the visualization. The interaction with the scene and the resulting possibilities of changing features causes the necessity to define a system that manages and synchronizes the `FeatureStyleInstructions`.

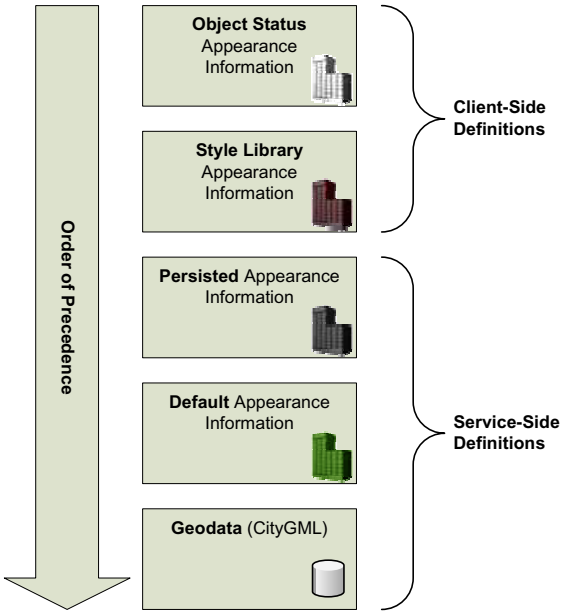

Figure 7: *Tiers of visualization instructions to be used by the system.*

6 Discussion

During the implementation of the presented approach within the CityServer3D technology and research platform, we found several aspects which warrant a discussion, such as the usage of the Web 3D Service interface and the inclusion of X3D Earth specifications. Regarding the first, we found the original W3DS proposal to be basically sufficient in many aspects; what we found lacking though is the profiling of WMS SLD/SE concepts to the 3D world, as well as a concept on how to stream parts of a response provided by a W3DS. The styling profile is an aspect we handled within the presented approach, whereas the streaming remains a somewhat open topic.

The usage of X3D offered us several advantages compared to other formats that we used in our scenarios. Traditionally, one of our main export formats for visualization was VRML, but VRML support is decreasing in editing and viewing applications and VRML is also quite limited in expressivity. Therefore, X3D is a well-suited replacement for VRML (including the X3D Earth specifications which add a geospatial context). In comparison to KML, we found that it was important to our users to be able to export their scenegraphs correctly in various spatial reference systems, something where X3D Earth offers more expressiveness. Compared to COLLADA, X3D is a more view- and interaction-oriented format, and it also enjoys good support through freely available viewers, which means a larger audience can be targeted with a X3D-enabled service. One example of implemented interactions is a scenario where clicking on a feature leads to a call of a WMS' GetFeatureInfo operation, and consequently the display of the object's attributive data - as a function of the scenegraph, not of the client application. These behavioral aspects go as far as including filters that can be applied on the base of attributes of a feature. Should an attribute change, the visualization will be updated accordingly. This capability can be used to visually group thematic collections on the base of their attached domain- or metadata, such as coloring all features newer

than 1956 in yellow. Even continuous updating of the scene based on sensor data is possible using X3D.

Another aspect worthy of discussion is the two-stage filtering we use. We found it quite helpful since it enables us to save bandwidth in the communication between service and visualization applications. However, the biggest advantage is that it enables us to separate data integration in the first step and data visualization in a second step, so that harmonized filter and portrayal rules can be used.

The data filtering also allows increased performance. Fewer objects have to be loaded and thus applying SLD rules is optimized. The actual performance depends on several factors like volume of the underlaying dataset (number of objects, number of textures, ...) and the complexity of the user defined scenario description. However, the presented software architecture divides a request into several small steps. First of all, this reduces the number of rules in the scenario description (as described in Section 4.3). Furthermore, the architecture allows caching the Data Description (and to a certain point) the filtered data, so consequtive requests can be executed very fast. The performance can be increased even more on multicore systems since the architecture is based on pipes and filters. Components like the Preparation Executor, the Data Filter and the Portrayal Service are agents which can run in parallel. For example: as soon as the Data Filter has prepared an object it sends it to the Portrayal Service. The latter can then apply its SLD rules to the object while the Data Filter already produces the next one.

One open topic of the approach is its limitation to the granularity of features. This limitation has been imposed by the GML/CityGML specification and it was decided that bypassing the existing structures would both be too much of an intrusion into the specification and would also increase complexity by a large degree. It is currently not possible to apply SLD rules to multiple representations of a single feature like different geometries for different Levels of Detail. A finer granularity should be aimed for.

Other approaches, such as the Generative Modeling Language[2], go far deeper in this aspect, allowing the full specification of the geometry and appearance on every level of the object tree making up a scene. There are also other generative approaches which use L-Systems, usually for specific object types such as vegetation or buildings. These technologies can complement the approach described in this paper quite well, as they are concentrating on a level of the visualization stack below the pure representational level; they actually generate whole geometries from very reduced input data (or even completely randomly), while the extended SE approach handles the adoption of the visualization to the current user's requirements and context.

7 Conclusion and Outlook

The presented approach builds a bridge between the classical, layer-oriented geographic visualization and a more flexible, tree-oriented 3D computer graphics concept. This is achieved using styling capabilities based on `Filters` and `FeatureTypes`. Also, a conflict resolution strategy for the cases where the increased flexibility leads to conflicting drawing rules is provided. It allows the creation, exchange and adoption of 3D maps on the base of existing and established standards.

Furthermore, our approach takes into account all processing phases required to produce a portrayal product of a 3D urban geodata set. This includes the selection of the content that is important for a given client's context, the definition of client-side visualization and effects as well as the description of how objects should react to interaction and the provision of this content in various scene graph formats such as VRML, X3D and M3G. This three-step processing pipeline has been fully implemented and was subject to tests based on several scenarios, including an urban plannning one, a car navigation one and an environmental information one.

Finally, there are also several points in which the appoach can be improved. One of these is to increase the extent to which adaptability to a user's context is possible. This might include a very high level pixel shader description language allowing users a very specific, even though complex, way of defining how their 3D visualization should look like. Furthermore, the first steps to set up a working group on the topic of SLD3D and SE3D to pursue the creation of an official standard profile of SLD and SE for the use with urban models have been initiated.

[2]also GML, see `http://www.generative-modeling.org/`

References

ARNAUD, R., AND PARISI, T., 2007. Developing web applications with COLLADA and X3D: A whitepaper, Sony Computer Entertainment.

COORS, V. 2001. Feature-preserving simplification in web-based 3D-GIS. In *International Symposium on Smart Graphics*, A. Butz, Ed., 22–27.

DÖLLNER, J., AND HINRICHS, K. 2000. Dynamic 3D maps and their texture-based design. In *Computer Graphics International (CGI '00)*.

HAIST, J., REITZ, T., AND COORS, V. 2006. 3D city models for navigation applications. In *Innovations in 3D Geo Information Systems*, Springer Verlag, Berlin, Heidelberg, New York, A. Abdul-Rahman, Ed., 11.

HUBONA, G., SHIRAH, G., AND FOUT, D. 1997. 3D object recognition with motion. In *CHI '97 extended abstracts on Human factors in computing systems*, ACM Press, 345–346.

HUNT, E., AND WALLER, D., 1999. Orientation and wayfinding: A review. Office of Naval Research, Arlington.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2008. X3D Architecture and base components Edition 2, ISO/IEC FDIS 19775-1.2:2008.

KHRONOS GROUP, 2008. COLLADA 1.5.0.

KRÄMER, M. 2008. *Regelbasiertes Management verteilter, heterogener Geodaten*. Master's thesis, Fachhochschule Gießen-Friedberg.

LIN, Q., NEO, H. K., ZHANG, L., HUANG, G., AND GAY, R. 2007. Grid-based large-scale Web3D collaborative virtual environment. In *Proceedings of the twelfth international conference on 3D web technology*, ACM, Perugia, Italy, 123–132.

MARCUS, A., FENG, L., AND MALETIC, J. I. 2003. 3D representations for software visualization. In *Proceedings of the 2003 ACM symposium on Software visualization (SoftVis '03)*, ACM Press, 27ff.

NEUBAUER, S., AND ZIPF, A. 2007. Suggestions for extending the OGC styled layer descriptor (SLD) specification into 3D towards visualization rules for 3D city models. In *Proceedings of the 26th Symposium on Urban Data management (UDMS)*.

OPEN GEOSPATIAL CONSORTIUM, 2001. Web Terrain Server 0.3.2. Discussion paper.

OPEN GEOSPATIAL CONSORTIUM, 2005. Symbology Encoding Implementation Specification 1.1.0.

OPEN GEOSPATIAL CONSORTIUM, 2005. Web 3D Service 0.3.0. Discussion paper.

OPEN GEOSPATIAL CONSORTIUM, 2007. Geographic Markup Language 3.2.1.

OPEN GEOSPATIAL CONSORTIUM, 2007. Keyhole Markup Language (KML) 2.2.0.

OPEN GEOSPATIAL CONSORTIUM, 2008. City Geographic Markup Language (CityGML) 1.0.0.

QUILLET, J., THOMAS, G., GRANIER, X., GUITTON, P., AND MARVIE, J. 2006. Using expressive rendering for remote visualization of large city models. In *Proceedings of the eleventh international conference on 3D web technology*, ACM, Columbia, Maryland, 27–35.

TAVANTI, M., AND LIND, M. 2001. 2D vs 3D, implications on spatial memory. In *Proceedings of IEEE Symposium on Information Visualization (INFOVIS'01)*, 22–23.

WANG, R., AND SPELKE, E. 2000. Updating egocentric representations in human navigation cognition. In *Cognition*, vol. 77, 215–250.

ZHAO, P., CHEN, A., LIU, Y., DI, L., YANG, W., AND LI, P. 2004. Grid metadata catalog service-based OGC web registry service. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, ACM, Washington DC, USA, 22–30.