

Generating Semantic 3D Models of Underground Infrastructure

Erick Mendez¹, Gerhard Schall¹, Sven Havemann¹, Sebastian Junghanns²,
Dieter Fellner^{1,3,4}, Dieter Schmalstieg¹

¹ Graz University of Technology ² GRINTEC GmbH ³ Fraunhofer IGD ⁴ GRIS, TU Darmstadt

Abstract. We present a modeling pipeline to create interactive three-dimensional visualizations from 2D geospatial databases. Our application area is the underground infrastructure owned and operated by utility companies. Consequently, our modeling framework creates urban models based on real world data used by these companies rather than on synthetic data. The 3D models are encoded in a scene-graph that is a mixture of visual models with semantic markup, used for interactive filtering and styling of the models. In our approach, the actual graphics primitives are generated on-the-fly by scripts that are attached to the scene-graph nodes. This combination of two previously unrelated techniques – semantic markup in a scene-graph and generative modeling – allows retaining semantic information until very late in the rendering pipeline. This is the crucial prerequisite for enhanced visualization effects and interactive behavior without compromising interactive frame rates.

Keywords. Augmented Reality, Generative Modeling, Semantic 3D Models, Urban Models.

1 INTRODUCTION

Large geospatial databases are the result of hundreds of person years of surveying effort, and rendering engines constitute highly advanced and optimized software toolkits. What is needed to connect both is commonly called *transcoding* in the geographic information systems (GIS) community: a process of turning raw geospatial data, which are mostly 2D, into 3D models suitable for standard rendering engines. Thus, the required transcoding is not simply a one-to-one conversion from one format to another. In the presented system detailed 3D models are obtained from 2D information by procedural 3D modeling.

The transcoding from semantic attributes in the geospatial database into purely visual primitives necessarily implies information loss. It is therefore important to find the right point in the pipeline for the transcoding: If the semantic information is discarded early, it cannot be used for interaction with the user at a later stage of the pipeline. If it is discarded late, this induces the significant overhead of a repeated interpretation of the semantics at runtime, and adversely affects performance. We call this the *transcoding trade-off*. The contribution of this paper is a

modeling framework that allows for optimization of the transcoding trade-off.

The framework transcodes geospatial data into three-dimensional interactive visualizations using a combination of conventional scene-graph with semantic markup and on-the-fly generation of procedural models enhanced with an embedded stack-based scripting language. The tight integration of these techniques allows choosing the transcoding and representation methods for every object individually and dynamically based on the available high-level semantic information. This approach also allows defining the visualization styles with relation to the semantic markup and thus independently of any actual object structures. Our application example is an Augmented Reality (AR) visualization of underground infrastructure, but the techniques in this paper are generally applicable. The contribution of this paper is therefore composed of two parts, the systems part, described in section 3, and the application part, described in section 4.

2 RELATED WORK (SIDEBAR)

The input parameters to a procedural modeling system can be either artificial or derived from real world measurements, such as surveying or satellite images. Moreover, the generated models can be tightly controlled by the input, such as facades modeled from textures [Müller07] and tangible modeling [Anderson00], or loosely controlled, such as synthetic plants [Prusinkiewicz90], buildings [Müller06] or castles [Berndt06]. Our users expect a reliable representation of the real world, so strict dependence on real-world measurements and user-controlled parameters is necessary.

The models in this work are generated from data exported from geospatial databases in the standard Geography Markup Language*, (GeographyML). CityGML [Kolbe05], for example, is a specialization of this language for 3D visualization of textured architectural models, but requires a special browser. Instead, our work aims at using a standard scene-graph system. The export of geospatial data usually takes place via a Web Feature Service (WFS) which encodes the information in vector format inside a

* www.opengeospatial.org

GeographyML instance. There has been other work on forwarding database information to scene-graphs, for example X-VRML [Walczak03], but these types of approaches generally do not involve on-the-fly procedural modeling.

References

- [Anderson00] D. Anderson, et al., “Tangible interaction + graphical interpretation: a new approach to 3D modeling,” Proceedings SIGGRAPH 2000, ACM Press, 2000, pp. 393-402.
- [Berndt 06] R. Berndt, et al., “3D Modeling for Non-Expert Users with the Castle Construction Kit,” Proceedings ACM/EG VAST 2005, ACM Press, 2005 pp. 49-57.
- [Kolbe05] Kolbe T. et al., “CityGML – Interoperable Access to 3D City Models,” Proceedings of the International Symposium on Geoinformation for Disaster Management,” Springer Verlag.
- [Müller06] P. Müller, “Procedural Modeling of Buildings,” Proceedings SIGGRAPH 2006, ACM Press, 2006,
- [Müller07] P. Müller, et al., “Image-based Procedural Modeling of Facades,” Proceedings SIGGRAPH 2007.
- [Prusinkiewicz90] P. Prusinkiewicz and A. Lindenmayer “The algorithmic beauty of plants,” Springer-Verlag 1990.
- [Walczak03] K. Walczak, Cellary W., “X-VRML for Advanced Virtual Reality Applications,” IEEE Computer 2003, pp. 89-92

3 THE GEOSPATIAL INFORMATION PIPELINE

In the following subsections we present the ingredients of our system. These are applicable to a variety of different scenarios. In particular in section 4, we present an underground infrastructure visualization augmented reality application.

3.1 Design Considerations

The overall objective of a 3D model for visualization of geospatial data is to provide comprehensible visualizations of the assets in question. Since there is a large variety of geospatial objects and possible visualization styles, we desire a system architecture which allows content types and visualization styles to be added as plug-ins of the actual client. Ideally, a compatible 3D browser should be capable of loading and displaying self-contained content. Moreover, the browser should be capable of displaying user interfaces for the selection and manipulation of parameters influencing the application. While these requirements apply to many visualization applications with complex datasets, we specifically aim at addressing the following requirements:

- **Appealing shape.** For example, intersections of branching pipes must be continuous. This cannot be achieved by simply converting raw vectors from the

geospatial database into cylindrical tubes, i.e., more advanced modeling methods are required.

- **Level-of-Detail (LOD).** In particular for low-performance mobile computers, strict control of geometric complexity is essential. The displayed content should adapt dynamically and continuously (no popping) in geometric complexity in a view-dependent way.
- **Information filtering.** Displaying the full content of a large database will likely lead to screen clutter [Höllner01]. Therefore the user must be able to filter data based on semantic (e. g., object type) and spatial information. To achieve this we employ magic lens techniques [Bier93].
- **Flexible styling.** A user may choose styles for individually selected objects or groups of objects. Styles are also dependent on the viewing situation: AR displays may require changing the styling parameters depending on the quality of video see-through and registration. To allow flexible handling of styles, the styles should be stored with the content and not be a feature of the specific 3D browser. Clean separation of styles from content makes styles reusable across multiple types of content.
- **Progressive information revealing.** Semantic level of detail is a technique that can manage screen real estate efficiently by using multiple representations of the same object which progressively reveal more visual and functional detail. For example, objects can be arranged in containment hierarchies, such as in the case of underground infrastructure where cables are arranged inside encasings that in turn are contained in tubes and so forth.

A straightforward conversion of the geospatial data into a static polygonal representation can address only some of these requirements. And even a dynamic multi-resolution tessellation will not help in addressing the needs for interactive manipulation and control. Such interactive capabilities could be added as a custom feature to the 3D browser, but this approach makes content and browser highly interdependent and defeats easy extensibility.

Therefore, we have opted for a combination of techniques which are implemented as extensions to a conventional scene-graph library, Coin3D (www.coin3d.org). New object types are handled through built-in interpretations of the scene-graph structure, and do not require any modifications of the scene-graph browser itself.

1. **Scene-graph level:** Scene-graph markup enables us to attach the semantic attributes from the geospatial database. The markup also provides the hooks for interactive high-level control provided

by the user, and allows filtering and styling as well as a certain degree of semantic LOD control.

2. **Object level:** To provide high quality tessellated objects with geometric and semantic LOD control, we have added a new type of scene-graph node with a lightweight embedded interpreter for a stack based language, the Generative Modeling Language (GenerativeML).

Semantic attributes are passed as parameters from the scene-graph to the GenerativeML nodes during scene-graph traversal. For details on scene-graph traversal itself, refer to [Strauss92].

In the remainder of this section, we present the components of our information pipeline: the transcoding from GeographyML to our scene-graph format, the GenerativeML, and the scene-graph markup that links scene-graph and GenerativeML together.

3.2 Transcoding

The first step in our information pipeline is a transcoding pass. This means a change of the data format, in our case from a GeographyML encoding to a scene-graph enriched with GenerativeML nodes. GeographyML is based on the notion of *features*, which include underground and above-surface infrastructure. A feature consists of several semantic attributes and a number of geometrical attributes describing the actual 2D coordinates. The result of the transcoding is a scene-graph description with per-feature grouping of shape objects and semantic attributes. The shape objects refer to embedded GenerativeML scripts for dynamic parametric objects, and to Coin3D classes for static non-procedural shapes. The transcoding pipeline focuses on common features found in the underground infrastructure.

3.3 Scene-graph markup with semantic attributes

The result of the transcoding step is a scene-graph with semantic attributes. This has an outstanding advantage: Traditionally, in order to highlight all objects of a certain type inside a scene-graph (e.g., by changing their color to red) the respective material properties of all affected nodes must be changed somehow. This could be done by changing a material field of the node, or by inserting material nodes into the scene-graph. However, a much better strategy is to change only one “style node” for all the desired nodes early in the traversal order. The changed parameters are propagated automatically when the scene-graph is traversed for rendering: Each affected node updates its styling because its attributes have been touched [Mendez06].

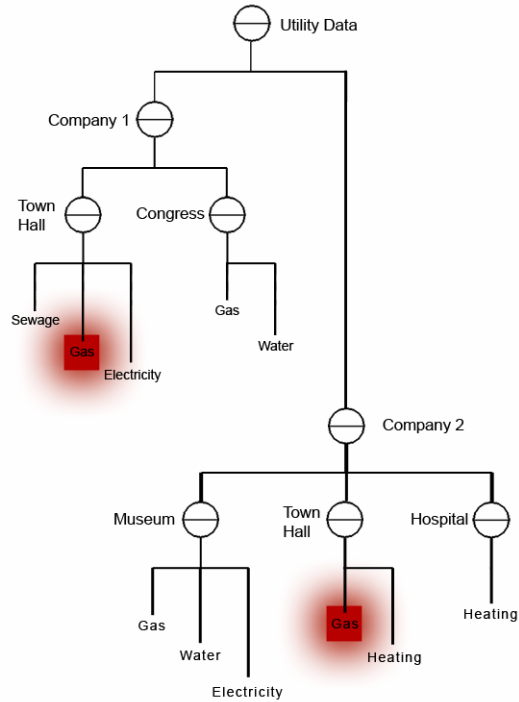


Figure 1: Scene-graph marked up with semantic attributes. Assume that reconstruction in the town hall requires welding, which must not occur near gas. The combination of attributes (town hall + gas) can be automatically used to locate and highlight dangerous objects. This happens during a single-pass scene-graph traversal and does not require additional effort.

Objects in our scene-graph are not only grouped by the hierarchy of the graph, but also by families defined by their semantic attributes. Objects from the same family do not have to belong to the same part of the scene-graph hierarchy – they can be scattered throughout the scene-graph and still be jointly affected (Figure 1).

Additionally, the family membership of an object is not explicitly given, but is the result of an aggregation of semantic attributes encountered while traversing the path from the scene-graph’s root node to the object itself.

This mechanism works exactly like other traversal states such as transformation matrix or material bindings. Semantic attribute nodes encountered during traversal add or overwrite arbitrary key-value pairs in the current attribute set. This set is propagated downwards with traversal, and can be queried by any node aiming to be environmentally sensitive.

Upon Traversal

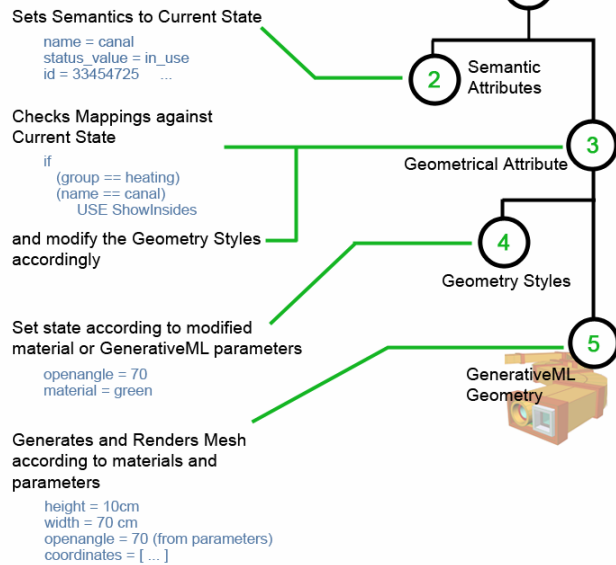


Figure 2: Upon traversal, every geographical feature aggregates its semantic attributes to the traversal state, and checks whether the current total of the semantic values match a style mapping. If so, predefined styles are fetched accordingly. Therefore, the rendering style of the shape nodes reflects its semantic attributes.

More specifically, a styling node relies on a style map to modify appearance parameters for the following shape nodes. The style maps are also hierarchically organized in a way similar to Cascading Style Sheets (CSS) used in XHTML. Entries in the style map are “style” sub-graphs consisting of nodes influencing appearance, which are dynamically inserted before the object to be styled during scene-graph traversal. In that way, object appearance changes dynamically with the associated semantic attributes. The advantages are:

- **Preserving semantics:** Semantic attributes from the geospatial database are retained until the actual rendering traversal, and used in this stage to determine appearance. Existing styles can be applied to new types of objects if they use compatible attributes.
- **Referencing:** Potentially complex combinations of semantic attributes can influence the appearance. Consider a situation where construction requires welding, which must not happen close to a gas pipe for security reasons. A style demarking “danger” can be automatically selected if attributes “gas” and “welding” are applied to an object group in spatial proximity.
- **Interactive Styling:** Selected semantic attributes can be exposed in the user interface, since they are just key-value strings. The user can directly modify attribute values and thereby influence the visualization. E. g., filtering of distracting objects can be achieved by

setting the style for the distracting object category to “off.”

- **Parameter Forwarding:** The semantic attributes can be forwarded as parameters to the GenerativeML engine creating the geometric primitives. More details are given in the following section.

Geographical features are encoded as small sub-graphs inside the scene-graph containing both its semantic and geometrical attributes (Figure 2). Semantic attributes add to, set, and modify key-value pairs in the traversal state. This provides to any geometrical attribute the ability to check whether any mapping exists that match its semantics. Every geometrical attribute is in itself a sub-graph containing a styling and a content node. If a mapping exists it modifies its styling node in order to affect the appearance of the content node.

3.4 GenerativeML

In our system the semantic attributes not only affect the appearance of 3D objects, but also their construction parameters. The technical device we use is the Generative Modeling Language (*GenerativeML*), a simple stack-based scripting language for creating parametric 3D models on-the-fly [Havemann05] (also see www.generative-modeling.org; note that its short name is GML actually, but it is called GenerativeML in this paper to distinguish it from the Geography Markup Language, whose acronym is also GML). Its original purpose is to serve as a general exchange format for procedural models, e.g., as a file format for encoding the construction history of complex objects. It is capable of generating large amounts of geometric data out of compact descriptions. Since the syntax is similar to Adobe’s PostScript language, GenerativeML can be thought of as a kind of “3D PostScript.”

Currently, the main surface representation used in GenerativeML is the combined boundary representation (cB-rep), polygonal meshes whose faces can be non-convex and of arbitrary degree. The special feature that makes the cB-Rep suitable for free-form modeling is that edges have a crease attribute to distinguish sharp and smooth edges: Faces containing smooth edges are rendered as Catmull/Clark subdivision surfaces, not as polygonal faces. The embedded OpenGL renderer is highly optimized and provides on-the-fly tessellation and LOD.

The idea behind generative modeling is to replace objects by operations. Consequently the cB-rep meshes are not represented as lists of geometric primitives, but generated using Euler operators. They are issued directly as operators in GenerativeML, or through higher-level modeling operations such as extrusion (also a built-in operator).

The GenerativeML interpreter and its integrated OpenGL renderer are contained in a shared library that can be linked

with any OpenGL application, e.g., a scene-graph engine. In this way GenerativeML has been embedded into Coin3D as a new node communicating with the GenerativeML engine. Script code is stored and maintained in a string field of the GenerativeML scene-graph node in the .iv file. This code can be executed during traversal to produce 3D models of pipes and tubes on-the-fly. This greatly reduces scene-graph size and permits to vary construction parameters to reflect changes on the semantic level, such as highlighting or database queries.

Note that generative modeling is complementary to scene-graph with semantic attributes: The strength of a scene-graph is the management of large scenes, providing tools such as a scene hierarchy, spatial structuring and flexible scene traversal. In contrast, GenerativeML operates on the object level. Its purpose is to create massive amounts of geometry based on rules and parameters. The interface between both software components is just the set of semantic attributes that are passed as model parameters to GenerativeML.

3.5 Visualisation with filtering and annotations

The semantic markup can directly be exploited in the rendering stage for information filtering. Information such as object categories, ownership, topological relationships and so on may be used, for example, to hide or to highlight individual objects.

Frequently, users wish to apply filtering to a spatially bounded area, which can be efficiently addressed through the use of Magic Lenses [Bier93]. We use magic lenses that are capable of changing the presentation styles of objects not only depending on their containment in the lens, but also on their semantic markup [Mendez06]. Objects may be color-coded or made transparent inside the lens, or it may change the parameters passed to GenerativeML, e.g., to open up the pipes in the lens area.

The magic lens in Figure 3 was designed to resemble an excavation hole. We reduce the saturation and opacity of objects outside the lens. Conversely, inside the lens only objects with certain semantic attributes are displayed. This allows the user to localize the region of visual disambiguation and at the same time retain the spatial context of the surrounding assets.

4 APPLICATION TO UNDERGROUND INFRASTRUCTURE

Consider the following task from the public utility sector. Utility companies need to provide private contractors, such as construction companies, with spatial information about their assets. This is essential to avoid damaging existing infrastructure during digging. Therefore, a common duty for field workers employed by utility companies consists of spray painting spots on the ground where digging should take place. Orientation at the construction site is

performed with paper maps plotted in advance using the utility company's GIS.

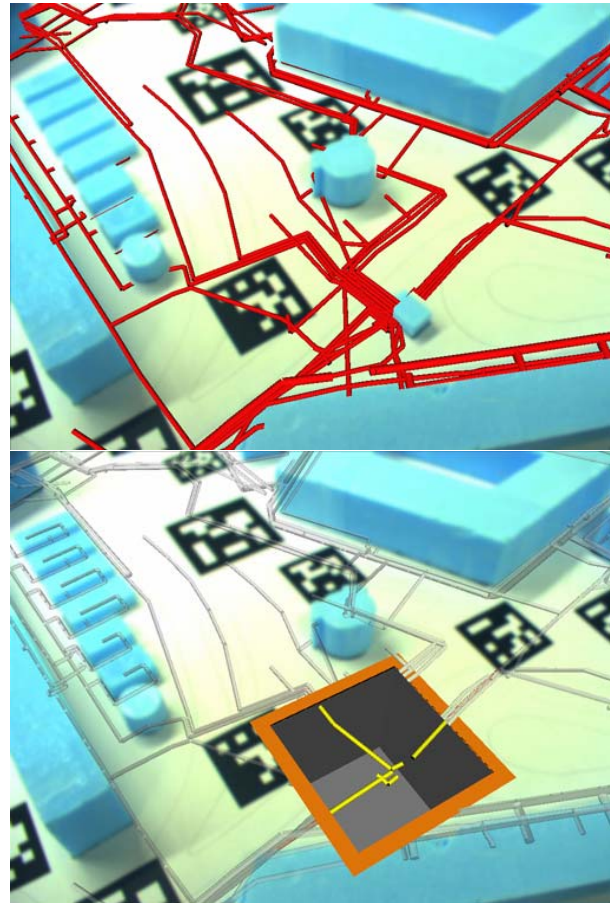


Figure 3: The propagation of semantic information allows combining spatial with semantic filtering: The screen clutter in the upper image is reduced by applying an “excavation” magic lens which only shows selected assets (lower image), while the surrounding area shows a toned-down version of all assets. Both images are produced with a table-top AR setup with real architectural scale models (light blue).

4.1 The GIS information model for underground infrastructure and its transcoding

Utility infrastructure, like underground water or gas distribution systems, is arranged in canals, divided in multiple tunnels at different depths. Depth is either given as an attribute value, or must be estimated based on heuristics (e. g., telecommunications are typically in the first layer at 0.5 m depth). The depth at which each utility system is buried also depends on the condition of the terrain.

The central object considered in the transcoding pipeline is the *pipe*. All types of pipe-shaped infrastructure, be it electricity, gas, water, sewer or heating pipes are abstracted by the same common geometric attribute. However, they

can be visually discriminated given their semantic attributes.

One pipe can consist of several segments. To avoid the creation of excess polygons, the transcoding deletes collinear points as well as duplicate points automatically. It is also possible to quantize coordinates for LOD generation, since millimeter level precision is typically not required in the visualization. The non-geometric attributes such as ID, purpose or ownership are converted to semantic nodes inserted on the scene-graph, while the geometric attributes are passed to the GenerativeML nodes. Tunnels are described analogously to pipes but with a rectangular rather than a circular cross-section.

Note that pipes are in fact hierarchical, since pipe objects are typically enclosed by other pipe objects, such as electricity lines or gas pipes arranged inside tunnels. The containment hierarchy is given through semantic attributes and not just derived geometrically. This permits to avoid rendering sub-pipes if the containing pipe is set to “opaque”. The procedural logic of the GenerativeML permits to open up tunnels (Figure 5) to reveal their interior, and only then the contained pipes are rendered as well.

In addition to pipes the underground infrastructure consists of special facilities, e.g., gate valves, water hydrants, T-fittings, and so on. We blend these facilities with the generated 3D environment based on special rules, using a library of pre-modeled 3D objects. Buildings are also included to provide geographic context. They are generated procedurally from their footprints in the transcoding step by simple extrusion, using a semantic height attribute, or a default value if this attribute is unavailable.

The result of the transcoding process is a scene-graph with semantic attributes as described in section 3.3.

4.2 Adaptive modeling of assets

All types of pipe-like objects are created procedurally from a small set of parameters, including the poly-line determining overall shape, radius, thickness, and type of the pipe, radius of curved pipe segments, collar radius, thickness, and so on (Figure 4).

The basic problem we face is to connect two straight pipe segments with a curved pipe. The curved portion of a pipe is a circle segment starting at the dashed line before the actual joint in a given normal distance from the pipe axis. The circle midpoint is computed as the intersection of the two dashed lines. Then the circle segment is sampled and radial line segments are produced, along which a vertical profile of choice is oriented to create a rotational sweep. The result is a cB-rep control mesh that is used to produce polygonal facets and Catmull/Clark subdivision surfaces. Note that the first and last faces of a pipe have sharp (red)

edges to produce a crease, resulting in flat faces with a B-spline boundary. Note that bad parameter sets, e.g., caused by bad surveying of the geospatial data, can lead to self-intersections of the control mesh (pipes too wide, curve radius too small, segment angle too acute). GenerativeML has no automatic mechanism to determine the valid range of parameters.

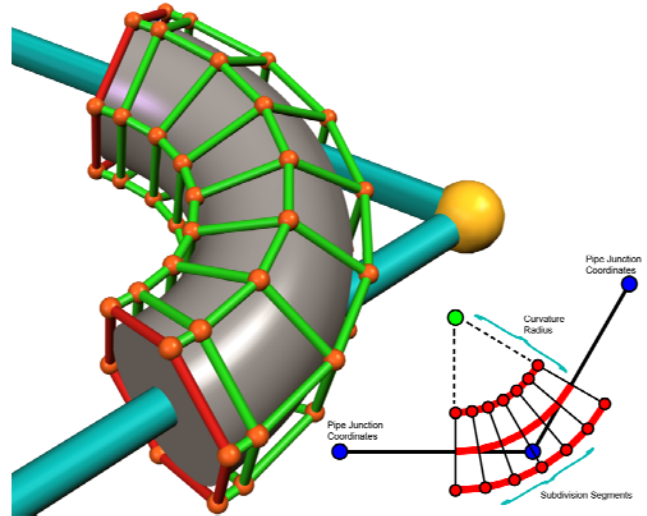


Figure 4: To achieve nice rounded edges, a circle segment is computed to connect subsequent tubes. The circle segment is offset to either side and sampled to obtain radial line segments (inset). The circular profiles (n-gons) are then converted to double-sided faces that are finally connected.

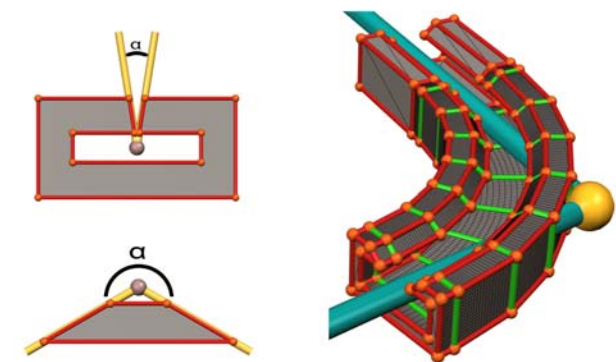


Figure 5: The open profiles are also created procedurally. The opening angle can vary smoothly and may even be animated. To open up any object, two sampled circle segments with different radii are connected. In this case we use “canals” with a rectangular profile.

The power of GenerativeML is that it supports processing chains quite efficiently. The stack is a flexible way of passing data produced by one function as input parameters to the next. This is exploited to create different types of pipes simply by passing different profiles to a connecting function. Figure 5 and Figure 6 show canals created from a

rectangular profile with a user-defined, interactively controlled opening angle. Note that the profile, and hence the vertex count, for open pipes or canals is variable as it depends on the opening angle.

A minor complication comes from the fact that sub-pipes often run next to each other (offset pipes). They seem to run in parallel, but the truth is that in curves, offset pipes share the same curve midpoint but have different radii depending on whether the curve goes to the left or right.

This problem can elegantly be solved by re-using the procedure used for creating the pipes, with the offset from the main poly-line as another control parameter. It enables dealing uniformly with hierarchical groups of pipes/canals that follow the same route. Specifically, creating the geometry for a cut-open hierarchical grouping requires only to propagate the opening angle.

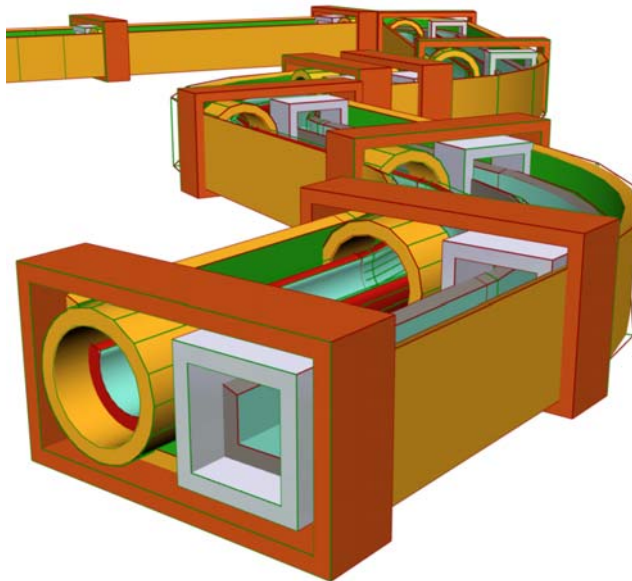


Figure 6: Collars are added to better distinguish the curved joints between straight pipe segments. Additionally pipes, canals and tunnels may be hierarchically arranged.

Collars around the pipe endings are important cues that help understanding the spatial relationships, so that also pipes of different types can be joined preserving a consistent visual appearance (Figure 6).

Per-object geometric LOD is generated dynamically per-frame by adjusting the subdivision-depth of the tessellation depending on the covered pixel-area and the surface orientation and curvature (Figure 7).

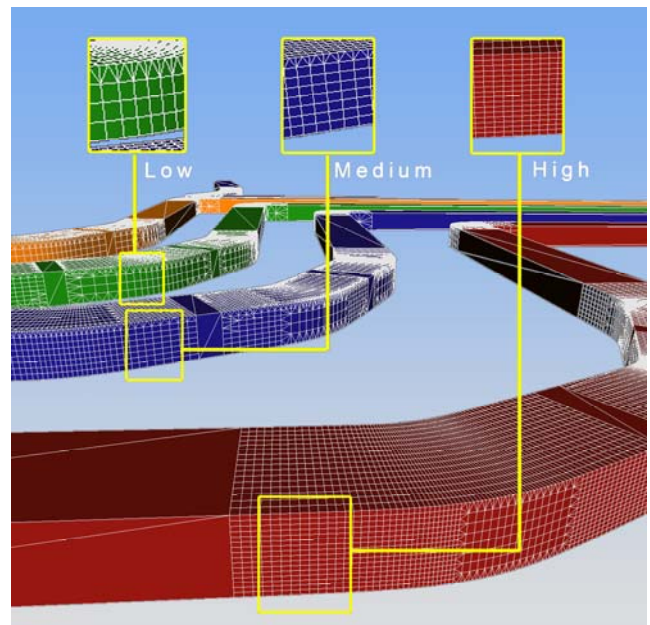


Figure 7: Interactive LOD allows for an increase of visual fidelity while minimizing the performance impact. LOD selection is automatically handled when the GenerativeML code is called.

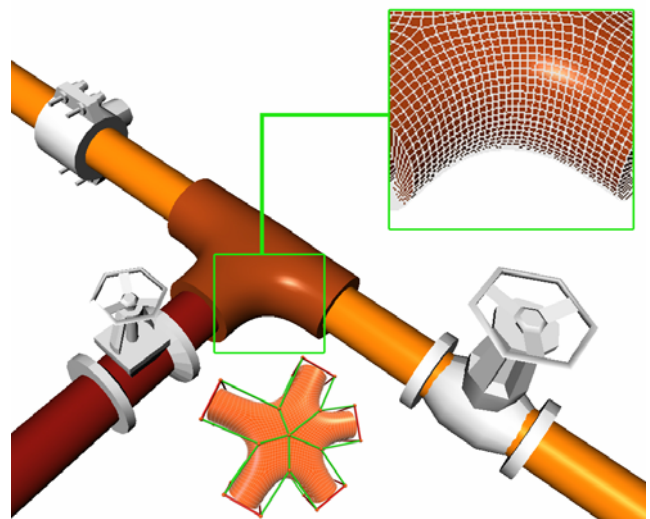


Figure 8: This image shows a composite model of a T-junction created by GenerativeML together with valve models from a stock library. Through the use of subdivision surfaces, GenerativeML can create complex junction shapes such as the 5-way junction in the inset (lower middle).

The interactive LOD enables us to generate more visually pleasing images. For example, intersections of pipe junctions are given as sharp connections in the geospatial data. GenerativeML models can soften these junctions by further subdividing the resulting mesh. Figure 8 shows an example of a combination of the different modeling techniques. The pipes and the T-junction were modeled

using GenerativeML, the smooth joint makes the connection clearly visible. The white objects are predefined models from a shape library, which are low-resolution to not impair the rendering performance. The geometrical position of all these models, as well as their orientation and semantic attributes are also extracted from the geospatial database by transcoding.

4.3 AR-style visualization

Next we will sketch a practical solution for the public utility sector scenario from the introduction of section 4. Assuming a WFS-enabled geospatial database, the field worker can perform on-line queries while working on the construction site by using a wireless mobile computer that sends the current GPS position. The server returns a set of geographical features encoded in GeographyML. Currently, field workers employ handheld computers which are capable of displaying 2D map information more or less equivalent to conventional paper maps.

The position of pipes, ducts, valves, alignments and other underground infrastructure are exclusively based on data from the utility company's GIS, since they cannot be acquired using aerial or terrestrial photography. Above-surface information such as building footprints, roads or property lines is also taken directly from the geospatial database. We are mainly considering the delivery of the final visualization as part of a video-see through AR display, but the visualization techniques are also applicable in desktop Virtual Reality applications in a planning office.

The outdoor AR system reported here has been previously used as a tourist navigation aid [Schmalstieg2007]. In this paper, we concentrate on the modeling techniques, which have been newly developed to visualize underground models.

The images in Figure 9 and Figure 10 show geospatial data superimposed in real-time on a video feed, using the AR interface. Figure 10 also shows a user with a robust handheld AR device used for the outdoor work. The AR interface is designed around a handheld tablet PC equipped with a sensor array consisting of GPS, an inertial orientation tracker and a camera to show a video-see through augmentation of the environment. This mechanism of styling objects given their semantic markup is a powerful tool in those scenarios where we do not have control over the arrangement of objects in the data.

Take for example the scenario illustrated in Figure 9: The visualized information represents heating assets, organized in tunnels and canals containing pipes. A simple task is to request all the objects of type canal to show their interiors. This action sets the GenerativeML parameter "opening-angle" to 70 degrees only on those objects with matching semantic attributes, i. e., canals. Notice that not all the objects have opening angles - those who have, are set to green (b).

This action has revealed the pipes inside the green conduct which is cut open. Alternatively, we may change the transparency of objects so that they show their interiors as illustrated (c).

Field workers usually need to refine their search using additional attributes, such as objects that are currently in use, or that belong only to a certain category. In the dialog in (c), the user would like to open only those tunnels used for heating and that are in use.

Different styles may be assigned to objects. Such styles are created by an application designer and may convey subjective meanings such as "dangerous" or "important." In (d), the user instructs the system to highlight an object of a specific id, which in this case is targeted for repair.

The benefit of this visualization is twofold. Firstly, underground infrastructure can be located and visualized before an excavation is performed – although in this particular example, for the sake of illustration it was done after. This allows "X-ray vision" of buried assets, for example, to locate damages or during construction planning.

Secondly, visual guidance during excavation activities and of technical infrastructure can be achieved. Information about properties of the pipes can be retrieved on demand - like purpose, specification, length and material of pipes as well as construction date, companies involved in maintenance and so on. The technician no longer needs to contact the company's back office or consult printed documentation when information related to the pipes or to the construction site is required. Due to our data transcoding technique all properties of the pipes that are stored in the corresponding source geospatial database are available for information visualization. Consequently, the on-site situation can be assisted more intuitively.



(a)

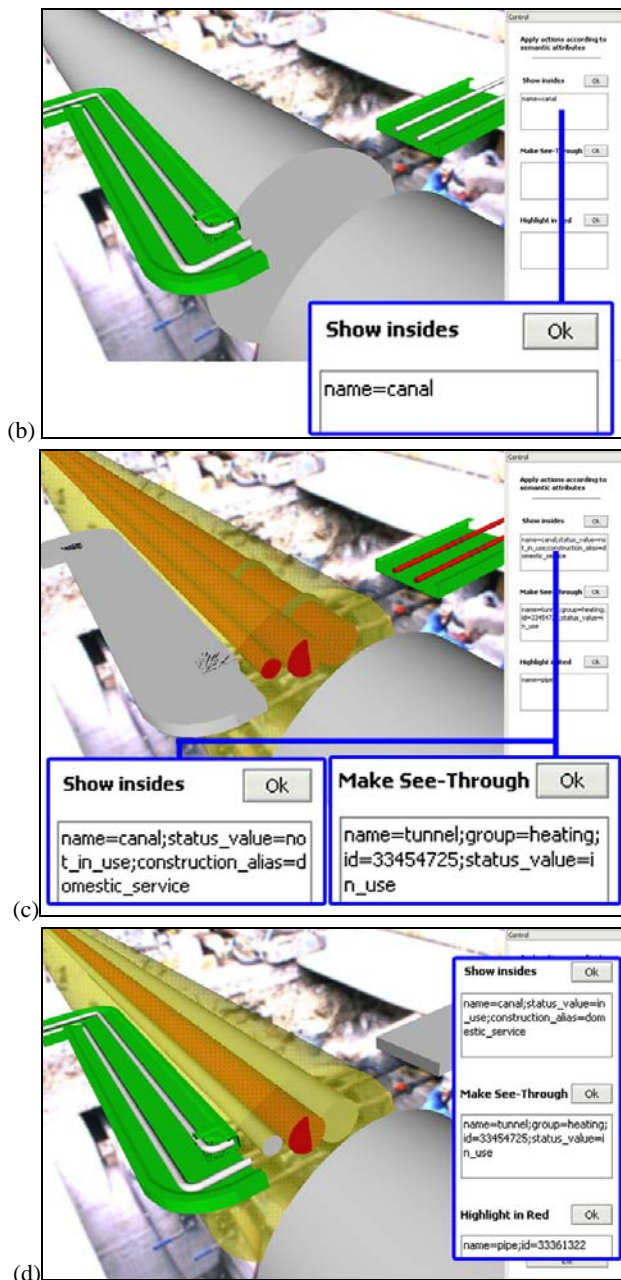


Figure 9: Illustrates how the superimposed geospatial underground model can be visualized in a series of different images. The first image shows the excavation at the construction site without augmented geospatial data. The supply and return pipes of the district heating pipe infrastructure are clearly visible. The task of the technician inspecting the construction site is to renew some of these pipes.



Figure 10: View through the AR display deployed outdoors. In the image a second AR user with a handheld AR display can be seen. On the left, wireframed building models are displayed to retain spatial context.

5 DISCUSSION

5.1 Transcoding trade-off analysis

In order to study the effects of the *transcoding trade-off* we have performed a series of tests involving different pipe networks as well as mesh sizes for three separate stages of the transcoding. The following conditions were used in the tests:

- **S0: Static.** Semantic attributes are ignored, one single mesh holds all generated 3D objects.
- **S1: Adaptive on Transcoding.** Semantic attributes are evaluated by the transcoding to generate material values for every geospatial feature before being deployed to the 3D browser.
- **S2: Adaptive on Traversal.** Semantic attributes are preserved and evaluated during scene graph traversal where template mapping and material bindings take place.

The S1 condition performed on average at 64% compared to S0 ($\sigma = 1.052931$). On the other hand, S2 performed on average at 57.5% compared also to S0 ($\sigma = 1.55204$). These tests indicate that the overall performance of S1 and S2 will remain regardless of the size of the mesh and will mainly depend on the separation of objects in sub-graphs. As expected, S1 and S2 performed at similar rates (6.5% difference). The reason for this is that the number of traversals is the same but the overhead on S2 is caused by the template mapping during traversal.

5.2 Limitations of the current system

The information flow in our system is strictly one-way from the scene-graph to the GenerativeML nodes that generate the geometry procedurally. The scene-graph itself

is static in a sense that at runtime, no nodes are added or deleted; only the connections between these nodes (or sub-graphs) can be changed at runtime. More flexibility could be obtained if it was possible to create also parts of the scene-graph procedurally.

For large networks is desirable that the scene-graph can be loaded progressively: nodes are refined by inserting a sub-graph, or sub-graphs are collapsed into a single node, based on proximity and visibility as well as on semantic queries.

A limitation on the technical level is that Coin renders each object immediately when it encounters it during traversal. Since a pipe has several different materials, this implies a great number of material state changes. This does not allow rendering optimizations such as collecting from different objects all faces with the same material.

The semantic markup allows the styling of objects during the traversal itself without prior knowledge of their position in the graph. This can cause caching problems since every sub-graph has to be reformatted to reflect its semantic style mappings upon traversal.

5.3 Conclusion and future work

We have presented a system for extracting geospatial data of underground infrastructure and for creating interactive models for AR displays. While the system is still in development, preliminary feedback from users of the utility sector is encouraging. We believe that the approach of retaining semantic information and using it not only for textual annotation, but also for influencing 3D shapes and visualization styles, has wide applicability in modeling of human-made structures from existing legacy data.

The decision on the amount of information to be reformatted during transcoding implies a trade-off between performance and flexibility. Preserving semantic information down to the traversal stage by far increases the flexibility to apply visual and modeling changes to the objects in the scene. But it also implies a decrease in performance. The number of traversals increases, since

every node is adapted to reflect its semantic mapping by traversing a styling node prior to the geometrical content. A better strategy was to consider particular user tasks (as in the case of infrastructure maintenance) which would reduce as much as possible the number of semantic attributes that need to be preserved by the transcoding pipeline.

Future work will involve hardening the system for real field use, and establishing full editing capabilities which allow returning changes observed in the field into the geospatial database directly from within the AR system.

Acknowledgements

This work was sponsored by Österreichische Forschungsförderungsgesellschaft *FFG* under contract no. BRIDGE 811000, and the Austrian Science Fund FWF under contract no. Y193 and W1209-N15. We also thank GRINTEC GmbH for providing geospatial data from Smallworld™.

References

- [Bier93] E. Bier, et al., "Toolglass and Magic Lenses: the see-through interface," Proceedings SIGGRAPH 1993, ACM Press, 1993, pp. 73-80.
- [Havemann05] S. Havemann "Generative Mesh Modeling. PhD thesis," TU Braunschweig, Germany, 2005.
- [Höllerer01] T. Höllerer, et al., "User Interface Management Techniques for Collaborative Mobile Augmented Reality," In Computers and Graphics 25, 2001, pp. 799-810.
- [Mendez06] E. Mendez, et al., "Interactive Context-Driven Visualization Tools for Augmented Reality," Proceedings ISMAR 2006, IEEE Computer Society, 2006, pp. 209-218
- [Schmalstieg07]. D. Schmalstieg et al., "Managing Complex Augmented Reality Models," IEEE Computer Graphics and Applications, July/August 2007.
- [Strauss92] P. Strauss and R. Carey. "An object oriented 3D graphics toolkit," Proceedings SIGGRAPH 1992, ACM Press, 1992, pp. 341-349

Biographies

Author Name: Erick Mendez

Bio: Erick Mendez is a research assistant and PhD student at the Graz University of Technology. He obtained his Masters of Science from the George Washington University and his Bachelor of Science from the Benemerita Universidad Autonoma de Puebla. His current research interests are on context information as well as on visualization techniques for augmented reality systems.

Contact Info: Technische Universitaet Graz, Institute for Computer Graphics and Vision, Inffeldgasse 16a/308, A-8010, Graz, Austria, +43 316 873 5078, mendez@icg.tugraz.at

Author Name: Gerhard Schall

Bio: Gerhard Schall is a research engineer and PhD student at the Graz University of Technology. He received a Dipl.-Ing. in Telematics from Graz University of Technology in 2002. His current research interests are on mobile augmented reality, ubiquitous and pervasive computing.

Contact Info: Technische Universitaet Graz, Institute for Computer Graphics and Vision, Inffeldgasse 16a/308, A-8010, Graz, Austria, +43 316 873 5079, schall@icg.tugraz.at

Author Name: Sven Havemann

Bio: Sven Havemann works as assistant professor at Graz Technical University in Graz, Austria. He graduated in computer science from the university of Bonn (Germany) and holds a PhD in computer science from Braunschweig Technical University (Germany). His major research interest is to develop novel shape representations for describing the shape of man-made 3D objects by pursuing the idea of Generative Modeling.

Contact Info: Technische Universitaet Graz, Institute for Computer Graphics and Knowledge Visualization, Inffeldgasse 16c, A-8010, Graz, Austria, +43 316 873 5403, s.havemann@cgv.tugraz.at

Author Name: Sebastian Junghanns

Bio: Sebastian Junghanns is a software engineer at GRINTEC GmbH. He also pursues a PhD at Graz University of Technology as an external researcher. His special interests lie in the field of mobile and web-based geospatial systems with application to the domain of utility asset management. Junghanns holds a master's degree in geoinformatics from Salzburg University.

Contact Info: GRINTEC GmbH, Anzengrubergasse 6, 8010 Graz, Austria, +43 316 383 706-32, sebastian.junghanns@grintec.com

Author Name: Dieter Fellner

Bio: Dieter Fellner is a full professor of computer science at Darmstadt Technical University (Germany) as well as at Graz Technical University (Austria). He is also the leader

of Fraunhofer IGD in Darmstadt. Dieter Fellner's research comprises a broad spectrum of areas from formal languages, telematics services, and user interface design, to software engineering, computer graphics and digital libraries.

Contact Info: Fraunhoferstraße 5 64283 Darmstadt, Germany, +49 6151 155-100, d.fellner@cgv.tugraz.at

Author Name: Dieter Schmalstieg

Bio: Dieter Schmalstieg is full professor of Virtual Reality and Computer Graphics at Graz University of Technology, where he directs the "Studierstube" research project. His current research interests are augmented reality, virtual reality, distributed graphics, 3D user interfaces, and ubiquitous computing. He received Dipl.-Ing. (1993), Dr. techn. (1997) and Habilitation (2001) degrees from Vienna University of Technology. He is a member of the editorial advisory board member of computers & graphics, member of the steering committee of the IEEE ISMAR, chair of the EUROGRAPHICS working group on Virtual Environments, and advisor of the K-Plus Competence Center for Virtual Reality and Visualization in Vienna.

Contact Info: Technische Universitaet Graz, Institute for Computer Graphics and Vision, Inffeldgasse 16a/307, A-8010, Graz, Austria, +43 316 873 5070, schmalstieg@icg.tugraz.at