TOWARDS SEMANTIC INTERACTION IN HIGH-DETAIL REALTIME TERRAIN AND CITY VISUALIZATION

Roland Wahl **Reinhard Klein**

AG Computer Graphik
Institut für Informatik II – Universität Bonn
{wahl,rk}@cs.uni-bonn.de

KEY WORDS: DTM, DSM, city model, LOD, multiscale methods, realtime visualization, semantic interaction, ontology

ABSTRACT:

On the one hand, the extent, modeled detail and accuracy of virtual landscapes, cities and geospecific content is rapidly increasing. Realtime visualizations based on geometric levels-of-detail (LODs) allow the user to explore such data, but up to now, the methods of interaction are very low-level. On the other hand, we have semantic categories for the objects which are modeled in ontologies. We propose an approach which allows to combine the advantages of both, realtime visualization techniques and semantic hierarchies, in a single application without establishing an explicit link. That way, we can achieve semantic interaction without interfering with the rendering techniques which is crucial for performance on large datasets. Moreover, we are able to exchange geometric and semantic models independently of each other.

1 INTRODUCTION

Recent advances in digitization technology and reconstruction methods have lead to the availability of huge high-resolution 2.5d digital surface models (Hirschmüller, 2005). As sensors also record views at slightly tilted angles, to a certain degree also 3d reconstruction from aerial data is possible. Reconstructions from these aerial data will help to match and integrate data obtained from terrestrial sensors (Früh and Zakhor, 2003), so that in future we will face captured data sets of cities which bear high detail in full 3d, i.e. huge raw point clouds with spatial resolution in the range of single centimeters. These advances in data acquisition and fusion go side by side with progress in realtime rendering methods which become capable of visualizing the ever growing data sets in full detail.

Concepts in our mind tell us that something is a house, a church, a balcony or something else depending on its appearance and our experience. We are used to perceive things with high visual detail and at the same time think about them in the compressed form of semantic categories. For efficient human computer interaction a system must match these abstract concepts of our mind, i.e. we have to close the semantic gap. To this end data corresponding to different semantic entities are labeled accordingly, which results in a semantic model.

Currently, coming along with the increasing detail of the captured data we also observe an increasing demand for semantic models. Developments in 3d GIS lead to domain-specific ontologies which are also rapidly growing in that they add more and finer semantic categories and metadata (Kolbe et al., 2005). Semantic models based on such ontologies represent the underlying geometry in different simplified versions depending on the semantic level of detail (LOD). Details which are not semantically relevant for the model are neglected and at most represented by textures. This way the reconstruction and semantic modeling of parts of the scene that are not required for a specific ontology can be avoided which saves reconstruction time and costs. Therefore, much information contained inherently in the captured data sets is not mapped into the semantic domain and therefore not available in the semantic model. However, these details, although irrelevant from a semantic interaction point of view might still carry important cues. Examples would be the natural cover and topography in front gardens which are important for the rating of real estates, whereas on the semantic level cadastral data and building data would suffice.

Unfortunately, current terrain and city visualization systems that allow for 3d semantic interaction build on the geometry of the corresponding semantic models only and omit the additional information contained in the captured data. To achieve interactive performance, only parts of the terrain or city-models are rendered and in addition only coarse representations like extruded ground polygons on LOD1 or extruded ground polygons with roof structures on LOD2 are used (Gröger et al., 2004). Due to the rising amount of data higher LODs like LOD3 and LOD4 would require additional multi-resolution techniques to achieve interactive performance and are therefore currently used only selectively. Combining these semantic LODs with view-dependent geometric LODs is a non-trivial problem that is currently not solved, since the geometric and semantic hierarchy must be intertwined. While from a rendering point of view representing planar facades of several neighboring buildings as a single polygon with texture is appropriate, the semantic model requires the geometric representation to respect the borders of the houses.

If semantics and geometry were not kept separate, one possible way to handle this situation would be to choose a representation based on the semantic category. The problems with this approach are firstly, that the optimal representation is not necessarily consistent throughout a semantic class and secondly, that we have to decide for every element of the raw data which category it belongs to. Also semantic categories often overlap, are ambiguous or decompose geometric entities into non-trivial subparts.

Therefore, we suggest to use different geometric representations for the semantic and geometric hierarchies of terrain and city models. The geometric representation, in the following called *rendering model*, which is actually visualized is based directly on the captured data. The representation of the semantic entities, in the following called *interaction model*, which is only used for interaction purposes is built on reconstructed and modeled data, like the above mentioned semantic LODs. These separate representations of geometric and semantic data are joined on-the-fly in interactive rendering systems. This approach enables us to implement semantically based interaction within high-resolution virtual worlds. Furthermore, it allows to combine interaction models for different ontologies with the same rendering model. Even on-the-fly exchange of ontologies can be achieved by loading different interaction models. This is especially useful as the ontology, the semantic LOD as well as the spatial extent of the semantic model can be selected dependent on the specific task. In addition, geometry data and semantic information are usually obtained and created by very different and separate processes as well as different people. Therefore, separate representations fit naturally in the corresponding graphics and GIS workflows and modeling of geometry as well as semantic information becomes substantially easier as none of them has to consider the intricacies hidden in the other, separate system. Last but not least, separate representations allow independent modification of either geometry or semantics at any time. This is especially of interest for update purposes where either the interaction model is further refined or the rendering model is updated, e.g. by adding new data.

In the following, we first concentrate on the rendering aspect. We discuss the problems arising during interactive visualization of high-definition terrain and city models, as well as the consequences for the rendering model in the next section. Then we discuss the interaction aspect in section 3 which describes several computer graphical methods that can be employed to connect the semantic data with the rendering model. This in turn leads to the definition of the interaction model. Some results are presented in section 4 before we come to a brief conclusion in section 5.

2 REALTIME TERRAIN RENDERING

2.1 High detail terrain and city models

For the purpose of photorealistic rendering we need a model which captures as much of the photometric and geometric details as could be perceived in the real world. Obviously, the perceivable detail depends on where the camera is placed in the virtual world. As long as the application shows the terrain from high altitude aerial views, an orthotextured digital terrain model (DTM) is appropriate. Closer to the ground, we need a digital surface model (DSM) in order to perceive the correct parallax and occlusion. Even high-detail textures mapped on a DTM will spoil the realism, due to the contradicting depth-cues (experience tells you that roofs are above ground, but the identical parallax suggests that the roof is at the same height as its surroundings).* For terrestrial or almost terrestrial views the 2.5d modeling approach suffers from systematic problems as facades and other steep parts of the geometry are not represented in orthotextures. Moreover, relevant geometry may be occluded from above. Thus, simply increasing the resolution of the 2.5d model will not suffice. Instead, we have to switch to a 3d model to be able to represent the scene with high precision from terrestrial perspectives as well.

2.2 Insufficiencies of terrain rendering

Multiresolution algorithms for fast rendering of large terrain data sets with viewpoint adaptive resolution have been an active area of research in the field of computer graphics for many years. Since giving a complete overview is beyond the scope of this paper, we refer to the surveys (Lindstrom et al., 1996, Pajarola, 2002).

Although there is a wide variety of methods regarding the details, the basic principles are always:

1. Choose a reasonable granularity for LODs. View-dependent simplification of individual primitives on-the-fly is more expensive than rendering a larger number of primitives. Batches containing a part of the model at the same LOD can be rendered more efficiently.
2. Use these batches for the choice of parts of the model which are visible for the camera (view-frustum culling) and the choice of required LOD.
3. As we deal with out-of-core datasets (i.e. data of such size that it breaks the bounds of physical memory of a computer) we need to implement a preprocessing stage where the data is processed into a hierarchy of batches, which can then be loaded on demand.
4. Choose effective compression algorithms for the data that lessen storage and bandwidth requirements for the model without introducing performance penalties during decompression.

We base our terrain and city rendering algorithm on the terrain rendering system presented by (Wahl et al., 2004) which is based on a quadtree data structure (see Fig. 1). The system has proven to be able to visualize very large terrain data sets efficiently and with high quality, e.g. data sets with a resolution up to a few centimeters for the aerial photography together with elevation models of about 1m, covering areas of hundreds of square kilometers, have already been visualized with realtime frame rates.

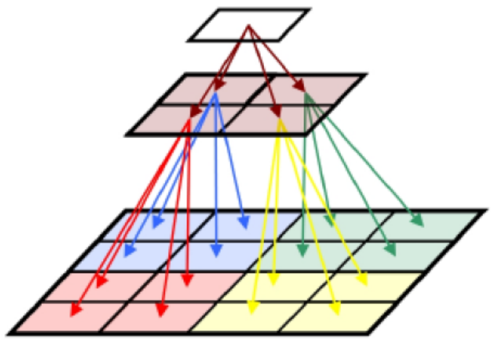

Figure 1: Quadtree layout of terrain rendering. Each part of the model belonging to a quadtree cell (called tile) has the same raster size independent of LOD.

In order to evaluate the different character of landscapes and cities over the scales, let us compare the number of primitives needed for a tile. In a city we may have about 20 buildings per block of size 100mx100m which is 2,000 buildings per km^2. If a tile of a square kilometer should be pixel correct we need ∼8m per pixel accuracy (1km per 128pixels) which means that every building needs to be present in the data. Even if each building is modeled by a hemicube with 5 faces only this leads to 20,000 triangles/tile just due to the buildings itself, not even accounting for the triangulation overhead at the floor and other features, like trees.

Let us now take a closer look at the root of the problem: In the presence of buildings and trees, the complexity in small tiles is comparable to that in alpine regions (∼1,000),† but as opposed to the latter reducing the approximation accuracy does not lead to a smooth diminution of complexity. Instead, the complexity is rising to a maximum and finally breaks down when building

*This effect becomes even more striking if seen on stereo displays, where it is noticeable even in still images.

†As measured by (Wahl et al., 2004) using Hausdorff-distance based mesh approximations.

heights fall below the necessary accuracy. As due to the LOD scheme, the screen-size of a tile remains constant, this means we get more than 1 triangle/pixel, which indicates clearly that such a mesh is the wrong representation of the data.

This example demonstrates that the assumptions stating that the complexity density of representation is roughly independent of scale, which was made for terrain models, do not hold for city data and the scalability of classical terrain models is violated.

Moreover, in 2.5d models the representation of steep geometry most prominently at facades is inappropriate. That is even if terrain visualization did work, it would need to adapt to the incorporation of 3d data.

In summary we can state that there is already a fundamental difference between rendering huge terrain data sets which are modeled as DTM and rendering even comparatively small DSMs. Given a data accuracy and density in the range below meters yields a lot of complexity in otherwise harmless (i.e. flat) data sets. The reason for this is the disproportionate distribution of features to scales. On a 20m scale only hills, rivers, shores and mountains dominate the complexity, whereas on a single meter DSM even in flat regions every tree, bush, car, building etc. leaves its high-frequency fingerprint. As a result of such details a visualization of a single city on a meter scale can be more demanding regarding the level-of-detail scheme than visualizing planet earth on a 20–30 meter basis. Moreover, the representation of the surface as 2.5d is less appropriate for high detail scales than it was for classical terrain models.

2.3 The rendering model

Nevertheless, on coarser levels the geometry still keeps its DTM characteristics and therefore, current highly optimized terrain rendering techniques remain the first choice. In addition, most of the earth's surface is not yet modeled other than with a DSM. The question therefore is how to represent the detailed geometry. We want to decompose the data set into a terrain model and highly detailed 3d geometry like buildings and natural cover which are added in the visualization. This decomposition does not need to respect any kind of semantics. That is, for mere visualization performance we will not necessarily distinguish between what belongs to terrain and what not, but we want to automatically generate representations which consist of hybrid mesh, points or volume data, or any other representation which is applicable.

One way to represent such non-2.5d details is to represent them as point clouds. The rationale behind modeling with points is simple. Point clouds can be trivially extracted from any other boundary representation by surface sampling. Laser scanners, other range finders and stereo reconstruction generate points natively and the raw point cloud thus contains all details present in the data without any interpretation. An additional advantage of point clouds is that they easily represent high-frequency features. A small bump on a plane is modeled using a small pyramid in case of a triangle mesh. This however takes 4 points and 8 additional triangles (3 for the pyramid and 5 to maintain the mesh topology in the plane), whereas with points we could use a single point. Therefore point clouds have become an important means of modeling and as a consequence a lot of rendering techniques and LOD-schemes for points have been developed (Sainz and Pajarola, 2004).

Despite the apparent advantages of the point cloud representation, it leads to problems due to the high depth complexity of the city model (i.e. a high number of intersections of a half ray emanating from the view point with the model) if viewed at acute angles. In this case the number of point rendering primitives remains high although the projected screen size of the model is small, i.e. multiple points are drawn into the same pixel. In such a situation billboards, where the geometry is approximated by textured planes are advantageous (Décoret et al., 2003, Meseth and Klein, 2004). We therefore build our rendering model on a combination of point rendering with billboards as described in (Wahl et al., 2005).

3 INTERACTIVE VISUALIZATION

In this section we will discuss how semantic information can be integrated into a visualization. Of course, the mere fact whether semantic information is available or not does not affect the visual appearance. So, the degree of semantic enrichment of a model is measured by how the user is able to interact with the model. We want to achieve an interactive application which gives the user the impression that the machine has a similar concept of the objects which are in the scene.

Apart from standard interaction features:

- the application reacts on user input through navigation
- the application renders metainformation of selected objects onto the screen

we want to be able to perform *semantic* interaction:

- the application activates objects or visualizes metadata it has linked with objects on demand (picking)
- the application emphasizes objects in the visualization based on semantic information (highlighting)
- semantics-based navigation
- the application is able to temporarily delete objects from the scene, such that the camera looks through them
- the application synthesizes objects on demand

For semantic modes of interaction, the application needs a concept to translate user queries from a semantic level to a geometric level (used for rendering images) and vice versa.

3.1 Implicit modeling of semantics

As we do not want to interfere with the rendering model, we cannot store semantic information directly with the original geometry. We store the missing link between the 3d rendering model and the semantic categories and metadata in the interaction model. The interaction model is linked with the rendering model implicitly by means of its spatial reference. Although this is a common technique in the 2d case (e.g. mapping thematic layers on orthophotos or topographic maps), the extension to 3d is not trivial. In the 2d case, the mapping is easily achieved by reprojecting the data into the target's coordinate system and highlighting or picking can be performed by using image overlays. In the 3d case, a 2d overlay will not suffice, since it does not discriminate along the vertical axis (e.g. windows on different floors). However, generating a full 3d overlay is no option, as it requires exactly the semantic 3d model we wanted to get rid of. Our solution to this dilemma is to use proxy objects in the interaction model. For example, in the semantic model a house can be modeled by a cube although it is a detailed point cloud in the rendering model. The link is established by performing a 3d (i.e. volume) intersection between the proxy geometry and the rendering model.

This approach effectively implements the relation between data objects and semantic entities and has the following advantages:

- There is no explicit link between the rendering model and the semantic model.
- There is no necessity to deal with the different LODs of the rendering model.
- The rendering model can be changed or replaced with no influence on the interaction capabilities.
- The interaction model can be updated dynamically without affecting the rendering, i.e. interaction models for different ontologies can be loaded on-the-fly.
- Direct compatibility with traditional 2d or 2.5d GIS data. Proxy geometry for such data can be trivially generated on-the-fly by extrusion.
- The intersection test with the proxy geometry can be evaluated on the graphics hardware (see section 3.3) which is very fast and easy to implement.
- Output sensitivity. The overhead is solely determined by the query complexity irrespective of the complexity of the rendered model. Especially, this means that without any semantic interaction we have no additional costs at all.

However, there are also some limitations:

- The rendering model must use accurate 3d coordinates, so that the volume intersection tests leads to the expected result. This constraint is obviously fulfilled by most geometry representations (triangles, points, voxels) but it does not hold for image based approaches.
- Skipping parts of the model based on semantic information (see section 3.4) is substantially more difficult to implement than with explicit links.
- As the user can interchange models, depending on the accuracy of these models and temporal changes in the meantime they can be inconsistent. This is unlikely for models derived from the same raw data sets and therfore rather a side-effect of interchanging models as a real disadvantage.

Alternatively to the implicit linking via volume intersection, we could also explicitly store the inverse relation between geometry and semantic entities. Instead of telling the geometry which semantic object it belongs to one would establish a mapping of semantic objects to geometric primitives. If this mapping is implemented with the ability to address sub-parts of a geometry (e.g. the part of a triangle belonging to a window), the model could be employed without constrainingthe simplification. The task during semantic interaction then consists of identifying the object by querying the inverse relation which is admittedly less efficient. Although the direct influence of the semantics on the preparation of the model is remedied, there is still a close explicit link between semantic and rendering model: In order to address the geometry of an object we need detailed information on how the model is represented. Moreover, a representation which lends itself to efficient rendering might still be very complex when we have to address parts for example in point clouds or billboards.

3.2 Interaction model

As mentioned earlier, the interaction model represents the semantic hierarchy and therefore its underlying ontology within the visualization application. It is composed of the actual semantic information i.e. categories and metadata as well as the proxy geometry. Apart from the difference in the geometric representation any type of semantic model can be directly used as interaction model, e.g. CityGML (Kolbe et al., 2005). Therefore, we will not discuss the semantic features here, but focus on the requirements for the proxy geometry.

Regarding the representation, the proxy geometry should consist of well-defined solids, ideally in form of an oriented boundary representation as these can be used directly for rendering. Apart from that there are three observations which influence the design of such proxies:

1. Interaction may take place on a coarser level-of-detail than visualization. Inaccuracies in the range of pixels would compromise the visual quality, but human interface devices are rarely placed with pixel accuracy.
2. Interaction models remain hidden from the user. As highlighting is not implemented as a 2d overlay, we can change the appearance of the model by evaluating the 3d proximity to the semantic model, thus the proxy geometry carrying the semantic information is an invisible layer.
3. Picking results are predictable even with coarse interaction models. The predictability of the picking is owed to the accuracy and simplicity of the proxies. Buildings and related objects in cities can mostly be well described by simple features like a small number of boxes or polyhedra. Moreover, an immediate feedback to the user can be used to verify whether the active object in the interaction model matches the intention of the user.

As a consequence, we do not need to model the proxies with the same detail as a representation intended for rendering. Actually, we just need to ensure that the object under consideration is within the volume and no other object intersects the volume. As we are dealing with models of bounded accuracy, this implies that we should inflate the proxy solids a bit in order to avoid that parts of the rendering model protrude the volume.

For performnce reasons, the proxy geometries should also be organized in a spatial hierarchy like the quadtree mentioned earlier. This will improve performance when a large number of highlighted objects is not within view and can therefore be skipped. Activating only parts of the interaction model, based on the specific application will further accelerate interaction in the case of very complex models.

3.3 Implementation issues

The implementation of semantic interaction based on implicit mapping using proxy geometry is straight forward. For highlighting, however, there is an optimized implementation which exploits graphics hardware.

The highlighting works as follows:

1. Render the whole scene. This sets the colors of the framebuffer (the buffer which is displayed on the screen) but also the corresponding z-buffer (an off-screen buffer which records the distance of each drawn pixel to the camera, used for correct occlusion).

2. Render the active object into the stencil buffer (a special-purpose auxiliary buffer which can be efficiently queried and updated during rendering):

   a) Disable depth-buffer and frame-buffer writes. Set stencil operation to increase stencil for pixels less distant than the z-buffer (z-pass) and render back-facing polygons of the proxy.

   b) Set stencil operation to decrease stencil on z-pass and render front-facing polygons of the proxy.

3. Now the stencil is positive in all pixels which have depth values within the proxy geometry. Activate frame-buffer write, blending for transparency and stencil test. Render screen-sized quad with the highlight color.

As only the colors are changed, it is clear that this process can be applied to any number of objects simply by iterating phases 2 and 3. In fact it suffices to simultaneously perform step 2 for all objects with the same highlight color.

Picking can either be implemented by using the highlighting feature to render unique IDs into an auxiliary buffer, which then tells which objects cover which part of the screen or by performing a z-readback (transferal of the depth-buffer from graphics hardware to main memory) which yields the 3d point at the given screen coordinates. This point can then be transformed into a spatial query to the semantic model.

3.4 Advanced interaction

While the picking and highlighting features are easy to implement because they only rely on information gathered in the hardware buffers of the graphics processing unit (GPU), tasks such as looking through (i.e. deleting) objects are more intricate. However, in many scenarios of future 3d-GIS applications such features may be especially worthwhile. Although omitting objects from rendering is a lot easier when we have direct access to the model, there are still possibilities to achieve the same result without touching the rendering model.

The problem with the transparency feature is that we need to know what would have been seen if the occluding (deleted) object was not there. Of course, we can apply our highlighting framework to determine which fragments [‡] of the image are the false occluders, but as our output consists only of 2d (color and depth) buffers there is no way to access information behind these. The obvious way to circumvent this problem is to modify the rendering such that only fragments not within the proximity of deleted objects are rendered. This can be achieved using the hardware-support on modern GPUs by either performing clipping of the rendering primitives in the geometry shader or by discarding fragments in the fragment shader based on an implicit representation of the proxy geometry. Clipping can be very efficient for convex polytopes as the in/out status can be established using a few halfspace inclusion tests. The disadvantages of this approach are that it relies on computing capabilities only available on very recent hardware, the computation needs to be done every frame and the number of halfspace intersections will introduce a performance penalty, if many or complex objects are to be hidden. Preferably, the fragments should be discarded based on a single lookup. The stencil lookup however is insufficient, as it has no depth value. We can render the hidden proxies into an auxiliary z-buffer, but there we could only store one depth per fragment. In a 3d scene, however a high depth complexity (many primitives project to the same pixel) can be present especially when rendering terrain from almost terrestrial views. Therefore we would need to compute a list of depth intervals for each pixel which correspond to the different parts of the hidden proxies. Generating such lists is not feasible on current hardware. Therefore we propose to use a technique similar to shadow mapping (William, 1978). We make use of the fact, that city models still have a 2.5d character. Therefore, the complexity in vertical direction will generally be bounded by a small constant. So instead of storing the visibility information in the screen domain we will use a map-projection and store height values instead. The remaining z-complexity of the proxies' volume can be accounted for by storing multiple upper and lower contours. This approach has the additional advantage, that the visibility information remains the same for many frames and does not have to be recomputed. Recomputation is only necessary when and where objects change their invisibility status. In order to exploit this locality we will bind these auxiliary invisibility maps to the tiles of the geometric LOD-hierarchy.

[‡] A fragment denotes the data necessary to generate a pixel in the frame buffer. This includes raster position, depth, color, stencil, etc.

Technically, the algorithm for hiding objects works in two parts:

Initialization of new invisible objects (if applicable):

- identify tiles of new objects
- initialize contour buffers
- set map projection
- render front/back faces in upper/lower contour buffer respectively

Scene rendering (every frame):

- reproject each fragment into map coordinates (gives height above ground)
- compare height to contour heights
- if height is within an upper/lower contour pair kill fragment

Note that again the additional complexity correlates with the query complexity (complexity of invisible objects) and does not depend on the scene, such that a true output-sensitivity is maintained.

4 RESULTS

We implemented the proposed scheme for semantic interaction within a realtime 2.5d terrain visualization framework to demonstrate the applicability. The rendering model is a $28cm$ resolution DSM colored with orthophotos of $7cm$ resolution. This model is courtesy of German Aerospace Center (DLR) – Institute of Robotics and Mechatronics and was derived by semi-global matching (Hirschmüller, 2005) from Vexcel Ultracam™ imagery. The semantic model was derived from "CityGML reference data on Pariser Platz" freely available for download[§]. The boundary representation of the solids modeling the semantic entities was used as the above mentioned proxy geometry.

Figure 2 shows a rendering of a part of Berlin with the highlighting feature. The extension of Hotel Adlon was selected by the user and is thus highlighted. In figure 3 the picking feature is demonstrated: As the mouse hovers over the image during realtime exploration the semi-transparent highlighting provides an

[§] `http://www.3d-stadtmodell-berlin.de/`

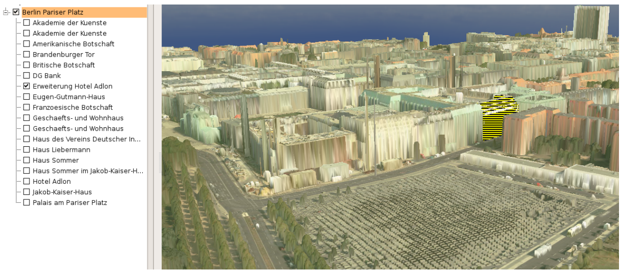

Figure 2: Screenshot from the rendering application showing a highlighted** feature of the semantic dataset within a visualization of the Berlin DSM.

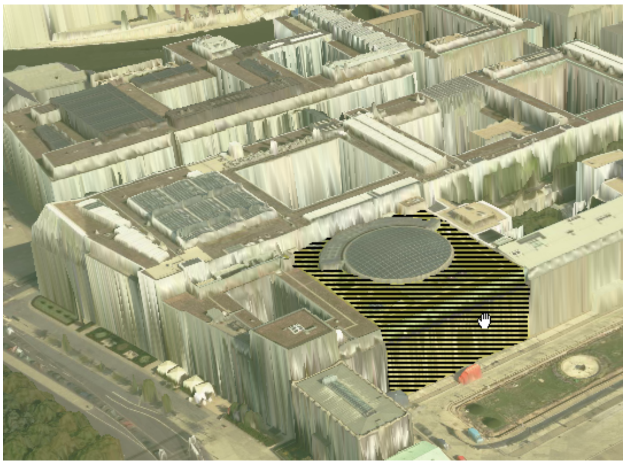

Figure 3: Screenshot from the rendering application. In picking mode semantic entities are highlighted** as the mouse hovers over them in the image. The user thus gets an instant feedback with which objects and on which semantic LOD he is about to interact.

instant feedback which objects are present in the active semantic category. With these objects we can interact by accessing meta-information or highlighting them as seen above. The glass roof of this building (Eugen-Gutmann Haus) is not highlighted because it is not within the volume defined by the semantic model which demonstrates that this mode of interaction is truly 3d.

5 CONCLUSION

In this work we introduced methods to allow semantic based interaction in highly detailed 3d terrain- and city models. In the proposed approach different ontologies can be used without changing the geometric LOD-hierarchy used for photorealistic rendering. In our opinion, the ability to switch the ontologies and thus the categories in which we think about the data is more important than the exact correspondence between geometric features and semantic entities. We have presented methods which allow interaction with photorealistic visualizations, so that the missing explicit link becomes unnoticable to the user.

5.1 Future Work

The ability to access detailed semantic information within a realtime photorealistic rendering framework is just a first step towards semantic interaction. With the combined strength of detailed semantic models and visually detailed instances there is a lot of place for new interaction paradigms especially concerning the modeling or synthetization of such scenes but also the exploration and visualization techniques.

Another direction of research will concern the flexibility of models. By now, all successful realtime rendering methods have to preprocess the data. Apart from the delay which is introduced by doing so, there is a conceptual difference whether the data needs to be static or it can be adapted on-the-fly.

**The black stripes were added to illustrate highlighting in the b/w hardcopy of the proceedings.

ACKNOWLEDGMENTS

This work was funded by the German Science Foundation (DFG) as part of the bundle project "Abstraction of Geographic Information within the Multi-Scale Acquisition, Administration, Analysis and Visualization". We thank the Berlin Senate and Berlin Partner GmbH for making the CityGML model available. Finally, we also want to thank the reviewers for helpful comments.

REFERENCES

Décoret, X., Durand, F., Sillion, F. and Dorsey, J., 2003. Billboard clouds for extreme model simplification. In: Proceedings of the ACM Siggraph, ACM Press.

Früh, C. and Zakhor, A., 2003. Constructing 3d city models by merging aerial and ground views. IEEE Computer Graphics & Applications 23(6), pp. 52–61.

Gröger, G., Kolbe, T. H., Drees, R., Kohlhaas, A., Müller, H., Knospe, F., Gruber, U. and Krause, U., 2004. Das interoperable 3d-stadtmodell der SIG 3d der GDI NRW. `http://www.ikg.uni-bonn.de/fileadmin/sig3d/pdf/Handout_04_05_10.pdf` (15.07.2007).

Hirschmüller, H., 2005. Accurate and efficient stereo processing by semi-global matching and mutual information. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, pp. 807–814.

Kolbe, T. H., Gröger, G. and Plümer, L., 2005. CityGML – interoperable access to 3d city models. In: P. van Oosterom, S. Zlatanova and E. M. Fendel (eds), Proc. of the 1st International Symposium on Geo-information for Disaster Management.

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L., Faust, N. and Turner, G., 1996. Real-time continuous level of detail rendering of height fields. Proceedings of SIGGRAPH'96 pp. 109–118.

Meseth, J. and Klein, R., 2004. Memory efficient billboard clouds for BTF textured objects. In: B. Girod, M. Magnor and H.-P. Seidel (eds), Vision, Modeling, and Visualization 2004, Akademische Verlagsgesellschaft Aka GmbH, Berlin, pp. 167–174.

Pajarola, R., 2002. Overview of quadtree-based terrain triangulation and visualization. Technical Report UCI-ICS-02-01, Information & Computer Science, University of California Irvine.

Sainz, M. and Pajarola, R., 2004. Point-based rendering techniques. Computers & Graphics 28(6), pp. 869–879.

Wahl, R., Guthe, M. and Klein, R., 2005. Identifying planes in point-clouds for efficient hybrid rendering. In: The 13th Pacific Conference on Computer Graphics and Applications.

Wahl, R., Massing, M., Degener, P., Guthe, M. and Klein, R., 2004. Scalable compression and rendering of textured terrain data. In: WSCG, pp. 521–528.

William, L., 1978. Casting curved shadows on curved surfaces. Computer Graphics (Proceedings of SIGGRAPH '78) pp. 270–274.