Automated Construction of Urban Terrain Models

Henrik Buchholz[1], Jürgen Döllner[1], Lutz Ross[2], Birgit Kleinschmit[2]

[1]University of Potsdam and [2]Technical University of Berlin

Abstract

Elements of urban terrain models such as streets, pavements, lawns, walls, and fences are fundamental for effective recognition and convincing appearance of virtual 3D cities and virtual 3D landscapes. These elements complement important other components such as 3D building models and 3D vegetation models. This paper introduces an object-oriented, rule-based and heuristic-based approach for modeling detailed virtual 3D terrains in an automated way. Terrain models are derived from 2D vector-based plans based on generation rules, which can be controlled by attributes assigned to 2D vector elements. The individual parts of the resulting urban terrain models are represented as "first-class" objects. These objects remain linked to the underlying 2D vector-based plan elements and, therefore, preserve data semantics and associated thematic information. With urban terrain models, we can achieve high-quality photorealistic 3D geovirtual environments and support interactive creation and manipulation. The automated construction represents a systematic solution for the bi-directional linkage of 2D plans and 3D geovirtual environments and overcomes cost-intensive CAD-based construction processes. The approach both simplifies the geometric construction of detailed urban terrain models and provides a seamless integration into traditional GIS-based workflows. The resulting 3D geovirtual environments are well suited for a variety of applications including urban and open-space planning, information systems for tourism and marketing, and navigation systems. As a case study, we demonstrate our approach applied to an urban development area of downtown Potsdam, Germany.

1. Introduction

Photorealistic virtual 3D city models and virtual 3D landscape models form a basis for an increasing number of applications and systems. They can be used, for example, in landscape and open-space planning to present

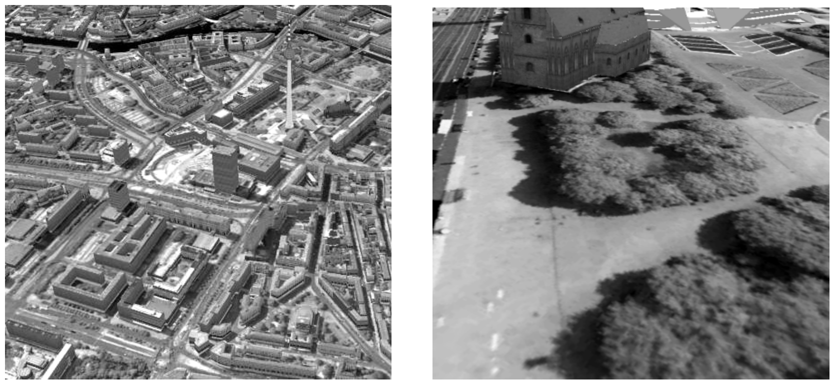

Fig. 1. Aerial images are well suited for bird's eye views (left) but fail for ground-level views as illustrated by the flattened trees (right).

planning scenarios to the public (e.g., Danahy 2005, Lange and Hehl-Lange 2005, Stock and Bishop 2005, Warren-Kretzschmar and Tiedtke 2005, Werner et al. 2005) or in tourism to allow visitors exploring a city virtually. Many geovirtual environments such as the example in Figure 1 (left) are based on a set of 3D building models and 2.5D terrain model draped by aerial images to represent areas that are not covered by buildings. These areas consist of manmade surface structures (e.g., roads, pavement, walls, stairs, or squares), natural terrain surfaces, which are often covered by vegetation (e.g., woodland, agricultural land, grassland, or rocks) and water surfaces (e.g., rivers, canals, or lakes). Aerial images are well suited for representing such surface cover information from a bird's eye view but do not provide sufficient detail for visualization from a pedestrian's point-of-view (Figure 1 right). In addition, the represented terrain elements cannot be analytically identified and modified because there is no concise object-oriented representation of these elements and the linkage between GIS planning data and virtual 3D model has been lost. This makes aerial images unsuitable for land use related planning tasks. Therefore, many applications require more detailed virtual 3D terrain models (Figure 2).

There are three common approaches for creating detailed terrain surface representations: *manual modeling, triangulated irregular network* (TIN) *models,* and *image draping*.

Manual Modeling

Using standard 3D modeling tools or CAD tools for manual modeling of terrain surface structures provides maximum flexibility. It involves, however, high manual efforts and requires a high degree of expertise. In addi-

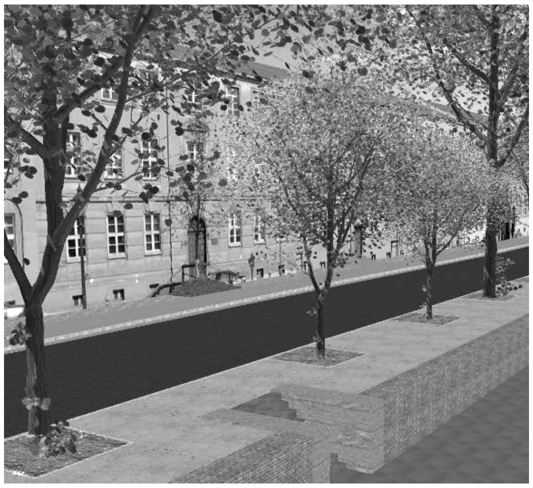

Fig. 2. Snapshot of the automatically created urban terrain model of our case study, a redevelopment of a downtown area in Potsdam.

tion, the result consists of computer graphics 3D models that are detached from any underlying 2D geo-data that have been originally used as a basis for modeling. The following limitations result:

- No object-specific modeling techniques: For most elements of urban terrain models, editing techniques could take advantage of object-specific construction rules and constraints (e.g., distribution, form, or height of stairs).
- No automated updating: The computer graphics 3D models cannot be reused if the 2D geo-data changes.
- No geo-data context: The computer graphics 3D models have to be created separately from related geo-data that could be helpful for editing. For instance, showing aerial images or topographic maps can be useful to validate a 3D model visually while editing. Since generic 3D editing tools are not aware of the geo-spatial context of the data, they do not provide such functionality.
- No geo-data linkage: The computer graphics 3D models do not preserve data semantics and thematic information contained in or associated with 2D plans.

TIN-based Modeling

TIN models can also be used to represent detailed terrain models. They allow for integrating land use information and built surface structures by using line or polygon features as break lines and by assigning different colors or textures to certain triangles. This way, detailed terrain models can be created from GIS input data. The TIN-based approach, however, has two major disadvantages:

- No linkage to 2D plan elements: No direct link is established between a TIN and the original data from which it has been created. Therefore, thematic and semantic information is lost and changing the original data requires rebuilding the TIN.
- Restricted geometry: As an inherent limitation, vertical faces cannot be modeled by TINs.

Modeling based on Image Draping

The most common approach to create detailed urban terrain models in landscape visualization is based on the principles of image draping. Specialized modeling tools for virtual landscapes such as World Construction Set, ArcScene, or VirtualGIS allow for integrating GIS data into the 3D scene by draping vector-based information onto the terrain and assigning colors or textures to the polygons. Such tools reduce the manual modeling effort and are able to create satisfying results at landscape scale (Appleton et al. 2002). In landscape planning, they are primarily used to visualize terrain surfaces carrying vegetation by textures (Muhar 2001) and prototype 3D plant models. In addition, the tools mentioned above are suitable to visualize roads or other manmade terrain surfaces. However, in order to integrate manmade surface structures that include textured vertical faces or that have to be created from polylines by buffering a line feature, manual modeling effort and extensive data preparation is required.

Smart Terrain Models

In this paper, we propose *smart terrain models*, an approach for generating high-quality urban terrain models automatically from 2D vector-based plans. Instead of converting available 2D data to new, detached 3D models, our approach is based on the idea to keep the original 2D data and to add complementary information to enhance them to a complete 3D scene specification. The approach is similar to the image draping method but extends it by defining specialized representations for typical terrain elements. This increases the modeling flexibility while providing significant advantages compared to generic 3D models:

- The underlying 2D vector data that form the core of the smart terrain model specification can still be accessed and edited by external 2D GIS tools and can be obtained and updated from 2D data sources.

- Thematic information and data semantics can be directly queried within the 3D visualization.
- Due to the permanent link between 2D data and their 3D representation, interactive visualization of the generated 3D models can be directly combined with editing tools for the underlying 2D data. This way, the effect of modifications of the 2D data can be directly shown in the 3D model. Particularly, it allows for immediate corrections of any errors in the underlying 2D data when they become visible in the 3D visualization.
- Since the data semantics is preserved in the 3D representation, smart terrain models form a basis for the development of visualization tools that are aware of the data semantics and use it to apply specialized real-time rendering techniques (e.g., Finch 2004, Shah et al. 2005) for certain surface elements to optimize the rendering performance or to improve the visual quality.

We demonstrate our approach in a case study, in which we modeled an urban area of downtown Potsdam, Germany. The combination of smart terrain models with 3D building models, e.g., provided by Smart Buildings (Döllner and Buchholz 2005) or CityGML (Kolbe et al. 2005), and 3D vegetation models (Deussen 2003) leads to photorealistic 3D city models that are suitable for real-time visualization at a pedestrian's point-of-view. Our approach has been implemented on the basis of the graphics library VRS (Döllner and Hinrichs 2002) and the LandXplorer geovisualization system (Döllner et al. 2003).

2. Smart Terrain Models

In this section, we describe our system for modeling and editing smart terrain models. Section 2.1 gives an overview of the system and defines the data components that specify the model. Section 2.2 gives a classification of the terrain elements that constitute a smart terrain model. In section 2.3, we describe the appearance specification for terrain elements.

2.1 System Overview

The principal workflow of our approach is illustrated in Fig. 3. The specification of a smart terrain model provides the complete information for

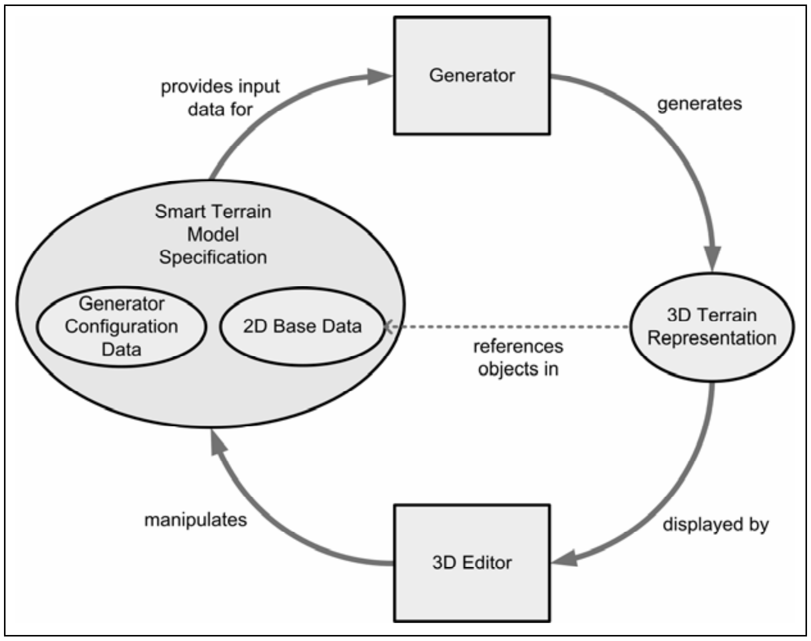

Fig. 3. System overview for the automated construction of urban terrain models.

automatic generation of a 3D model representation and consists of two parts, the *base data* and the *generator configuration*.

The *base data* consist of any 2D vector-based land use data containing polylines and polygons, in the following referred to as *base vector-objects*, with attached attribute tables. Each attribute table defines a set of numerical, textual, or Boolean values that can be accessed via certain attribute-table key strings. The base data can be obtained from existing geo-data bases but can also be edited directly in the 3D-editor, e.g., based on a given aerial image. The base data can be specified in any format that allows for describing 2D vector-data with associated attribute tables, e.g., ESRI shapefiles or GML.

The *generator configuration* specifies the way in which base data and related attributes are interpreted to generate the 3D terrain model. For instance, in a given base-data set polygons representing water areas may be indicated by defining the value "Water" for the attribute-table key "Area Type". The generator configuration also defines one or more material catalogues. A *material catalogue* consists of a set of material descriptions that are referenced by 2D base vector-objects. Materials are discussed in Sec-

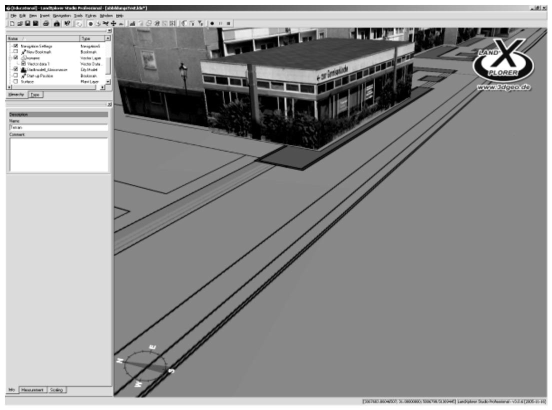
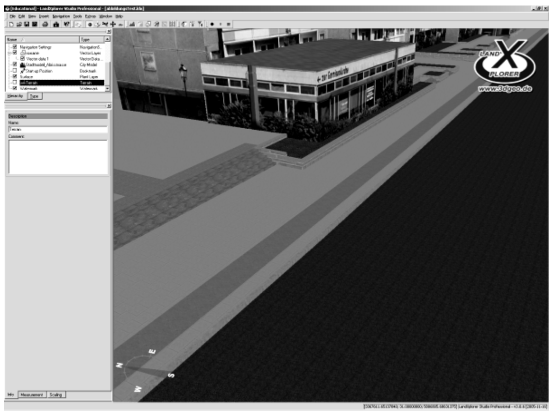

Fig. 4. Snapshot of the editor system: The user can switch between the 2D view of the base data (top) and the resulting 3D representation (bottom).

tion 2.3. The generator configuration and the material catalogues are stored as separate XML files.

The *generator* takes the smart terrain model specification as input and creates a 3D representation of the terrain that is used for real-time rendering. The generator also maintains for each generated 3D object a reference to the underlying base vector-object. This information is used by the editor to support selection and editing of surface-model elements directly in the 3D scene.

The *interactive 3D editor* allows for creation and management of 2D base data, generator configuration, and material catalogues. It provides real-time visualization of both the underlying base data as well as the resulting 3D representation (Fig. 4). The 2D base data can be displayed on the surface of a digital elevation model using the technique described by Kersting and Döllner (2002). If a terrain element is selected and edited by the user, the modification is applied to the underlying base data, and the generator immediately updates the corresponding parts of the 3D representation, so that changes of the base data are directly visible in the 3D environment. The 3D representation itself, however, is never changed by the user but always fully determined by the smart terrain model specification. Hence, editing effort is never lost when the 3D model has to be re-built by the generator.

2.2 Smart Terrain Model Elements

The elements of a smart terrain model are organized in the classes `GroundArea`, `WaterArea`, `Stair`, `Wall`, `Kerb`, and `Barrier` (Fig. 5). Instances of these classes are not explicitly part of the specification but are defined implicitly by the base vector-objects and their related attributes. Most terrain elements correspond to exactly one base vector-object. The only exception holds for irregular stairs, which require multiple polygons for specification. The attributes of a base vector-object determine the class of the corresponding terrain element. Depending on the respective class, the required attributes for 3D shape and appearance of a terrain element are also taken from the attribute table of the base vector-object.

Our primary design goal was to make the refinement of pure 2D base vector-data to a full smart surface-model specification as simple as possible for typical cases. If the class model would provide the unrestricted geometric flexibility of a generic 3D tool, it would not be possible anymore to specify the terrain elements completely via 2D base data, and the editing process would become very complicated. Thus, the main advan-

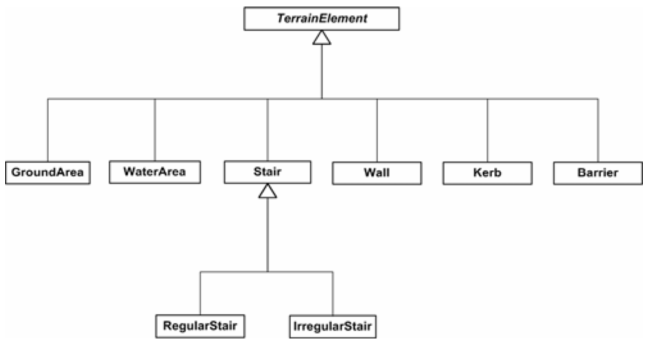

Fig. 5. Overview of the terrain element classes.

tages of the system would be lost. Therefore, we restricted the class model to cases that can be intuitively described via 2D polygons or polylines and support optional refinement of individual objects by external 3D tools. For this, for each terrain element, the automatically generated 3D model can be exported, externally refined, and finally referenced by the terrain element. This way, the effort for fitting an externally created 3D model correctly in the scene is avoided and the 3D model is still related to the underlying base data.

Ground Areas

A `GroundArea` represents a polygonal surface on the ground that is displayed with a certain appearance. For instance, a ground area may represent a part of a street, a sidewalk, a lawn, or a wasteland area. Since the `GroundArea` class covers several kinds of terrain elements, it could theoretically be split up into several subclasses but there are some reasons to use a single class instead:

- To enforce a generic sub-classification of ground areas can be impossible or ambiguous. For example, a single asphalted area might be interpreted as a part of a parking area, a part of a street, or a part of a square.
- From a technical point-of-view, a finer classification is not necessary because the created 3D representations differ only by material.
- From a user's point-of-view, a finer classification is not necessary because thematic classification can be obtained from the base vector-objects' attribute tables, and visually thematic sub-classification can be achieved by assigning different materials.

Each `GroundArea` is defined by a single base-data polygon or a polyline that is buffered to a certain width. If the data format used for the base data supports 2,5D vector data, i.e., if it allows for specifying per-vertex height values for each vector object, as in the case of shapefiles or GML, the surface geometry can be completely defined by the geometry of the underlying base vector-object. If height values are not provided by the initial base data, they can either be automatically derived from a digital elevation model or edited manually. To assign height values by projecting the dataset onto a digital elevation model corresponds to the principle of image draping and is a good solution for continuous surfaces without vertical breaks.

In the general case, however, `GroundAreas` do not necessarily define identical height values at their borders. Therefore, some `GroundAreas` must be rendered with vertical border faces to avoid holes in the terrain model. For this, an optional extrusion depth can be specified, by which the surface is extruded downwards. If desired, a separate material can be specified for the vertical border faces.

Water Areas

A `WaterArea` represents a polygonal surface that appears as a water surface in the 3D visualization. The geometry of a `WaterArea` is specified in the same way as a `GroundArea` but it must always be fully horizontally and allows for specifying a desired flow direction of the water. We integrated water areas as an own class to allow future viewing applications for rendering the water different to solid surfaces, e.g., by animated water textures.

Stairs

In the general case, a `Stair` is described by multiple base-data polygons, whereby each polygon represents a single step and is extruded downwards (`IrregularStair`) to obtain the 3D shape of the step. Many stairs can be described more simply via a single rectangular footprint by moving certain edges of the rectangle inwards to describe the footprints of the upper steps (`RegularStair`, Fig. 6). A `RegularStair` can be specified by a single base-data polygon, whose attribute table defines number of steps, step height, step width, and the information which edges to move inwards. Each stair defines a material for its surface and optionally an additional material for its vertical faces.

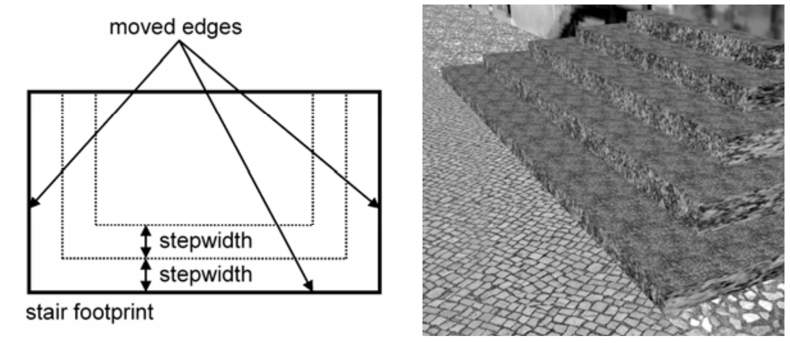

Fig. 6. Construction of a simple stair: The footprints of the upper steps are defined by moving inwards certain edges of the full stair footprint (left). The 3D shape is then defined by extruding each step polygon downwards (right).

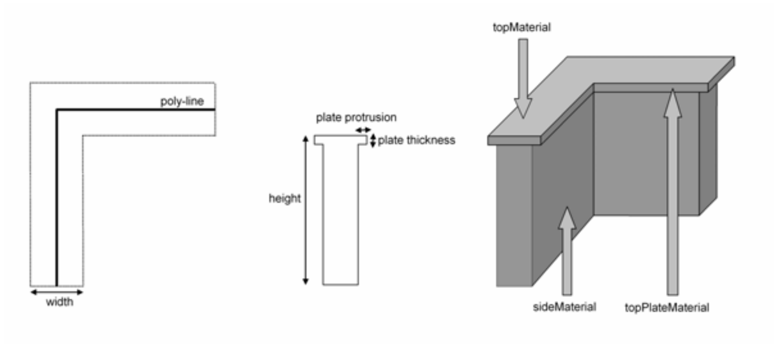

Fig. 7. Construction of walls: a) Specification of the footprint (left); b) 3D extrusion parameters (middle); c) materials for each surface (right).

Walls

The class `Wall` represents freestanding walls and retaining walls. Many walls contain a separate top plate, which might differ from the rest of the wall by another material and by slightly exceeding the wall footprint. Therefore, the `Wall` class provides optional parameters to specify a separate top plate. Fig. 7 shows an example. Walls are represented by polygons or polylines of the base data and their attributes. By buffering line features to a width that is specified by the attribute table, we obtain the footprint of the wall (Fig. 7a). The 3D shape is now determined by the extrusion parameters shown in Fig. 7b, and finally, materials are specified for each side (Fig. 7c).

Kerbs

The class `Kerb` represents boundary objects between two `GroundAreas` with a separately specified appearance, e.g., by an own kind of stone or by a different height (Fig. 8). This corresponds to typical construction methods for, e.g., pavements and roads, where a kerbstone is necessary to stabilize the construction. A `Kerb` is defined by a polyline of the base-data set.

For the frequent case of two `Kerbs` along a single boundary line, both `Kerbs` should appear side by side and not overlapping. For this, each `Kerb` must specify one of the two adjacent `GroundAreas` as the *parent area* to which the `Kerb` belongs. This defines the order in which the `Kerbs` appear between the two adjacent `GroundAreas`. The parent area, i.e., its underlying base vector-object, is referenced in the attribute table of the `Kerb`’s underlying polyline. To allow for referencing, the base-data vector-objects

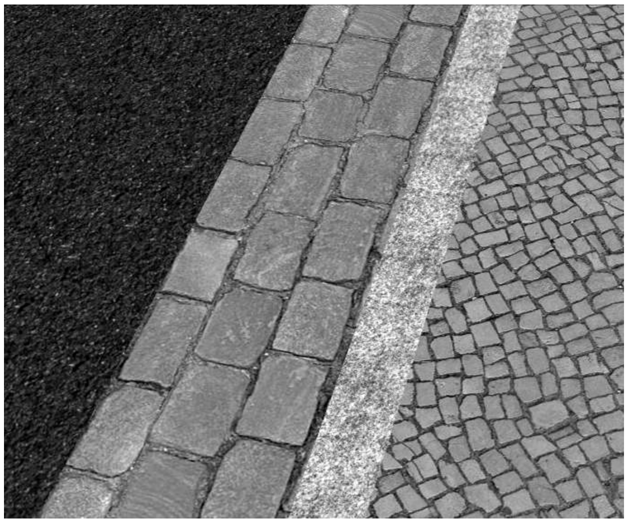

Fig. 8. Examples of kerbs along the border lines of streets and sidewalks.

must provide unique ID values in their attribute tables. In the rare case of three or more parallel boundary objects along a single boundary line, the middle objects must be represented by additional `GroundAreas`.

The footprint of the `Kerb`'s 3D representation is obtained by buffering the `Kerb`'s underlying poly-line to a certain width. Since the Kerb shall appear completely as a part of its parent area, the buffering is performed only in the direction pointing inside the parent area. Finally, the 3D shape is obtained by extruding the footprint to a certain height. Buffering width and extrusion height are specified via the attribute table.

Barriers

The class `Barrier` represents boundary objects such as fences or balustrades. In contrast to walls, barriers do not cover any area on the ground but their footprint is only line-shaped. Each `Barrier` object is defined by a polyline in the base-data set which specifies the height of the `Barrier` in its attribute table. The generator provides a set of standard barrier types. In our current implementation, `Barriers` are simply represented by extruded lines covered with partially transparent texture-images. More advanced future `Barrier` types could include actual 3D detail geometry. If the standard types are not sufficient, the `Barrier` can be refined by an external 3D modeling tool.

2.3 Appearance Properties

The material catalogue provides a set of materials that can be referenced by the base vector-objects via unique names. A material can be of one of two types: Color and texture. A *color material* defines an RGB color value.

A *texture material* defines a reference to a texture image file and scaling parameters that define to which width and height the texture is stretched when it is applied to a surface. Each terrain element that references a texture material must define an anchor point onto which the texture origin is mapped in the 3D model and an orientation angle of the texture. Since these values are usually different even for objects of the same material, they are not stored as a part of the material itself.

3. Case Study

We used our approach in an open-space planning task concerning a part of the German city Potsdam. The aim of our modeling project was to provide a detailed photorealistic 3D geovirtual environment that could be explored interactively. The result can be seen in Fig. 2 and Fig. 4.

The initial data of the modeling project were provided to us by the city of Potsdam and were part of the digital municipal town map. These data originate from ground survey, are geometrically very accurate, and hold detailed information about buildings, surface cover, installations and vegetation. Information about the terrain surface cover is maintained in the data by mapping borderlines between different surface types and placement of cartographic symbols for different surface materials and vegetation areas. Borderlines can either represent a change in surface materials or can represent kerbstones. Buildings are represented by polygons with attached information about the number of floors and additional information about balconies and car passages. Walls and stairs are represented through polylines or polygons depending on their size. Installations and trees are represented through point symbols.

To create a smart terrain model from the data, all features representing a change in surface type or material were used to create an area wide polygon dataset. Thematic information, stored in point features, was then transferred to the polygons through a point in polygon selection. Using the original thematic information all objects were classified into the smart terrain element classes. The resulting dataset contained all polygonal terrain elements that were represented by the town map. In order to integrate walls and kerbstones that were represented by polylines, these were classified and specified as well. In the next step a material table was created and assigned to the features. It holds material names, the names of the texture files and texture scaling parameters. Finally, height information was assigned to the terrain elements. Height information for `GroundAreas` and `Kerbs` were derived by projecting the respective features onto a digital elevation model. For features representing `WaterAreas` and `Stairs` constant values were assigned. Objects representing walls were assigned constant height values or the height was calculated as an offset of the terrain.

The resulting terrain model was combined with models for buildings and plants as can be seen in Fig. 2. For the plants, we used plant models and the plant rendering engine of the project Lenné3D (Paar & Rekittke 2005). For modeling and representation of buildings we used the approach of Döllner and Buchholz (2005). The buildings were automatically generated from 2D GIS input data and refined afterwards, e.g., by textures for roofs and facades. As the footprints of the buildings were used in creating the terrain model, no inconsistencies could appear between buildings and terrain. Furthermore any thematic information that was stored within the attribute table of the input data was preserved.

4. Conclusions and Future Work

The presented approach simplifies and enhances the construction, manipulation, and usage of complex urban terrain models. Its major advantages include the persistent linkage to 2D vector-based plans, the rule-based and heuristic-based automated model generation, and the inherent functionality and smartness of urban terrain objects. Base 2D geo-data can be taken from GIS and integrated seamlessly into the 3D modeling process. The ability of smart terrain models to maintain semantic and thematic information provides a technical basis for smart 3D geo-visualization tools. In addition, the approach represents a step towards 3D geovisualization from a pedestrian's point-of-view in contrast to "fly-through" based systems.

As future work, we are investigating the integration of algorithms for procedural generation of textures and geometric details such as asphalted streets or stone mosaics. In addition, we are working on related real-time 3D rendering techniques to improve photorealism, including complex 3D vegetation models and shadows.

Acknowledgements

We would like to thank the urban surveying office Potsdam for the data set used in our case study, the Remote Sensing Solutions GmbH (www.rssgmbh.de) for the city model of Berlin used in Figure 1, and the 3D Geo GmbH (www.3dgeo.de) for providing us support to the landXplorer technology. We are also grateful to the German environmental foundation Deutsche Bundesstiftung Umwelt for supporting our research within the Lenne3D project (www.lenne3d.de) and to the Lenné3D GmbH for providing the tree models used in Figure 2. Finally, we would like to thank our students Oleg Dedkow, Anselm Kegel, Manuel Wellmann, Wolfgang Braunisch, Christian Schubert, Andrea Hentschke, Sebastian Kuhn, Jan Rehders, and Jan Zimmermann, who contributed to the SmartTerrain editor implementation.

References

Appleton K, Lovett A, Sünnenberg G, Dockerty D (2002) Rural landscape visualisation from GIS: a comparison of approaches, options and problems. Computer, Environment and Urban Systems 26, 141-162.

Danahy JW (2005) Negotiating public view protection and high density in urban design. In: Bishop, I. & E. Lange (eds.), Visualization in landscape and environmental planning, Spon Press, London: 203-211.

Deussen O. (2003) A framework for geometry generation and rendering of plants with applications in landscape architecture. Landscape and urban planning 64 (1-2), 105-113.

Döllner J, Hinrichs K (2002) A generic rendering system. IEEE transactions on visualization and computer graphics, 8(2):99-118.

Döllner J, Baumann K, Kersting O (2003) LandExplorer - ein System für interaktive 3D-Karten. Kartographische Schriften, Band 7, 67-76.

Döllner J, Buchholz H (2005) Continuous level-of-detail modelling of buildings in Virtual 3D City Models. Proceedings of the 13th ACM International Symposium of Geographical Information Systems, ACM GIS 2005, 173-181.

Döllner J, Hagedorn B, Schmidt S (2005) An approach towards semantics-based navigation in 3D city models on mobile devices. Proceedings of the 3rd symposium on LBS & TeleCartography, Vienna (*to appear*).

Finch M (2004) Effective water simulation from physical models, GPU Gems, Addison Wesley, 5-29.

Kersting O, Döllner J (2002) Interactive visualization of 3D vector data in GIS. Proceedings of the ACM GIS 2002, ACM Press, 107-112.

Lange E, Hehl-Lange S (2005) Future scenarios of peri-urban green space. In: Bishop, I. & E. Lange (eds.), Visualization in landscape and environmental planning, Spon Press, London: 195-202.

Muhar A (2001) Three-dimensional modelling and visualisation of vegetation for landscape simulation. Landscape and urban planning 54 (1-4), 5-17.

Paar P, Rekittke J (2005) Lenné3D - Walk-through visualization of planned landscapes. In: Bishop, I. & E. Lange (eds.), Visualization in landscape and environmental planning, Spon Press, London: 152-162.

Shah MA, Kontinnen J, Pattanaik S (2005) Real-time rendering of realistic-looking grass, Proceedings of the 3rd conference on computer graphics and interactive techniques in Australasia and South East Asia, 77-82.

Stock C, Bishop I (2005) Helping rural communities envision their future. In: Bishop, I. & Lange, E. (eds.): Visualization in landscape and environmental planning – technology and applications. Taylor & Francis, Oxon, UK.

Warren-Kretzschmar B, Tiedtke S (2005) What role does visualization play in communication with citizens. In: Buhmann, E., Paar, P., Bishop, I. & E. Lange (eds.): Trends in real-time landscape visualization and participation. Proc. at Anhalt University of Applied Science 2005. Wichmann Verlag, Heidelberg.

Werner A, Deussen O, Döllner J, Hege HC, Paar P, Rekittke J (2005) Lenné3D – Walking through landscape plans. In: Buhmann, E., Paar, P., Bishop, I. & E. Lange (eds.): Trends in real-time landscape visualization and participation. Proc. at Anhalt University of Applied Science 2005. Wichmann Verlag, Heidelberg.