From CAD to VR – Implementations for Urban Planning and Building Design

Mikael Johansson[1], Mattias Roupé[1]
[1]The Visualization Studio, at the Department of Civil and Environmental Engineering, Chalmers University of Technology, Gothenburg, Sweden.
http://www.chl.chalmers.se/main/visualiseringsstudion/index.htm

Abstract. *At present time, three-dimensional objects are often represented with 2D-data in urban planning and building design. In order to get all the involved parties to fully understand a certain project, this may not be enough. More and more projects therefore take use of the Virtual Reality (VR) technique as a complement to traditional 2D drawings and sketches. All the involved parties can then share a common frame of reference for all discussions regarding a certain project. Unfortunately, the technique is not yet adapted to fit the current building design process.*
In this paper, we present a solution for semi-automatic generation of a VR-model based on 3D CAD information and aerial photos obtained from the City Planning Authorities in Sweden. The data is imported to support real-time editing of terrain, roads and buildings. We also present a framework for importing 3D-models created in Autodesk Revit which enables a seamless integration of modern 3D CAD and VR-models. The features are implemented in a software developed at Chalmers Visualization studio (Gothenburg, Sweden) and technical details about terrain handling and speed-up techniques will be given.

Keywords. *Virtual Reality; 3D City modeling; Urban planning; Terrain; Visualization.*

Introduction

Today, the 3D content for real-time visualization is usually created by 3D artists or specialists in the VR-field. While this results in realistic models, they are more or less built from scratch and have a low degree of integration with the actual CAD-data for a certain project. The introduction of object oriented 3D CAD systems, such as Autodesk Revit, will hopefully ease the use of CAD-data in VR. Architects are now able to draw/sketch buildings directly in 3D and due to its object oriented and parametric nature, the content is no longer only a collection of lines and polygons. In order to visualize future buildings and infrastructure, existing surroundings and terrain has to be digitally created as well. One common solution is to use GIS-data and digital maps to create a terrain model covered by aerial photos. Information about existing buildings is often present and they can be modeled as volumes with façade textures applied. A major limitation in current implementations is the lack of editing possibilities of the VR-content during run-time. In order to fully serve as a base for discussion it has to be possible to insert/remove/edit the content interactively during meetings and presentations. In this paper we are focusing on developing a framework for semi-automatic construction of a VR-model from data already present in the current urban planning and building design process.

Previous Work

Limited information is available on a complete pipeline or framework for semi-automatic construction of a VR-model from existing data that take both existing and future terrain, buildings and infrastructure into account. We will therefore separate Previous Work into two main categories; Visualization of city-models and Visualization of buildings from 3D-CAD-models.

Visualization of City-models

Visualizing large city models in real-time has been a challenge for several years. Many papers has been presented that take both terrain and existing buildings into account. In general, the vast majority of presented solutions use a triangulated terrain surface, draped with aerial photos and combined with simplified 3D-volumes representing the existing buildings. Besides using a view frustum culling scheme controlled by a spatial structure (quad- or octree), the rendering of buildings is often improved by using level of detail (LOD), occlusion culling and impostors. The actual terrain is usually rendered using some sort of continuous LOD (CLOD) algorithm, such as ROAM ,(Duchaineau 1997), or Geometrical MipMapping, (de Boer 2000) which are based on a regular grid of height-samples. Some implementations, such as osgTerrain, (www.openscenegraph.org: May 2005), use a triangulated irregular network (TIN) divided into patches with several LOD's for each of them.

In some cases, the terrain is enhanced with 3D representation of streets, roads and pavements. Detailed information about implementations are rarely to find (www.vterrain.org: May 2005), but two different approaches are usually used; draped or stitched. The former just draw the roads on top of the terrain, whilst the second solution re-tesselates the terrain in order to stitch the road boundaries against it.

Visualization of buildings from 3D-CAD-models

In the field of building design, generic 3D-modeling software such as Autodesk's 3D Studio MAX or MultiGen Creator are today the most widely used ones for digital 3D content creation (Stamboulo-glou 2002). The models can be exported to generic 3D file formats and several commercial real-time visualization software's also enable strong compatibility with the 3D modeling software. Even if graphics hardware is getting faster, a detailed 3D model of a building can still be to complex to render in real-time, especially when viewed together with other detailed content. Several techniques is available for reducing the workload on the graphics hardware, such as LOD, geometry simplification and impostors. A disadvantage is that the previous mentioned applications aren't really used as the main tool for building and structural design in a project. They only serve as a tool for visual representation of buildings and structures. In order to be fully compatible with the actual design process, it seems that 3D content has to be created using some of the new object-oriented parametric 3D CAD-systems, such as Autodesk Revit.

Our implementations

The main challenge in our implementation has been to adapt the content creation pipeline to fit the current building design process. Effort has been spent on reusing the information present today, rather than adding time-consuming steps to construct a VR-model over a certain area. The techniques and innovations presented in this paper has been implemented in a software developed at Chalmers University of Technology. The application, called MrViz, is based on the open-source libraries OpenSceneGraph and wxWidgets.

TIN vs. Regular Grid

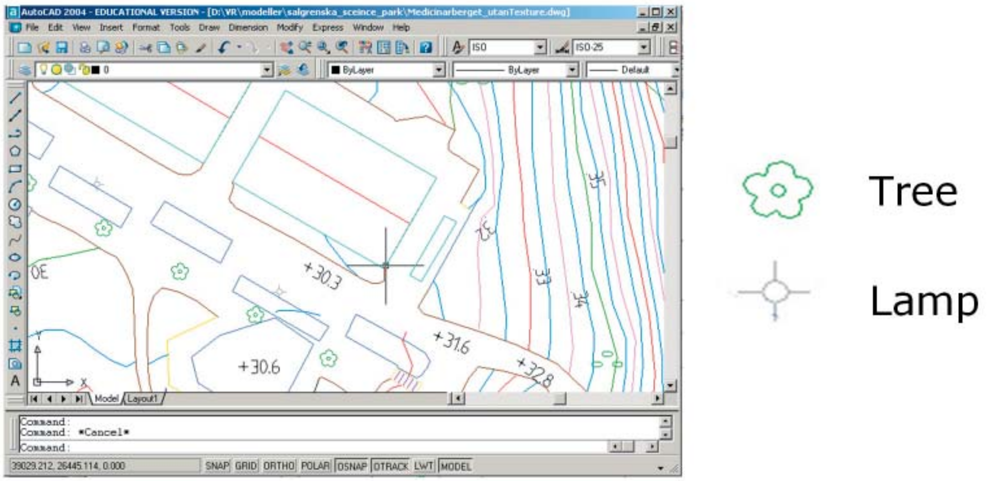
Terrain representations can roughly be separated into two main categories; Triangulated Irregular Network (TIN) and Regular Grid. A TIN has the advantage of representing arbitrary terrain geometry and is not constrained to a regular resolution. The technique can take local curvature into account when tessellating the input data (Heckbert 1995). Unfortunately, the structure is not very edit-friendly in real-time and can become difficult to use in a general LOD-system. Regular grids, on the other hand, have a fixed resolution but are very edit-friendly in real-time. The data can be stored as a grayscale image and indexing individual vertices is very fast. In addition, most recently presented terrain LOD-systems is based upon the principle of a regular grid.

In order to support terrain editing we have chosen to use a regular grid in our implementation. However, the implementation supports that local areas is replaced with TIN-based terrain when non-regular resolution is needed.

Existing environment

The City Planning Authority in Sweden often has 3D Base Map's available from the GIS-database. The model contains X, Y and Z coordinates for the roofs and roof details as well as 3D information about streets, pavements, water, open park areas, trees, streetlights, fences, etc. The database is exported to 3D dxf and dwg files from AutoDesk Inc.'s AutoCAD product, which is used by architectural companies when they project and draw the proposals for new buildings. Another input source is aerial photos and their attribute files that contain X and Y coordinates for the corners of the aerial photos.

The information in the dwg/dxf file is stored in different layers and blocks. Contour lines rep-


AutoCAD 2004 - EDUCATIONAL VERSION - [D:\VR\modeller\salgrenska_sceince_park\Medicinarberget_utanTexture.dwg]
File Edit View Insert Format Tools Draw Dimension Modify Express Window Help
ISO
ISO-25
ByLayer
ByLayer
Default
+30.3
+30.6
+31.6
+32.8
35
33
34
Model
Layout1
SNAP GRID ORTHO POLAR OSNAP OTRACK LWT MODEL
Tree
Lamp

```
Command:
Command: *Cancel*
Command:
```

39029.212, 26445.114, 0.000

Figure 1. Trees and streetlights are stored as "block symbols" in the dxf/dwg-file.

resenting the terrain are stored in one layer and roofs in another layer etc. Trees and streetlights are represented by “block symbols”.

During the import process of dwg/dxf files in MrViz, the user is able to choose what layer that contains terrain and what contains roofs. The trees and streetlights etc. are represented by 3D-objects that the user chooses during import. After this process the terrain, building walls and roofs are generated and trees and streetlights are placed automatically. The terrain surface is constructed by doing a Delauney triangulation using the irregular height samples as input (Sloan 1984). Roof-lines are extracted down to the terrain surface, resulting in simple volumes representing existing buildings. Finally, the aerial photos are automatically mapped on the terrain and roofs based on the information contained in their attribute files.

As stated before, our solution requires a regular grid in order to support real-time editing of the terrain. To build this structure we need to convert the irregular set of triangles to a regular grid of height-samples. This could be calculated by ray-casting on the CPU but we choose a much faster way by using the GPU and render a top-down orthographic view of the TIN and saving the resulting depth-buffer to a grey-scaled height map. In order to keep the precision we save it as a binary 16 bit .raw image. This information is then used to convert certain areas in the landscape where terrain editing has to be supported.

Depending on the size of the visualized area and the resolution of the aerial photos, texture memory could become critical. In order to avoid texture-swapping S3TC texture compression (www.developer.nvidia.com/attach/6585: May 2005) is used.

After the import process the user still needs to apply façade textures. We have implemented a very user-friendly and fast tool that automatically snaps the chosen façade texture to the extents of any selected geometrical face. This way we are able to apply textures to a very large amount of existing buildings in a short time.

Future buildings

The introduction of object-oriented parametric 3D CAD systems, such as Autodesk Revit, has given architects and engineers tools that not only seem to increase productivity but also represent geometry in 3D by default (Khemlani 2004). In addition to support real-time visualization of 3D content from generic 3D modeling packages, we have focused on implementing a simple framework for importing 3D CAD models created in Revit. Due to its intelligent and object-oriented database, the end result is more prepared to support automatic optimization for real-time rendering. The data is not a structured “triangle-soup”, but instead has detailed information on every object contained in it. This makes the conversion from detailed 3D-CAD to a real time suitable representation more convenient. For instance, detailed doors and windows is easily replaced with textured quads when viewed from a distance. Walls can work as occluders to cull away geometry behind them, and windows can be used as occluder holes to the wall occluder when the view is closer. The Revit API has been released at the time of writing this paper. We therefore choose to present our “pre-Revit API” import solution where the AutoCAD dwg-format is used as a bridge between Revit and MrViz. By use of a text-based preference file, Revit can export its 3D content to the AutoCAD dwg-format in a specified manner. In order to utilize some sort of object-orientation, all doors, windows and furniture’s are placed as blocks in the exported file. In the same way, roofs and different wall-types are placed in unique layers. This way we can “simulate” the appearance of detailed information when importing the content in MrViz. For instance, doors and windows are automatically replaced by textured quads when viewed from a distance.

Terrain Rendering

The terrain rendering is based on the technique called Geometrical Mipmapping introduced by de Boer (2000). The terrain is constructed from a grey-scaled height map and divided in rectangular patches of indexed triangle-lists. Every patch has several levels of detail and the complete dataset is organized in a quadtree for fast view frustum culling. Neighboring patches could end up with different resolution during rendering and in order to eliminate cracks we store all possible combinations of index-arrays. The memory consumption is kept low by using a set of global, pre-computed index-arrays. Due to its regular nature, every patch can also share the same set of global X- and Y-coordinates, translated to its right location during rendering. The unique z-coordinates are sent to the hardware through a 1D texcoord pointer and the data is combined in a simple vertex-shader to form 3D-coordinates. One problem with a patch-based LOD system is the unwanted effect of popping. This could be reduced by a technique called Geomorphing, (Wagner 2003), where the transition between different LOD's is performed by morphing. For the time being this is not yet implemented in our solution but the overall structure will support future implementation of it.

Terrain Editing

Real-time terrain editing is supported in MrViz by implementing a solution often seen in 3D game-editors such as CryENGINE Sandbox. By using a regular grid as the terrain representation we can actually treat the data as a grayscale image where the intensity corresponds to the z-coordinate (height-value) at a specific location. This allows us to edit the terrain directly in the 3D view or by painting in the corresponding height-map. With this functionality it is easy for any user to "sketch" modifications in the landscape and see the result of it before more accurate design is done using conventional methods.

Roads

Roads are created in MrViz by clicking out a path directly on the terrain in the 3D view. A 2D shape is then extracted along the path to generate the textured road geometry. In order to enable a more organic creation interface a piecewise quadric Bezier curve is evaluated from the paths control-points. A linear representation could have been used instead, but the current solution has shown to function well.

The actual rendering of roads can generally be performed by either draping them onto the terrain or by cutting out the roads from the terrain and stitching them at the boundaries. The later will result in a realistic visual appearance but does not really match our terrain rendering technique which is based on a regular grid in multiple resolutions. Draping the roads is a fast and edit-friendly way because the terrain never needs to be re-triangulated. The so called Z-fighting can be eliminated by geometrically offsetting the road or by using hardware Polygon Offset functionality (Shreiner 2004). Unfortunately, the technique will still be limited to render roads in or above the terrain plane. We have tried to enhance this functionality by using the stencil buffer to mask out portions of the terrain where roads will be drawn. First we render all the roads into both color and stencil buffer. When we render the terrain we do stencil testing to make sure no terrain is rendered where we have roads. Roads can, however, be naturally occluded by terrain hills and this has to be accounted for. One solution is to first render all the back-faces of the terrain only into the z-buffer. We then do depth testing against the back-faces of the terrain when we render the roads which means that all roads occluded by hills in the terrain will not be drawn at all. When finally rendering the front-faces of the terrain, only areas where non-occluded roads are present will be omitted. Unfortunately, this technique fails when a occluded road is intersecting the actual occluder. This is often not noticeable in a practical situation, but the technique definitely needs more research in order to be fully functional.

As for now, both Polygon Offset and the en-

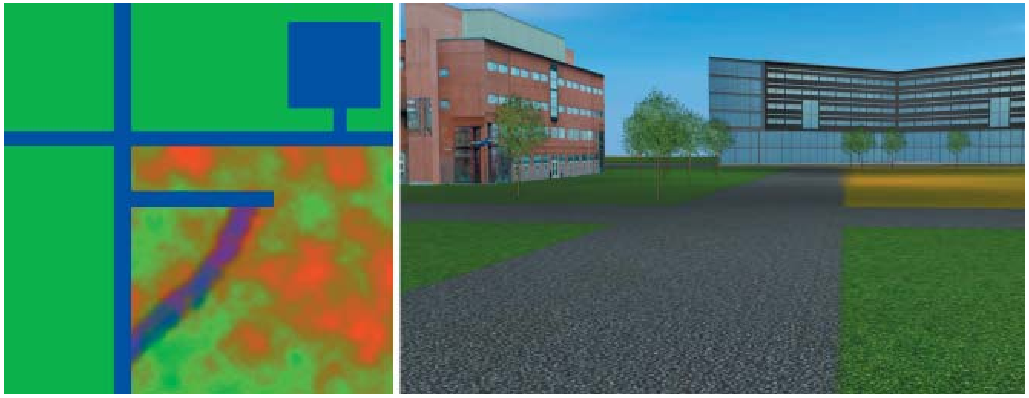


Figure 2. Image-based 2D planning. Color mask (left) and result (right).

hanced stencil version is implemented for the rendering of roads.

Image-based 2D planning

Image-based 2D planning refers to a technique where roads, pavements, grass areas, etc. is represented by different colors in an image. These colors are used as masks in a pixel-shader, and replaced with detail textures (figure 2, left). This way, the materials appears to have high resolution although the input image mask doesn't. This is a very simple interface for the user, who can create the mask image in any image editing software, such as Adobe Photoshop. The end result will lack 3D details but still serves as a good representation (figure 2, right). The technique is implemented in MrViz, but the actual image generation is still referred to any image editing software.

Reference Project

The techniques described in this paper are being used in a large urban planning project in Gothenburg, Sweden, at the time of writing. The project is a result of a new tunnel being constructed and is covering approximately seven square kilometers in the heart of the city. At an early stage it

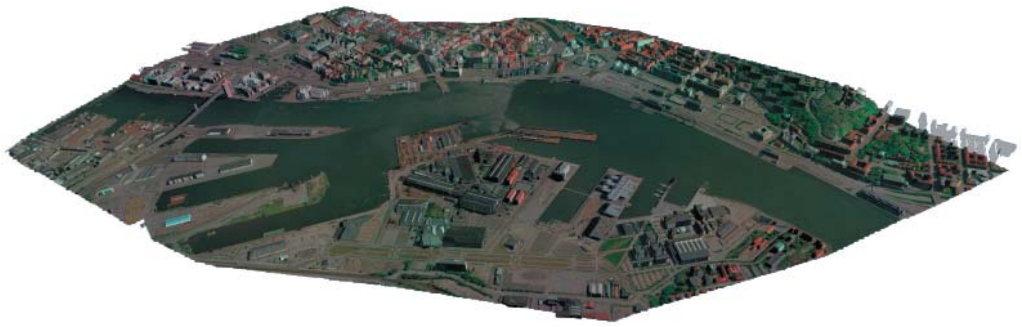


Figure 3. Result of semi-automatic import of existing buildings and terrain.

was decided that VR-technology should be used as an interactive communication base for decision making and public information. Furthermore, architects should be able to use the VR-model as a base for presenting different proposals. The area covers both existing and new buildings and infrastructure which makes it a good test-bed for the implementations described above. Figure 3 shows the result of the semi-automatic import of the existing environment.

Conclusions and Future Work

In this paper we have presented practical solutions and implementations for rendering large city-models together with representations of future buildings in real-time. The methods and input-data used are integrated with the actual building design process. In contrast to many other implementations we support real-time editing of the terrain and 3D content. Future research and implementations will be focusing on performance optimizations together with further development of the road visualization technique.

References

S.W.Sloan, G.T.Houlsby.: 1984. An Implementation of Watson's Algorithm for Computing 2-D Delauney Triangulations. Advanced Engineering Software, Volume 6, Number 4, 1984

Khemlani, L.: 2004. Autodesk Revit: Implementation in Practice. Autodesk available online at www.autodesk.com/bim

Horne, M.: 2004. Beirut: Three dimensional modeling of the city. 1st International SCRI Symposium, University of Salford.

Stambouloglou, E. and Shan, J.: 2002, Building modeling and visualization for urban environment. Symposium on Geospatial Theory, Processing and Applications, Ottawa 2002.

Heckbert, P.S. and Garland, M.: 1995, Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

de Boer, W. H.: 2000. Fast terrain rendering using geometrical mipmapping, Flipcode, October 2000, available online at http://www.flipcode.com/articles/article_geomipmaps.pdf

Shreiner, D (ed.): 2004. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.4, Fourth Edition, Addison-Wesley.

Kada, M., Roettger, S., Weiss, K., Ertl , T. and Fritsch , D.: 2003. Real-Time Visualisation of Urban Landscapes Using Open-Source Software In: Proceedings of the ACRS 2003 ISRS, 24th Asian Conference on Remote Sensing & 2003 International Symposium on Remote Sensing, Busan, Korea.

Ulrich, T.: 2002. Rendering Massive Terrains using Chunked Level of Detail Control. In Course Notes of ACM SIGGRAPH 2002, volume Course 35, 2002

Duchaineau, M. A., Wolinsky, M., Sigeti, D. E., Miller, M. C., Aldrich, C., and Mineev Weinstein, M. B.:1997. ROAMing terrain: real-time optimally adapting meshes. In IEEE Visualization, pages 81-88.

Wagner, D.: 2003, Terrain Geomorphing in the Vertex Shader, ShaderX2: Shader Programming Tips & Tricks with DirectX 9.0, Wordware publishing.