Final Report Subproject “Interactive Visualization of 3D topography”

BSIK Program: “*Ruimte voor Geo-informatie”*
Project RGI-011: *“3D topografie”*
Projectleader: prof. dr. ir. Peter van Oosterom (TBM)
Projectmanagement: Elfriede Fendel (TBM, 84548)
Subproject: *“Interactive Visualization of 3D topography”*
Running time: 1.1.2008 – 31.12.2008 (1 year)
People involved: ir. Gerwin de Haan, prof. dr. ir. Erik Jansen (supervision)

Project Overview

This document provides a global overview of the activities and results of the subproject *“Interactive Visualization of 3D Topography”*. The overall goal of this subproject was to explore possibilities of interactive 3D visualization of various models and datasets in the context of the RGI-011 3D topography project. The following topics were considered:

- 3D Visualization on the PLANAR stereoscopic display
- Interactive Visualization of 3D Topographic models in general
- Interactive Visualization of large terrains and point cloud data

Parts of this work appeared as publications and were shown during many demonstrations. Selected aspects of interest will appear in a separate technical report.

The PLANAR stereoscopic display

A PLANAR 3D display system was available in the 3D Topo project but its use had not been explored. This 3D stereoscopic display consists of two TFT monitors mounted at a 90 degree angle with a semi-transparent mirror in between at 45 degrees. When viewed with a pair of stereoscopic glasses, each eye sees only the image of one of the monitors. If each image is carefully constructed with a correct perspective for the corresponding eye, the human visual system can interpret depth from the images.

Illustration 1: The PLANAR display

The 3D display requires two DVI video signals, one for each monitor. In this setup, these signals are provided by a single video card (nVidia Quadro) through two output channels. One of the outputs is wired through an add-on PCI card (provided with the PLANAR system), to mirror the video signal.

To get a correct stereo image from graphics software, one can either adjust the software for stereo viewing or let a stereo driver provide emulated stereo for general 3D application. Provided installation instructions for the system were not compatible with recent video drivers. For the Windows OS, a regular Quadro driver set provided by nVidia was installed. Recent driver versions (~ Feb. 2008) were found not compatible with older nVidia stereo drivers. For this reason, only applications supporting quad-buffered stereo can be viewed without modifications.

Visualization Software

One important 3D Topography viewing application, *Google Earth*, does not natively support quad-buffered stereo rendering. For this application, a stereo plug-in was purchased and installed (TriDEF). This plug-in intercepts 3D graphics commands from the application and duplicates them for the two screens at slightly different viewpoint, resulting in a stereoscopic view. Although this slows the application down considerably and can increase (video) memo-

ry consumption, it provides a useful stereo view.

To experiment with the exploration of 3D topographic models we use our VRMeer toolkit as the main software component. This toolkit was developed at the TuDelft computer graphics group and is used to build interactive VR applications for various VR systems in the TuDelft VRLab. It builds on features of the OpenSceneGraph (OSG) 3D graphics rendering toolkit. This is a modern open-source software toolkit that is widely used and under active development. The VRMeer toolkit adds support for various 3D display systems and 3D input devices and basic 3D interaction techniques. The VRMeer toolkit does support quad-buffered stereo rendering. It was installed and configured for the provided PLANAR system such that all VRMeer and other OSG-based applications can display in stereo.

Although the base programming language is C++ for both OSG and VRMeer, we constructed wrapping layers for the Python programming language for both. This allows for a more high-level prototyping of new functionality and applications. During this project, features on this area have been extended. It must be noted that many other software toolkits and libraries for GIS data, such as the Geospatial Data Abstraction Library (GDAL), come with Python interfaces. This facilitates fast integration and linking of general 3D visualization applications with GIS specific data formats.

Available Data, Activities

There was a variety of 3D Topography datasets available in this project. The first main dataset was the Tetrahedronized Network (TEN) created by Friso Penninga. A second dataset consists of TuDelft campus buildings, mainly manually modeled in Google Sketchup and provided by Sisi Zslatanova. A third dataset was the test data provided by *"Actueel Hoogtebestand Nederland"* (AHN), consisting of LiDAR datasets and its derivatives. In this project, we have focused mainly on the challenges that accompanied the AHN dataset, although preliminary loading of Google Sketchup models was used as well.

The main activity to perform on the models was visual exploration, characterized by the task of 3D navigation. Other tasks of visualizing derived data aspects or manipulation were only shortly discussed, and were not considered a main aspect of this subproject.

Data processing

The wide variety of models was constructed in a wide variety of formats and needed some form of processing before it can be used in a 3D visualization application. Here, we will briefly highlight their use in this section:

Building models

There were building models made available from TuDelft campus. These were manually modeled in Google Sketchup, and are mainly used for rendering in Google Earth. The data is stored in the KML format (Keyhole Markup Language). This format mainly consists of basic geometrical information (position, orientation, basic geometric shapes) in an XML-based document, and detailed 3D models in COLLADA documents. The OpenSceneGraph toolkit can be extended with a plug-in that allows the loading of COLLADA based models directly in the scene. This basic functionality was used to inspect separate building models within VRMeer applications. A more extensive support of KML features can be extended by using Python, e.g. to transfer basic geometric information into the scene graph.

Terrain model

One goal was to render the AHN digital elevation maps (DEMs) with textures. OSG provides an extension for rendering large terrain models. The accompanying VirtualPlanetBuilder (VPB) program was used to generate these models. This program intelligently combines (several tiles of) DEMs and textures into a single Triangulated Irregular Network (TIN) model in a pre-processing phase. The final model contains scene graph nodes which in turn contains terrain nodes with discrete level-of-detail tiles as children. These nodes are constructed in a set of files, which can be read from disk or from a webserver. During runtime, OSG performs the calculation to determine which tiles should be loaded, and in which level-of-detail.

The AHN grid data provided consists of several tiles in the ESRI ASC format (e.g. 0.5m grid cells, height value in meters for each cells). Two types of texture information were provided: raw RGB data aqcuired by the LiDAR scanner and high-quality aerial photos. The raw RGB data was provided in geo-referenced GeoTIFF images. The

high-quality imagery was in the ESRI ECW format, but did not contain any geo-referencing and was not used in the final result.

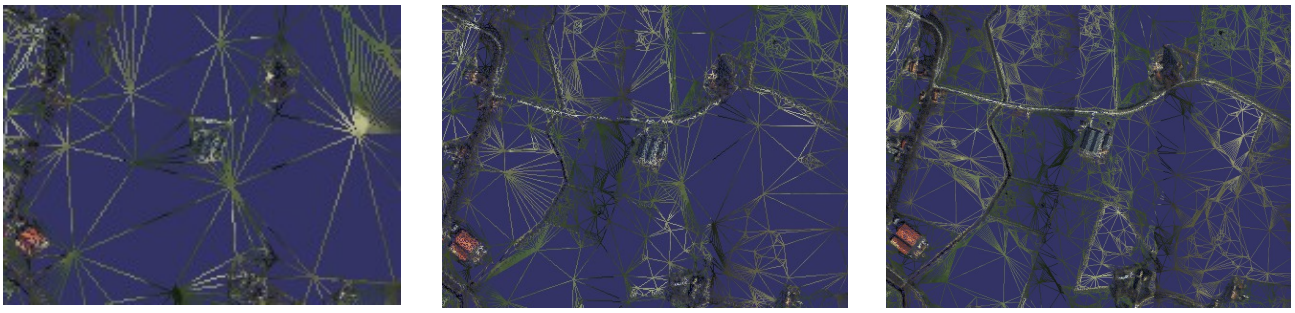

Illustration 2: Different levels-of-detail visible in the wireframe of the TIN terrain model

The AHN digitial elevation grids and raw RGB textures could be fed directly to VPB. The read-in of this data and necessary coordinate system conversions was performed by GDAL toolkit. VPB generates compiled binary OSG files (IVE format). Several quality and terrain settings can be selected. Depending on the amount of tiles and required quality of the results, generation can cost between several minutes to several days.

The resulting terrain model can be loaded directly in any OSG model viewer or our specialized, VRMeer based terrain viewer. The visual quality differs with the distance the viewer takes to the landscape, and the spatial frequency of landscape features. When using a DEM without filtering, buildings appear to be “draped” with a sheet. It must also be stated that the low color quality of the raw RGB textures hindered a consistent appearance. A consistent framerate of 60 frames per second is achievable during exploration, without disturbing artifacts when loading in different levels of detail.

Illustration 3: TIN terrain rendering for Middelburg (AHN data). Notice that sharp features (e,g, buildings,trees) are not well captured by the TIN.

Point cloud

The raw LiDAR measurements from the AHN contain a wealth of information. Much of this information is lost in the post-processing steps that are used to generate DEMs. A sample set of pointcloud consists of approximately 20 million points with XYZ coordinates (floating point number, Rijksdriehoek coordinate system, cm notation, ~3 cm accuracy). When this pointcloud is directly converted to a scene graph model, frame rates can drop to below one frame per second.

Our goal was to also be able to render these raw point clouds at interactive frame rates. There are two main challenges

here, the amount of data and the visual rendering. In this project we created a proof-of-concept system with one is able to explore large sets of point clouds. Just as in the terrain model generation, this involves pre-processing and rendering optimizations.

We have created a preprocessing program to generate a level-of-detail representation of the point clouds that can be paged, similar to what the VPB does for TINs. This program constructs a hierarchical tree (bintree, quadtree, octtree) for a pointcloud, and distribute the available points over the nodes at different levels in this tree. When viewed from a distance, the level-of-detail algorithm displays only the higher level leaves, so only a subset of points in the tree are visible. A careful balancing of number of points and the number of levels of detail is necessary to avoid loading and rendering too much data. The filtering process to determine which points should be visible at what distance is an item of feature research.

The visual representation of the LiDAR point samples is done through rendering point primitives. If points were rendered as a single pixel, only limited depth perception is gained. A graphics shader program therefore determines the size of a point to generate a larger (or smaller) splat, based on the distance to the viewer. It also includes a fogging factor to decrease the saturation of colors when points are more in the distance. Coloring can be done based on texturing or derivative data (e.g. mapping point height to a color scale)

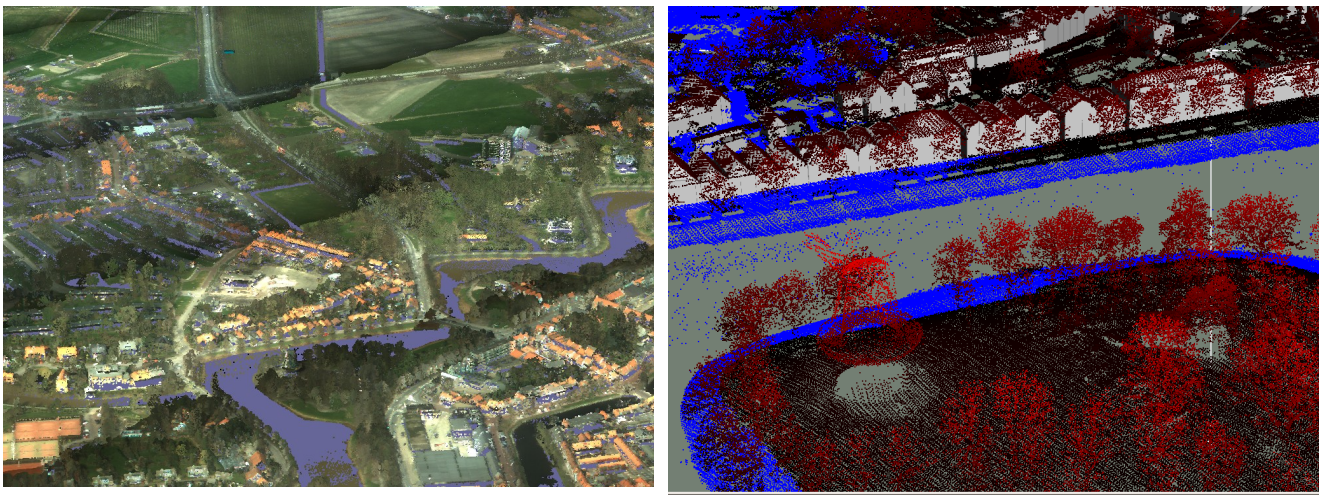

Illustration 4: Point cloud visualizations of Middelburg (AHN data). The left image shows multiple data tiles, variable point sizes and color reconstructed from raw RGB colors. The right image uses small point size and black-to-red color to indicated point height. It also includes reconstructed buildings.

The drawback of pointclouds is that they do not necessarily give a continuous or filled visual representation, e.g. no sampling gaps are filled. This is not a problem when viewing from a distance, but can be if observed from closeby (compare left and right images of illustration 4). A combined representation of poth point clouds and terrain model gives the best general impression of a landscape with sharp features such as buildings. For demonstration purposes, we also integrated 3D surface models of buildings that were reconstructed from the point cloud.

Interaction

General 3D viewers often do not include good interaction techniques to study 3D topography data. Google Earth is an example of a specialized viewer for topography content. However, this application has a main drawback that it cannot simply load custom terrain models and/or point clouds. Based on our VRMeer system, we have build a viewing application geared towards exploring 3D topography data.

The main aspect of this viewer is the 3D navigation. We created simplified mouse based view navigation, as well as navigation with specialized devices such as the *Space Mouse* and the *Wii Balance Board*. On more immersive VR systems, such as a Workbench or CAVE system, one can also use head tracking and 3D pointing devices.

By using our Python-based prototyping system, the application can be quickly reprogrammed to include case-specific viewing techniques, specialized interfaces, or loading of models. As an example, we included mouse-based sketching to quickly draw sketches on a 3D terrain for communicating details.

Conclusions and Future work

In this subproject we have explored possibilities of interactive 3D visualization of 3D topography datasets. From our experience and observer reactions during demonstrations, especially the exploration of large, detailed datasets was impressive. A combination of stereographic 3D display with smooth 3D navigation really invites users to explore the data freely from different viewpoints.

From a technical point of view, much can be gained from optimizing and balancing the data structures and visual representations, and this is part of future work. An extensions to web-based viewing is also of interest, but would require more research to minimize network bandwidth and optimize rendering on low-end clients.

For end-users, specialized interaction tools for measurement, annotation and even editing could be included for use in practice. Here it is also important that a user-friendly content-pipeline should be constructed, to translate aqcuired data and models for integration in a 3D interactive viewer.

Publications

- Publication *"Flexible Architecture for the Development of Realtime Interaction Behavior"*, IEEE VR 2008 SEARIS workshop.
- Proposal GIN/RGI symposium, *"De wereld is niet langer plat; de toekomst van NGII begint met 3D-Topografie"*, Gerwin de Haan, Friso Penninga, Edward Verbree, Sisi Zlatanova and Peter van Oosterom, submitted June 27, 2008.
- Poster, *"Using the Wii Balance Board(TM) as a Low-Cost VR Interaction Device"*, by Gerwin de Haan, Eric J. Griffith and Frits H. Post. ACM conference on Virtual Reality Software and Technology 2008, ACM VRST 2008, Oct. 27-29th in Bordeaux, France.

Demonstrations and Publicity

- Demonstration, RGI-011 Internal project meeting, OTB TU Delft , March 6, 2008.
- Student Volunteer, IEEE Virtual Reality 2008 & 3D User Interfaces Symposium,Reno, USA, March 8-12, 2008.
- Panelist, IEEE Virtual Reality 2008, "Building the future of - and a career in - Virtual Reality", March 11, 2008.
- Industrial Event organizer, Eurographics Symposium on Virtual Environments, Eindhoven, May 29-30, 2008.
- Demonstration, AHN Gebruikersdag 2008 , Zeist, October 1, 2008.
- Invited Talk "Virtual Reality for the Next Generation Traffic Control Room" at Time4innovation symposium, Uithoorn, October 2, 2008.
- Student Volunteer, ACM User Interface Software and Technology (UIST) Conference, Monterey, USA, October 19-22, 2008.
- Demonstration, RGI-011 Internal project meeting, Nedgraphics, Vianen, October 30, 2008.
- Demonstration, Geomatics/GIS-day, Aerospace TU Delft, November 19, 2008.
- Demonstration, Core Spatial Data seminar., december 18, 2008.