VISUALISATION USING GAME ENGINES

Dieter Fritsch, Martin Kada

Institute for Photogrammetry (ifp), University of Stuttgart, Germany
Geschwister-Scholl-Strasse 24D, D-70174 Stuttgart
firstname.lastname@ifp.uni-stuttgart.de

Commission V, WG 6

KEY WORDS: Visualisation, Virtual Reality, Real-Time, GIS, Modelling

ABSTRACT:

Geographic Information Systems (GIS) and Computer Aided Facility Management-Systems (CAFM) are currently undergoing the transition to storing and processing real 3D geospatial data. Applications for this type of data are, among others, location based services, navigation systems and the planning of large-scale construction projects. For presentation purposes and especially when working in the field, powerful visualisation systems are needed that are also capable of running on mobile devices like notebooks, personal digital assistants (PDA) or even cell phones. In such application areas, the free movement of the viewer's position and the interaction with the data are of great importance. Real-time visualisation of 3D geospatial data is already well established and also commercially successful in the entertainment industry, namely in the market of 3D video games. The development of software in this field is very cost-intensive, so that the packages are often used for several game products and are therefore universally applicable to a certain extend. These so-called game engines include not only visualisation functionality, but also offer physics, sound, network, artificial intelligence and graphical user interfaces to handle user in- and output. As certain portions or sometimes even the whole engine are released as open source software, these engines can be extended to build more serious applications at very little costs. The paper shows how these game engines can be used to create interactive 3D applications that present texture-mapped geospatial data. The integration of 3D data into such systems is discussed. Functionality like thematic queries can be implemented by extending the internal data structures and by modification of the game's accompanying dynamic link libraries.

1. INTRODUCTION

Since the time computer graphics has been introduced, the demands for visualisation techniques have grown continuously. Today, the visualisation of three-dimensional worlds seems to be a demanding task requested by many geo-related disciplines. This has led to Scientific Visualisation, which is associated with solving visualisation problems of all kind (McCormick, DeFanti and Brown, 1987). It offers algorithms, software packages and advanced interactive tools (such as data gloves and other haptic interfaces) for graphics workstations, high end rendering machines and CAVE environments (see also Fritsch, 2003).

Complementary are the developments in the computer game industry that has been developing game engines with amazing 3D computer graphics capabilities since the early 1990s. Due to the increasing interest in the consumer market, tremendous progress can be observed in the hardware and software. Game engines are powerful software packages that efficiently use rendering pipelines, special data-structures and speed-up techniques to visualise texture mapped 3D objects, scenes and 3D worlds in real-time (see e.g. Harrison, 2003). These software packages run nowadays on every commodity PC and 3D games already make their way on PDAs and even cell phones. The overall question is how to make best use of available technology to make the right application.

Only few large projects use sophisticated hardware and software. For many 3D mapping applications, only commodity hardware and software is available. But even the daily user of computer graphics still aims at high quality visualisation at low costs. Game engines might be the missing part for realising visualisation software for geo-related applications. The rendering performance and quality continuously increases as the game industry develops and implements new visualisation technologies. And many of the last generation engines or game-related libraries are now available for little or even no cost in the form of open-source software. The following sections will focus on both indoor and outdoor visualisation, will introduce some assorted game engines and show prototypical applications that have been built upon them (see e.g. Figure 1). Another aspect will be

Figure 1. Indoor visualisation showing a workspace at the Institute for Photogrammetry (ifp) rendered in real-time by the Quake 3 Arena game engine.

the real visualisation of vegetation as this has been of special interest in game engines in the last years and big progress has been achieved.

2. INDOOR VISUALISATION WITH GAME ENGINES

Today, 3D computer games are highly complex systems that consist of a universal game engine and the specific game elements like the game rules and game data (e.g. geometry, textures and sound files). Main emphasis is put here in the game engine. This module is the heart of the computer game and represents the basic framework independent of the game. This general purpose feature allows the use of the engine for other applications, e.g. the indoor visualisation of building elements. Game engines incorporates all sorts of elements that are vital to a game like physics, graphical user interface (GUI), artificial intelligence, network functionality, sound and event engine. Some game engines even contain scripting languages which makes it very easy to adapt the engines to one's own needs.

The computer games Quake III Arena (developed by id Software) and Max Payne (developed by Remedy Entertainment Ltd.) are action games, also called 3D Shooter or First Person Shooter. The player moves around in an ego-perspective and fights by means of several weapons within the 3D world. This can be done either alone in single player mode or with multiple players in a network environment like e.g. a LAN or the internet. Both games have in common that the visualisation engine is what's generally called an indoor engine. These engines are optimised by the use of specialised indoor speed-up techniques like portal culling, a very popular technique first introduced by (Airey, Rohlf and Brooks Jr., 1990). Based on the idea that walls are often large occluders, a viewer can only see into adjacent rooms through portals, which can be e.g. a door or a window. A potentially visible set (PVS) is pre-computed for all sets of viewpoints, a sort of database from which the rooms that are visible to the viewer are identified. For densely occluded architectural scenes, the algorithm is able to cull away the better part of the scene. Unfortunately, because the engines support very detailed environments, the virtual worlds are rather small and delimited.

2.1 Data Acquisition and Integration

The aforementioned game engines are of notable interest as both offer very good support for modifications. The game related parts are available as source code and there exist free editors for the creation of virtual 3D environment, which are in the context of game engines usually called maps. These maps are modelled via Constructive Solid Geometry (CSG) by logically combining simple forms like cuboids, pyramids and spheres. It is well-advised to create a coarse model before modelling the finer elements of the map. The resulting geometry can then be texture mapped with images that can either be artificial or be generated from photographs. All necessary steps are demonstrated in Figure 2 (Beck 2002).

Once the map is complete, it can not yet be used by the game engine, because the engine itself does not understand the data that comes from the editors. So the map must first be compiled with designated tools that transform the CSG data into a boundary representation (B-Rep). In order to improve the performance of the scene rendering, the visibility database is generated and the lighting and shading are pre-

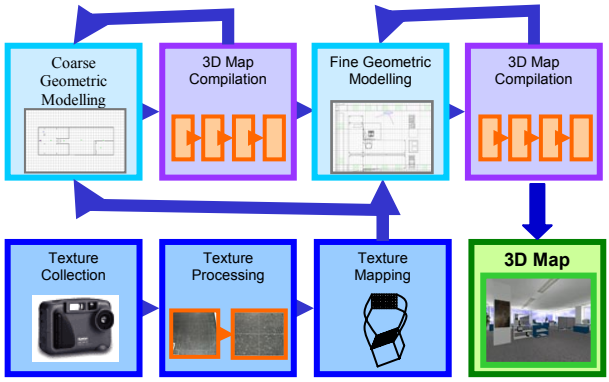

Figure 2. Workflow for generating 3D textured maps in Quake III Arena (Beck, 2002).

computed (Abrash, 1997). The whole compilation process is also depicted in Figure 3 (Beck 2002).

A serious problem is, however, that existing datasets are unlikely stored in a format that the game engine's map compilers do understand or support. But fortunately, the game tools are often available as source code so that the compilers can be modified to one's own needs.

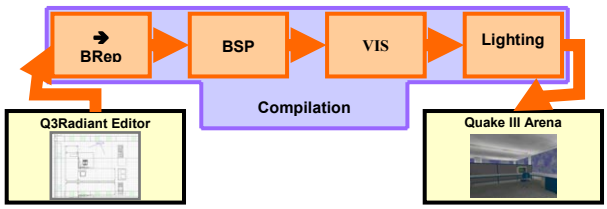

Figure 3. Compilation process in Quake III that transforms a CSG model into a boundary representation and pre-computes a potential visibility set and light maps (Beck, 2002).

2.2 Analysis Functionality and Interactivity

The analysis of data is a very important aspect of GIS. Even though game engines do not offer a great variety of that kind of functionality, they at least feature some interesting possibilities. By means of adding new entities to the Quake 3 Arena engine, it is possible in the Q3Radiant editor to define new data that can be bound to every object in the map. (Beck, 2002) e.g. realises a thematic query in this way where the objects are prompted by "shooting" at them. By removing the weapons from the game, the result is a simple point and click mechanism. The underlying object data, which is textual information about rooms and workspaces, is then displayed on the screen (Figure 4).

Another engine element of great use is the path finding algorithm that is utilised by the artificial intelligence unit to control the non-player characters in the game. This functionality can be turned into an indoor navigation system that guides the user through the virtual building. (Pfeiffer, 2002) implements a virtual museum guide that walks the visitor through the exhibitions using the routing and trigger functionality of the Max Payne game engine (see Figure 5).

Figure 4. The result of a thematic query shows that this door leads to the room of the GIS group.

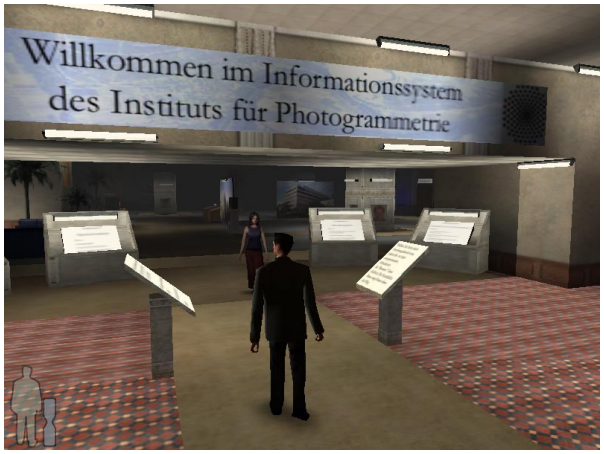

Figure 5. A virtual guide leads the user through the different fields of science in the exhibition room of the information system of the Institute for Photogrammetry (ifp).

3. UNREAL ENGINE 2

The Unreal Engine 2 (developed by Epic Games) is one of the most widely used game engines to date. Because it is a cross-platform solution, a broad range of products from PC and video games to architectural visualisations have been already developed with it. Being optimized for both indoor and outdoor environments, it is one of the most modern and versatile engines (see Unreal, 2004). Like most other game engines, the technology is encapsulated in a binary runtime library, while the game related parts of the Unreal games are available as source code in a scripting language called UnrealScript. The novel approach of Epic Games is that they released the Unreal Engine 2 Runtime free for non-commercial and educational use. This means that there is no need to buy the game itself to run modifications and applications that have been developed by the community. The runtime even includes the map editor UnrealEd and header files for C++ programmers. Beginners do find lots of technical documents and even video tutorial that teach level design, script programming and much more.

Figure 6. The Unreal Engine 2 is equally suited for indoor and outdoor environments and produces stunning landscape images in real-time.

4. OUTDOOR VISUALISATION USING OPEN SOURCE LIBRARIES

So far, the focus of this article was on the visualisation of indoor environments. Most geo-spatial data are, however, rather outdoor data like e.g. digital height models or 3D city models. There also exist powerful game engines that feature outdoor rendering capabilities. The game engine Torque (developed by GarageGames) combines both indoor and outdoor visualisation modules into one software package (Torque, 2004). But there also are powerful open source game-like libraries available that make stunning outdoor visualisation applications possible with very little effort. These libraries have matured so that very little programming effort is needed to create one's own visualisation application.

4.1 Open Scene Graph

The Open Scene Graph (OSG) is a cross-platform C++ / OpenGL library for real-time visualisation. It has become a powerful alternative to traditional tools like Performer and is freely available under the GNU LGPL at (Osfield, 2004). The library not only features high performance rendering capabilities and excellent support for PC graphics accelerators, but also offers stereo mode and a broad variety of loaders for many common data formats. Several people from the open source community already contributed plug-ins and exporter for a number of popular modellers like 3D Studio Max or Blender. For the purpose of moving through the datasets, there exist camera manipulators that simulate movement in a car or in an airplane. The drive camera manipulator even uses collision detection so that the virtual vehicle stays on the ground. OSG has been successfully used in non-commercial games and virtual reality applications.

4.2 libMini

For the real-time visualisation of digital terrain models, a continuous level-of-detail (C-LOD (Lindstrom et al., 1996)) approach is generally used. The C-LOD terrain rendering library libMini recursively generates triangle fans during the view-dependent generation of the quad-tree structured triangulation (Roettger et al., 1998). The library is licensed

under the terms of the GNU LGPL and can be downloaded from the home page of the author (Roettger, 2004). The API is simple and can be easily integrated in software packages like e.g. OSG by calling the terrain rendering functions right before the main render action. To suppress popping artefacts that can be otherwise experienced because of changes in geometry, a technique called geomorphing is applied which slowly moves newly introduced vertices from a position on the terrain to its final position.

4.3 GISMO

Within the scope of the project GISMO, an application for the real-time visualisation of large-scale urban landscape models has been realised that is based on Open Scene Graph and libMini libraries (Kada et al., 2003). The aim of the project was to generate and interactively visualise the city of Stuttgart, Germany. With respect to the desired flexibility to support walkthrough and flyover applications, a combined approach using continuous level-of-detail and the impostor technique was used to speed up the visualisation. The software proved to be able to render a 50*50 km area with 36.000 building models at interactive frame rates (Figure 7 and Figure 8).

Figure 7. Overview of a visualisation of the city of Stuttgart using Open Scene Graph and libMini.

Figure 8. To improve the visual appearance, the façade textures of 500 buildings that are located in the main pedestrian area were captured.

5. VEGETATION

The visualisation of landscape models that only feature terrain and building data tend to look dull and lifeless. Strikingly is particularly the absence of trees and other vegetation. Because of the complexity of these types of objects, image-based rendering techniques are generally used for their visualisation. So called Billboards, e.g., replace complex objects by an image that is projected on a transparent quadrilateral. Provided that the viewer stays close to the ground level, the image of a tree is a good approximation of the real geometry for all view points. As the viewer moves around the scene, the quadrilateral is rotated so that the image always faces forward. If only one single image is used, however, the tree looks bogus when looked at from above or from close distance. A good overview on real-time tree visualisation is given in (Remolar et al., 2002).
The commercial software package SpeedTree (developed by Interactive Data Visualization, Inc.) visualises trees by using real geometry for the trunk and branches and billboards only for the leaves (SpeedTree, 2004). It is becoming very popular

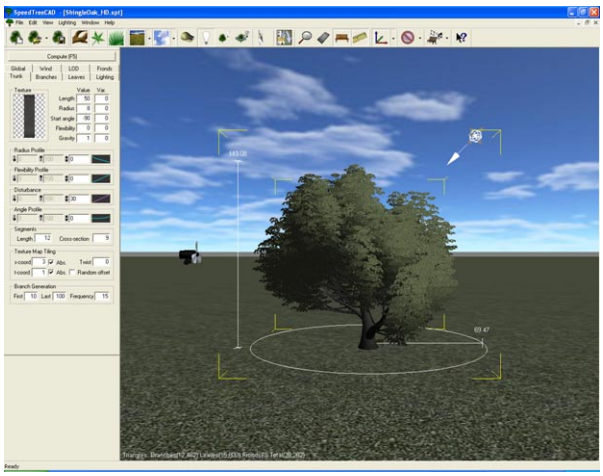

Figure 9. The CAD application of SpeedTree.

in game engines as it features realistic tree models and proves to be able to visualise literally thousands of trees in real-time. Included in the software package is a three dimensional CAD application, where tree models can be created and modified in an interactive environment (see Figure 9). The finished tree models can then be imported into standard modelling and rendering software like Discreet's 3D Studio Max or Alias | Wavefront's Maya via the use of plug-ins. For real-time applications, a C++ library called SpeedTreeRT is also available which efficiently renders the tree models. Integrating SpeedTree models into existing game engines is straightforward and involves only a few calls to the easy to use SpeedTreeRT API (Figure 10).
In order to increase the visual impression, SpeedTree features wind effects that are based on vertex shaders. Shaders are small programs that run on the 3D graphics card and make all kinds of special effects possible in real-time. For the real-time rendering of grass and grass-like vegetation, the developers of SpeedTree currently work on a library called SpeedGrass. This library will further improve the visual appearance of real-time landscape visualisations.

Figure 10. The GISMO client uses SpeedTreeRT to visualise realistic looking trees in real-time.

6. CONCLUSION

The article gave an introduction to 3D game engines and how these software packages can be used to develop serious visualisation applications. A problem is still, however, to get existing spatial data into a format that the game or the accompanying editors and tools understand. Once this obstacle has been overcome, the powerful rendering modules create beautiful indoor and outdoor scenes in real-time. If desired, the other modules of the game engine enable physically correct walkthroughs or fly-over even in a multi-user world. Such applications are far more desirable then pre-generated movies that are still commonly used for presentation purposes and the demand for real-time systems steadily increases. Especially in application areas like urban planning, emergency response, tourism and traffic management, the freedom of movement that game engines offer are of great interest.

Game engines aim for PC environments, so that the visualisation of photo-realistic, textured landscapes is now available for almost everyone. With the introduction of mobile 3D graphics chipsets that will soon make their way into cell phones, handheld video consoles and portable multimedia stations, 3D games and visualisation will be at everyone's hand at all times. There are many game engines available as source code or as binary runtime library that are free for non-commercial applications. Other sources for powerful visualisation software are the many game-like libraries available in the internet. Two such libraries, namely OpenSceneGraph and libMini have been successfully used in the real-time visualisation of large-scale urban landscapes. The project GISMO has demonstrated that these libraries are capable of rendering huge data sets at interactive frame rates and are thus a good alternative to commercial products. They are also very easy to use and can be easily integrated into other software products.

A big improvement to the realism of virtual environments is the inclusion of vegetation. Tree libraries are now available that are successfully used in commercial game products. Other libraries will soon follow that will be able to render all kinds of vegetation like grass and bushes.

7. REFERENCES

Abrash, M., 1997. Graphics Programming Black Book, Special Edition. Coriolis Group Books. Scottsdale, AZ, USA.

Airey, J., Rohlf, J., Brooks Jr., F., 1990. Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. In: Proceedings of the 1990 Symposium on Interactive 3D Graphics, pp. 41-50.

Beck, M., 2002. Realisierung eines Geoinformationssystems – Visualisierung und Analysefunktionalität mit einer 3D-Engine. Master Thesis, Stuttgart University, Institute for Photogrammetry (ifp), (not published)

Fritsch, D., 2003. 3D Building Visualisation – Outdoor and Indoor Applications. In: Photogrammetric Week '03, Wichmann Verlag, Heidelberg, pp. 281-290.

Harrison, L.T., 2003. Introduction to 3D Game Engine Design Using DirectX 9 and C#. Apress, Berkeley, CA, USA.

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. (On CD-ROM)

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L.F., Faust, N. and Turner, G., 1996. Real-Time, Continous Level of Detail Rendering of Height Fields. In: Proceedings of SIGGRAPH '96, pp. 109-118.

Osfield, R., 2004. Open Scene Graph. http://openscenegraph.sourceforge.net/downloads/index.html.

Pfeiffer, R., 2002. Untersuchung der Realisierbarkeit von Geoinformationssystemen mittels des Game-Editors MaxED. Student Thesis, Stuttgart University, Institute for Photogrammetry (ifp), (not published)

Remolar, I., Chover, M., Belmonte, O., Ribelles, J., Rebollo, C., 2002. Real-Time Tree Rendering. Departamento de Lenguajes y Sistemas Informaticos, Universitat Jaume I, Technical Report DLSI 01/03/2002, Castellon, Spain.

Roettger, S., Heidrich, W. Slusallek, Ph and Seidel, H.-P., 1998. Real-Time Generation of Continuous Levels of Detail for Height Fields. In: Proceedings WSCG '98, pp. 315-322.

Roettger, S., 2004. libMini Terrain Rendering Library. http://wwwvis.informatik.uni-stuttgart.de/~roettger, 2004.

SpeedTree, 2004. SpeedTree. http://www.idvinc.com, 2004.

Torque, 2004. Torque Game Engine SDK. http://www.garagegames.com/pg/product/view.php?id=1, 2004.

Unreal, 2004. Unreal Engine 2 Runtime. http://udn.epicgames.com/Main/WebHome