

3D Building Visualisation – Outdoor and Indoor Applications

DIETER FRITSCH, Institute for Photogrammetry (ifp), University of Stuttgart

ABSTRACT

Visualisation of spatial 3D objects is a general task of nearly all science and engineering disciplines to easily assess complex object features, object grouping, environmental goodness-of-fits and many other applications. Photogrammetry as 3D object data provider profits from the progress made in scientific visualisation during the last decade, especially from algorithmisation, resulting software packages and fast hardware implementations. Complementary to scientific and industrial product visualisation a new field evolved triggered by the computer game industry: the development and application of game engines used by our children, spare time and professional computer game players.

The paper aims at a combination of both worlds with texture mapping as a key issue. Separate workflows for outdoor and indoor 3D building visualisation are defined and explained by some examples. It is interesting to note common approaches of the two “visualisation worlds”, even if differences result from dissimilar perspectives. In future, research ought to be directed to interfaces linking outdoor and indoor visualisation in streaming modes to overcome gaps of data and scene transmission within hand-shaking areas.

1. INTRODUCTION

1.1. What About Visualisation?

Since the introduction of computer graphics (~ 1950) the demands for visualisation techniques have grown continuously. Today, the visualisation of three-dimensional scenarios and worlds seems to be a pretentious task requested by many geo-related disciplines, among others also photogrammetry. Tremendous progress to visualize spatial objects can be observed providing 3D perspectives, interactive VRML walk-throughs, high performance rendering and CAVE environments. The corresponding hardware and software to run any visualisation algorithm is equally diverse as the application itself: nowadays PCs, PDAs, graphic workstations, high end rendering machines and high performance computers are used to solve visualisation problems of all kind. This has led to a new discipline called *Scientific Visualisation* (B.H. McCormick et al., 1987). This young discipline offers algorithms, software packages and advanced interactive tools (such as data gloves and other haptic interfaces). An introduction to scientific visualisation is given by U. Lang et al. (2003, this proceedings).

Complementary to scientific visualisation the movie and computer game industry has developed game engines with amazing computer graphics capabilities since the 1990s. We are wondering about the performance of standard graphic boards and chips - millions of polygons enriched with texture maps seem sometimes to be more advanced than scientific visualisation environments. Big investments are being made to implement latest math formulas, to increase the performance of graphic boards and to offer more and more photorealistic scenes to game players. Game engines as powerful software packages use rendering pipelines and special data structures to visualize objects, scenes and 3D worlds in real-time. This means at least 30 and even more frames per second and a continuous (streaming) walk through spatial rooms, buildings and scenes, like a movie. The only difference to movies is the image sequence – this is not fixed beforehand but the user decides individually what to see and in which direction to move. The overall question is: how can we make best profit of both *visualisation worlds*?

If there is any chance to link scientific visualisation with computer game engines – we should try! However, we should also ask the question: is scientific visualisation still an issue for daily

workflows of photogrammetric service providers, companies and vendors? The general answer is *No!* Only a few projects use sophisticated hardware and software. Therefore the introduction above has to be matched with reality of photogrammetric projects, data processing and workflows. For many 3D mapping applications only general purpose hardware and software is available: a standard Personal Computer (Pentium IV, >2GHz, >60GB harddisk, 32 MB graphic board, any Internet browser) with a high resolution screen – we call this *poor man's visualisation environment (PoMaViEn)*, and not a high end computer graphic rendering machine or even a CAVE. We, the daily users of computer graphics are aiming at high quality visualisations at low costs – to maximize our performance/investment ratio (PIR).

Summarizing the lines above simply leads to the following recommendations:

- Use standards for photogrammetric graphics generation (HTML, XML, VRML, Java, etc.)
- Simplify the workflows for textured 3D building models (indoor, outdoor)
- What about game engines?
- What about Computer Aided Facility Management (CAFM) techniques?

The following chapters demonstrate the performance of simplified processes for fully textured outdoor and indoor building elements. It is out of question that these methods are not yet optimized according to latest software technologies – the mission of the paper is directed towards the methods used and not optimizing computing speed and hard disk space.

1.2. 3D Building Reconstruction

At the Institute for Photogrammetry, Stuttgart University, research was directed since the mid 1990s to fully automatically reconstruct 3D city models. Two PhD theses solved the problem: B. Ameri (2000) introduced a three-step workflow consisting of recognition, reconstruction and hypothesis verification using stereoscopic imagery (briefly described in D. Fritsch, 1999). In other words: a method was developed for automatic 3D reconstruction of polyhedral-like objects in the context of a generic building model. A boundary representation of a coarse building hypothesis is constructed in a data-driven, bottom-up approach, from simple geometric primitives (2D plane-roof regions) in image space, to a more complex geometric model (3D plane-roof structure) at a higher level (in object space). Finally a hypothesis model verification is performed in a top-down approach by back-projecting the reconstructed model to the corresponding aerial images.

C. Brenner (2000) reconstructs fully automatically 3D city models using dense LIDAR data in combination with a sophisticated math formulation for complex roof structures. The approach feeds ground plan information of buildings into the reconstruction process to be decomposed into rectangular elements for which corresponding roof structures are computed in a least-squares process. The approach is most advanced and well-suited for complex 3D building ensembles as often given in old and big cities. Parallel to the automatic workflow he introduced an interactive building editor to eliminate erroneous reconstructed buildings.

For the next chapters it is assumed that the geometry of 3D building hulls is available. It is out of question that 3D buildings can manually and semi-automatically reconstructed using photogrammetric stereo restitution processes, this is not part of the paper here.

2. OUTDOOR VISUALISATION OF BUILDING ELEMENTS

2.1 Texture Mapping of Outdoor Building Elements

A textured 3D model for outdoor visualisation basically consists of the definition of its geometry – the convex hull of a building – and one or multiple textures mapped onto the surface of the building elements. The geometry is reconstructed through geometric modeling methods or any other virtual model. To get a fully textured building model the convex hull is decomposed into its boundary elements generating several data files which contain the building's topology and geometry (coordinates). It seems most appropriate to use the boundary representation method (BREP) as successfully applied in CAD (A. Meier, 1986) and also in GIS (D. Fritsch, 1990).

Texture can be generated using artificial patterns modeled in a software package and digital still videos of the original building, respectively. Aerial imagery provides generally texture for building roofs. The mapping process here is fully automatic using interior and exterior orientation as input of the collinearity equations.

In the following a five step method is introduced which was implemented by J. Hekeler (2000) and proven with some examples. Using digital still videos of the original building these images are distorted and have to be rectified to provide homogeneous image scales. For reasons of simplicity a plane is used as rectification surface neglecting interior and exterior orientation of the still video image.

The rectification process therefore needs only 4 control points of a boundary element. Image coordinates are measured in the original still video. To get object coordinates the 3D boundary element is transformed into a XY plane ($Z=0$). In other words: a two-dimensional coordinate system is attached to the 3D boundary element, and its corner points are transformed into this reference system. The final rectification process is fed with the computed surface coordinates of the corner points and measured image coordinates. Finally, the texture mapping process uses the rectified still video images.

In order to complete texture mapping the individual boundary element has to be matched to its corresponding texture. Texture coordinates (texels) provide this matching process. Here, every corner point of the boundary element is matched with a texture point of the rectified image. The computation of texels can be implemented in a software module due to the relations of a corner point and texture point within the rectification process.

Once the textures for all boundary elements are generated the textured 3D building model is complete. The result is visualized using a standard VRML viewer. Summarizing the five steps described before delivers the following workflow:

1. Generation of a geometric model
2. 3D boundary element transformation in surface coordinates
3. Rectification of the still video image
4. Computation of texture coordinates (texels)
5. Export in VRML 2.0

Fig. 1 overviews this workflow of the texture mapping process for outdoor building elements. Emphasis is put here to simplify the processes for fully textured 3D building models.

Nowadays, many software packages offer methods for rectification and texture mapping, but very often the handling is time-consuming and circumstantial. Therefore J. Hekeler (2000) performed with his Master Thesis well offering an own software package capable to run efficiently all five steps given above. Programming was made in MS Visual C++ 5.0, a hardware-driven programming language for fast computations as required in rectification and texture matching steps.

In some cases the 3D geometry of the building or building elements has to be generalized to overcome object details or simply to neglect some geometric information. This is under investigation by the paper of M. Kada (2003, this proceedings). The texture mapping process always compensates a loss of geometric details and is therefore very helpful for photorealistic visualisations.

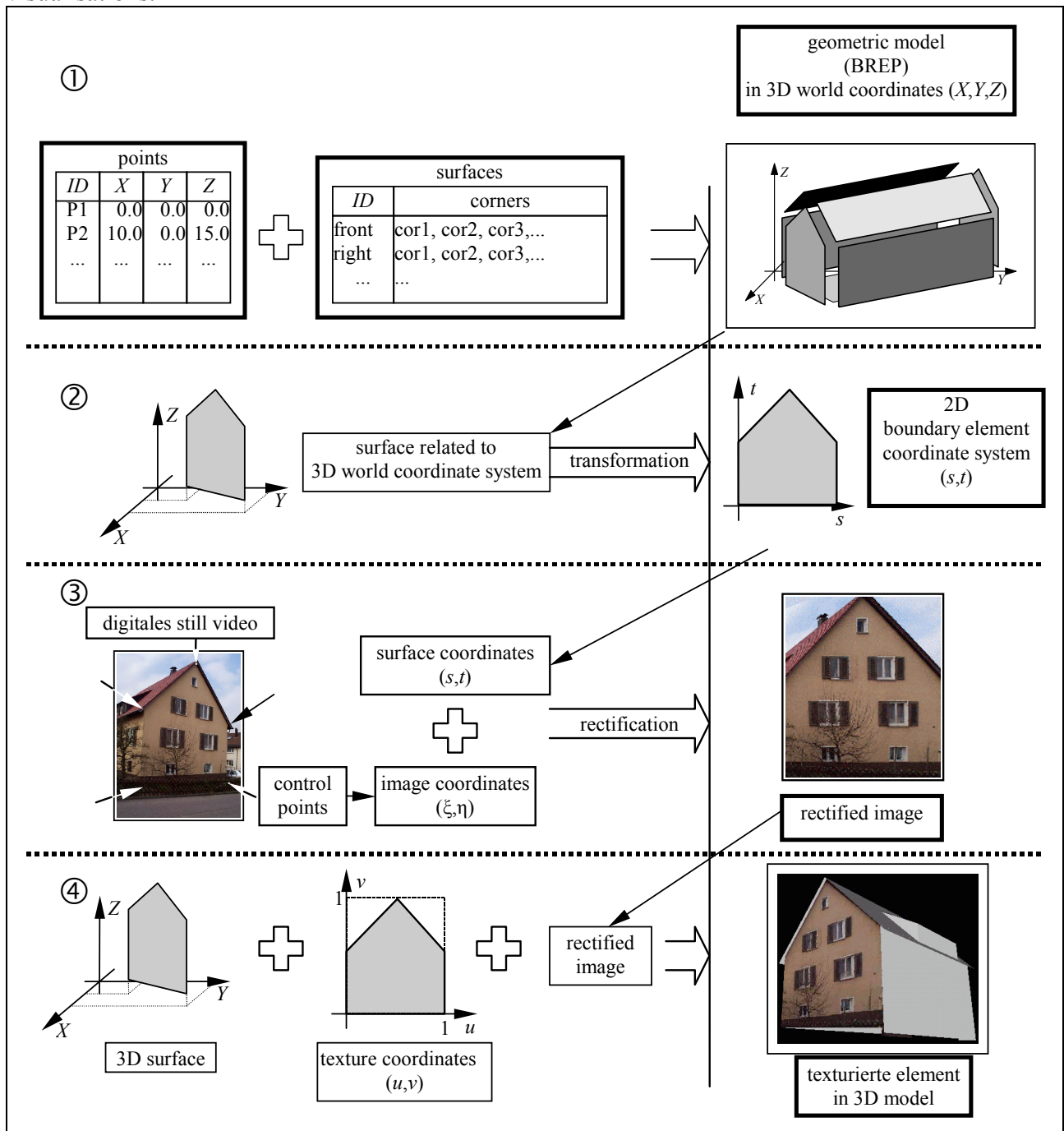


Fig. 1: Workflow of texture mapping for outdoor building elements (J. Hekeler, 2000)

2.2. VRML 2.0

VRML 2.0 (Virtual Reality Modeling Language) is a description language for 3D objects with and without texture. Several objects are joined to form a *scene*, complex scenes are called *worlds*. Scene

objects can be animated and attached with multimedia effects (audio, videos, etc.). The user moves interactively through a scene and can immediately force an object to rotate, to translate and to glide perspectively.

Links to further VRML scenes can be set easily to allow for comprehensive virtual reality models. Moreover, so-called *scripts* can be attached to the VRML 2.0 data file to be realized by Java and JavaScript program modules. Herewith sequences are controlled during the visualisation process (D. Nadeau et al., 1996).

The 3D objects are defined by a simple ASCII data file. Therefore *nodes* and *arrays* are used to describe simple objects or object parts. The processing and visualisation provides an additional program, the VRML viewer or VRML browser. During VRML visualisation free navigation is supported in real-time.

VRML data files are compact and independent of the operating system, therefore they are ideally suited for dissemination via Internet. Several VRML viewers are free-of-charge plug-ins for standard Internet browsers (e.g. CosmoPlayer for Netscape) – also stand-alone modules are available.

Due to its simplicity and compactness VRML is accepted since 1997 as ISO standard graphic exchange data format. In the following the VRML advantages are briefly summarized:

- ISO standard graphic data format
- Fast computation and visualisation capabilities
- Integration of artificial and real texture possible
- Operating system independent and therefore Internet capability.

Fig. 2 presents the Photogrammetric Week Building (*Kollegiengebäude II*) in a VRML viewer. The blue skies have been generated in the software package 3D StudioMax using its VRML editor for the conversion process (R. Pfeiffer, 2002).



Fig. 2: VRML visualisation of the Stuttgart University City Campus (R. Pfeiffer, 2002)

3. INDOOR VISUALISATION OF BUILDING ELEMENTS

3.1. General Remarks to Computer Games

As stated in the introduction, parallel to the evolution of the young discipline *scientific visualisation*, producers of animated movies and computer games have in the meantime gained a leading role in 3D visualisation. Their products run on standard PCs with simple graphic boards, what means on PoMaViEn devices.

Today, 3D computer games – in this paper reference is made to Quake III Arena (ref. to Quake Style) – consist of a game engine, individual game rules and game data (geometry and texture, see fig. 3).

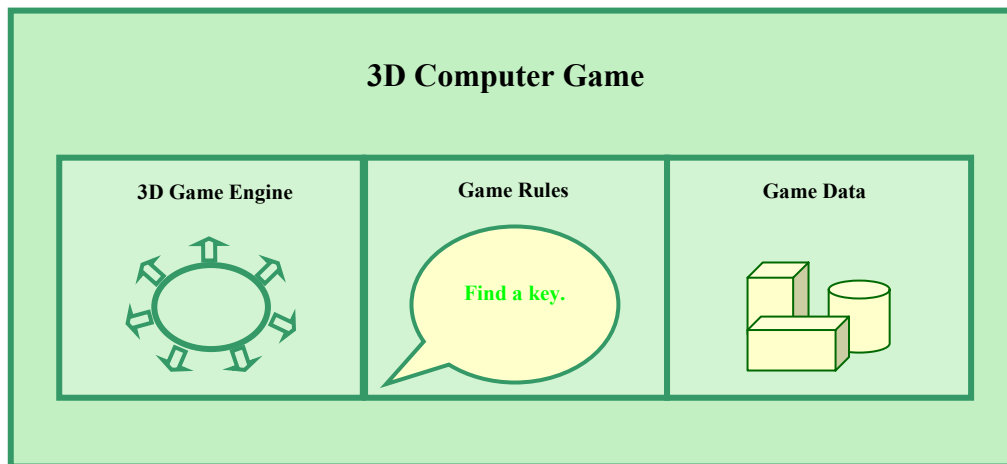


Fig. 3: Components of a 3D computer game (M. Beck, 2002)

Main emphasis is put on the game engine. This module is the *heart* of a computer game and represents the basic framework independent of the game. This general purpose feature allows to use the engine for other games, for example *indoor visualisation of building elements*

John Carmack, Chief Designer of id Software, made game engines very popular. A game engine is some sort of machine to visualize 3D worlds in real-time, using basic elements of a game which are not to be programmed over and over again. Today a game engine is a very complex software system containing many elements and which is also capable of distributed processing (through a network, see fig. 4).

Game physics contains the realization of the Laws of Nature (LoN). Implementing well-defined algorithms allows for the exact simulation of game objects. If a player wants to shoot to a object the ballistic laws are driving the shot. Object crashes, for example, have to be observed and to avoided by simply translate and rotate the crashing objects.

The *Graphical User Interface (GUI)* defines the devices the user interacts with the game, for example a keyboard, keypad or mouse.

Artificial Intelligence is implemented to make a game more interesting. Herewith computer enemies are controlled such, that certain human behaviour is visible. Using routing algorithms objects can overcome a path from A to B.

Network capability allows for distributed game environments. Several players can simultaneously play (1000 and more) the same game. Artificial players can be substituted by real players and vice versa.

The *SoundEngine* provides an appropriate sound backdrop. Here playing noise and game music are mixed to let the player assume to belong to a real team acting as in a real world

The *EventEngine* defines events, which have to be brought up depending on the reaction of a player. For example, if a player comes near a door, the door should automatically open.

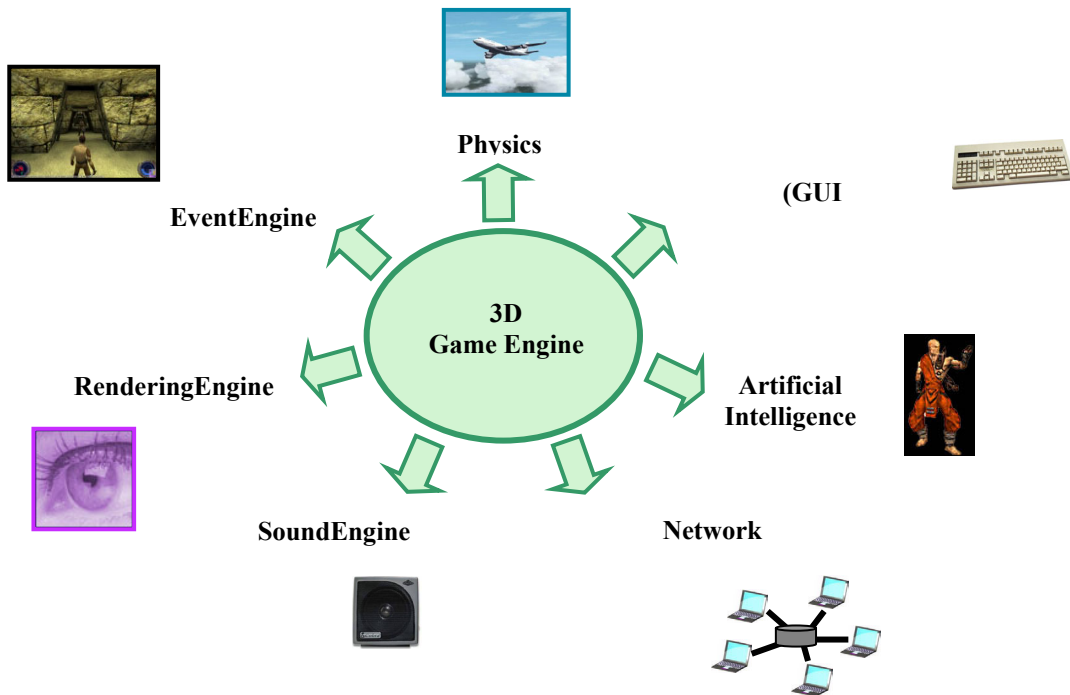


Fig. 4: Components of a game engine (M. Beck, 2002)

Using the game engine and rendering engine for advanced texture mapping is a more detailed process to be described later in this paper.

3.2. Quake III Arena

The computer game Quake III Arena is a product of the id software company. It is the recent version of the Quake generation (Quake 1996 and Quake II, 1997) – it is available since 1999 (Q3Arena.com). Quake is an *Action Game*, also called a *3D Shooter* or *First Person Shooter*. The player moves around and fights by means of several weapons within a 3D world. In single player mode the player fights against so-called *Bots*, enemies controlled by the computer. In multi player mode the player can fight with up to 64 players, which can be located around the world and connected via Internet.

The main difference to other game engines is the quality of graphics and the options for individual design of 3D worlds. Quake III Arena allows for the first time for transparent (windows) and curved surfaces. Moreover, Quake III Arena codes texture with 32 bit instead the 16 bit of its predecessor. The main progress, however, is swapping of the player logics into a dll file and its publication. As known from the MS Windows dll files represent binary programs which are linked during the runtime mode – for the game they allow for additional functions.

The source code of Quake III consists of about 115 files. The function of the files was identified and therefore available for new textures taken from the interior of the Institute for Photogrammetry (ifp), Stuttgart University.

3.3. Q3Radiant Editor

Any geometric modification within Quake III is performed using the Q3Radiant Editor. This free software allows for the construction of new 3D worlds. The editor runs only under MS Windows operating systems and is the interface for data exports of all kind.

Textures are stored as so-called *targa* in *.tga or .jpg format. Only those textures are accepted having a width and height of multiples of 2 (8, 16, 32, ...).

A Q3Radiant Editor generated 3D map is stored in the *.map format. This is an ASCII file which cannot directly read by Quake III and has to be compiled. The compilation process is given in fig. 5

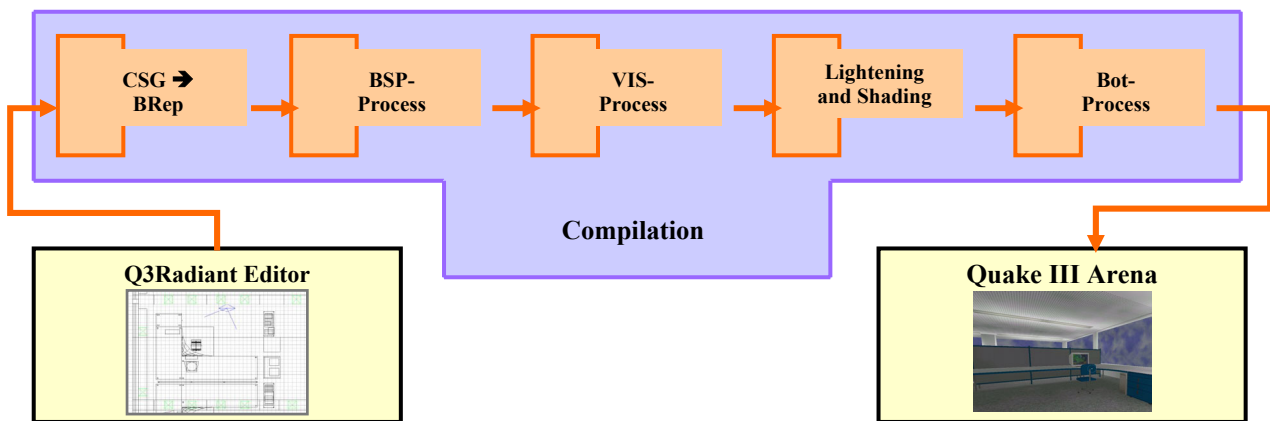


Fig. 5: Compilation in Quake III (M. Beck, 2002)

3.4. Implementing a 3D Digital Textured Map

In order to feed Quake III with own 3D textured data the rooms of the Institute for Photogrammetry (ifp), Stuttgart University were digitized and reconstructed in 3D. Artificial and real textures were used to complement the 3D model. All necessary steps are demonstrated by fig. 6.

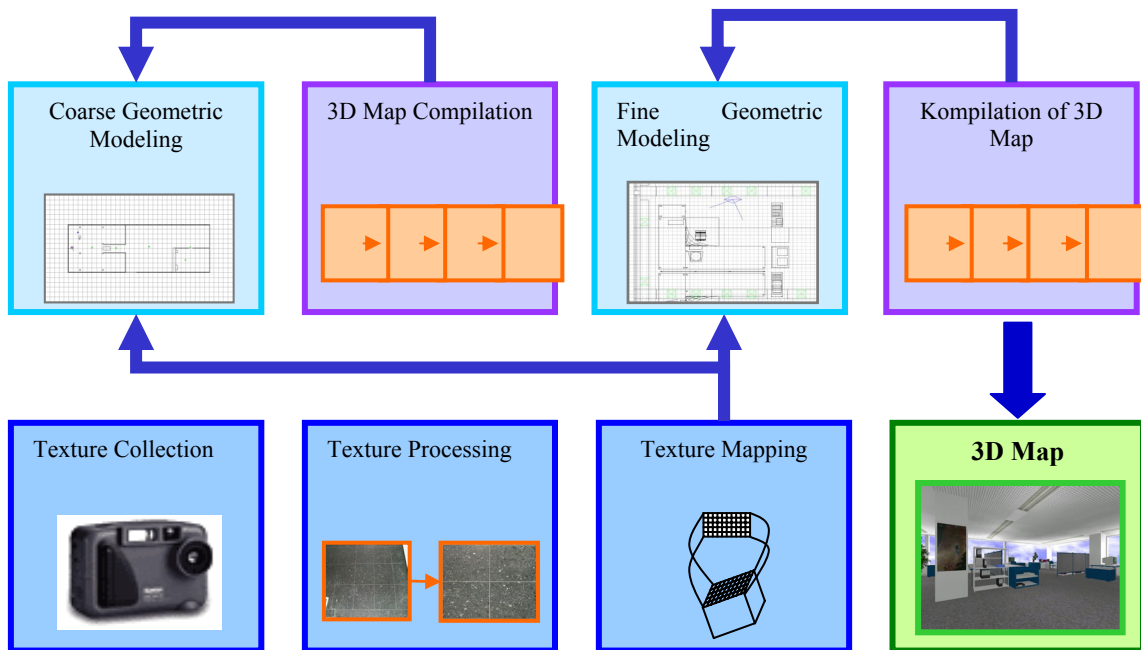


Fig. 6: Workflow for generating 3D texture maps in Quake III Arena (M. Beck, 2002).

4. CONCLUSION AND OUTLOOK

The paper started with general remarks on scientific visualisation, computer game engines and photogrammetric computer environments. The primary role of photogrammetry as 3D spatial data engine was highlighted. It is out of question that photogrammetric methods are reliable, accurate and complete as far as 3D geometric data deliveries and object reconstructions are concerned. Texture mapping becomes an issue when photorealistic visualisations are desired, no matter which mode is desired: still or streaming. In between of still and interactive (human controlled) streaming modes are movie-like virtual walk-throughs – rendered image sequences resulting from a fixed path set beforehand by the user.

The methods introduced in this paper aimed at visualisation methods in PC environments (PoMaViEn) for daily use in simulation, animation and visualisation of textured 3D building models. VRML 2.0 can play an important role because of its easy-to-use capability and real-time navigation option.

In future the user is more interested in random walk-throughs to be generated by graphic engines. These engines are fed by photorealistic 3D city and building models. Softcopy VRML versions are first steps for outdoor graphic engines, though game engines for indoor visualisations are much more advanced as demonstrated with QUAKEIII. Future research will show, how to profit from the extensive development work in the computer game area.

5. ACKNOWLEDGEMENTS

The author thanks Martin Kada for his support in indoor mapping and visualisation using game engines. Also the contributions of Jan Hekeler, Michal Beck and René Pfeiffer, who did excellent development work in their Master Theses to implement outdoor and indoor visualisation algorithms, are gratefully acknowledged.

REFERENCES

- Ameri, B. (2000): Automatic Recognition and 3D Reconstruction of Buildings through Computer Vision and Digital Photogrammetry. Deutsche Geodätische Kommission, Reihe C, No. 526, Munich.
- Beck, M. (2002): Realisierung eines Geoinformationssystems – Visualisierung und Analysefunktionalität mit einer 3D Engine. Master Thesis, Stuttgart University, Institute for Photogrammetry (ifp), (not published)
- Brenner, C. (2000): Dreidimensionale Gebäuderekonstruktion aus digitalen Oberflächenmodellen und Grundrissen. Deutsche Geodätische Kommission, Reihe C, No. 530, Munich.
- McCormick, B.H., DeFanti, T.A., Brown, M.D. (Eds. 1987): Visualisation in Scientific Computing. Computer Graphics, Vol. 21.
- Fritsch, D. (1990): Digital Cartography as Basis of Geographic Information Systems. Proceed. EUROCATO VIII, Palma de Mallorca.
- Fritsch, D. (1999): Virtual Cities and Landscape Models – What has Photogrammetry to Offer? In: Photogrammetric Week '01, Eds. D. Fritsch/R. Spiller, Wichmann, Heidelberg, pp. 3-14.

