An Evaluation of Contemporary Game Engines

Beginnings of a Comparative Study

Tanja Koehler ¹, Andreas Dieckmann ², Peter Russell ³
RWTH Aachen University, Germany
http://caad.arch.rwth-aachen.de/
¹ tanjakoehler@arcor.de, ² dieckmann@caad.arch.rwth-aachen.de, ³ russell@caad.arch.rwth-aachen.de

Abstract. The paper deals with a close examination of several contemporary game engines regarding their usability for architectural visualisation that was conducted by third to fifth year students in a seminar context. The study examines the technology of these game engines with the main focus on graphic quality and adaptability to architectural purpose. The evaluation included not only the workflow of importing the 3D model into the engine but also general handling of the software and its user interface. The documentation of the test process and the evaluation of the various engines have been documented in a wiki compiled and written by the students. The information therein and the cost-benefit analyses provided for every engine should assist future users to quickly find the most suitable game engine for their specific purposes.

Keywords: game engine; visualization; virtual reality; real-time graphic; 3d model.

Introduction

Game engines, already highly developed with all features also needed for real time architectural visualisation, provide a useful development environment for high quality interactive architectural presentations. They allow the architect to use the existing technology and environment of the game engine to import his own 3D models of buildings or architectural structures with realistic textures and present them in an interactive walk-through from outside as well as from inside, even placed in a virtual urban context. Compared to conventional 2D renderings, the whole building or structure can be understood very easily. It is even possible to change geometry or textures in real time. This can improve the communication between architect and client significantly, especially in multi-user environments.

Game engines also provide useful additional features like different light or weather settings, integrated sounds, animation of objects, use of avatars or network based implementation of virtual objects. Equipped with these opportunities, game engines can be powerful tools not only for presenting virtual architectural buildings or structures, but also for developing them interactively and simultaneously with
The use of game engines in architectural education has been a popular research subject since the 1990s. Several engines such as Unreal (Pickersgill, 2007), Quake (Hoon and Kehoe, 2003), Torque (Moloney, 2002) etc. have been used, with varying levels of success, in a design studio context. Additionally, game engines have been examined as to their suitability for collaborative design on several occasions. However, the reasons why a specific engine was chosen for a certain project are often not offered in full detail. To the authors’ knowledge, there is so far no comparative study of game engines regarding their aptness for real-time architectural visualization. This paper attempts to start to bridge that gap.

Examined game engines

The study that this paper is based on was conducted in a seminar context by third to fifth year architecture students. The students were asked to form teams of two students each. Every team could choose a game engine to examine from a number of preselected engines.

In total, four game engines were examined, among them one game engine of a contemporary computer game (Oblivion) as well as an open source 3D engine (Irrlicht) and two commercial graphic engines (Quest 3D, Nova 2008). In order to guarantee the comparability of the study, all engines were tested with the same architectural 3D model (3D Studio Max, Autodesk). This specific 3D model had a complex geometry and contained various textures and maps.

The engines were chosen according to several criteria. First of all, the engines should be able to import .3ds-files or support an exporter which allows the use of those files. Another important point was the availability of documentation and support in terms of online forums or communities. The students should be able to get acquainted with their specific engine easily by themselves in a very short time. As the study was conducted by architecture students with rudimentary knowledge of programming languages, engines containing editors or graphical user interfaces were preferred. Another aspect was the availability of the software and the expense factor. The engines should be either open source or the availability of a trial or student version should be given. Last but not least, the engine should be state-of-the-art in terms of graphical aspects.

As the focus of the study lay in the visualization of a static model, the engines were not examined regarding their capability concerning character animation or physical behavior. In the following, all four engines are introduced with their main aspects.

Irrlicht (Version 1.4, 2007)

Irrlicht (figure 1) is an open source high performance real-time 3D engine, developed by Nikolaus Gebhardt. Written in C++, it is running on Windows PC, Mac OS, Linux and Sun Solaris and supports six different rendering APIs (Direct3D 8.1, Direct3D 9.0, OpenGL 2.0, Irrlicht Engine software renderer, Apfelbaum Software Renderer and a null device). For using the engine a C++ compiler (e.g. Microsoft Visual Studio) is compulsive.

The engine integrates numerous graphic features like dynamic shadows, customizable particle systems, realistic water surfaces or vertex and pixel shader support.

Irrlicht can be used together with IrrEdit, a

Figure 1
3D-model in Irrlicht engine
realtime 3D world editor and radiosity lightmap generator. IrrEdit supports all common 3D filetypes. Besides irrEdit there is also IrrKlang, a powerful sound engine and audio library.

As usual for open source projects, Irrlicht has a large online community and various forums. Additionally, an extensive documentation is available.

**Oblivion/Gamebryo-Engine (The Elder Scrolls IV: Oblivion, 2006)**

Oblivion (figure 2) is the fourth part of ‘The Elder Scrolls’ series, the single player role-playing game by Bethesda Softworks, developed for Windows PC and Xbox 360. Mainly because of its outstanding graphics and the immersive environment the game has gone down well with critics, winning numerous awards and scoring best marks at computer game rankings.

The game is based on the gamebryo-engine (Emergent Game Technologies), which has been used in more than 200 computer games so far, among them titles like ‘Sid Meier’s Civilization IV’, ‘Gothic III’ or ‘The Dark Age of Camelot’ series. For Oblivion, the engine has been strongly modified and adapted to the specific needs of the game.

In addition to the game Bethesda Softworks provides an editor (The Elder Scrolls IV Construction Set) in order to allow players to create their own ingame-graphics. The editor is freeware and was used for the examination.

**Quest3D (Educational VR Edition 3.5.2, 2007)**

Quest 3D (figure 3), created by Act-3D Software, is a professional 3D development environment for generating high-end real-time 3D applications. The network-compatible software runs on the Microsoft Windows platform with DirectX 9.0. The applications developed with Quest3D can be saved as Windows executables (*.exe) or can be integrated in other programs. The software works with program modules (templates) which makes the need of using a programming language unnecessary.

With its graphical user interface it is possible to develop in real-time (WYSIWYG) without the need of using any programming language although scripting (LUA) is optional. The software supports numerous graphic features like vertex and pixel shader, dynamic lighting, special effects such as fog, particles, a powerful physics engine or real-time shadows.

Quest3D comes with an extensive documentation and tutorials and in addition Act-3D also provides an online forum and support team to assist the user community.

**Nova 2008 (5.9.0.8 PLE, 2007)**

Nova 2008 (figure 4), developed by Vertice, is a quite new real-time 3D engine which is created for the use directly in 3D Studio Max. It is fully integrated in this software and comes along with all parameters necessary for generating high-end real-time visualization.
and an Explorer to show the 3D-scene in real-time.

The engine supports DirectX 9.0 as well as all hardware functionalities of the latest generation of graphics boards (e.g. shaders 3.0). In terms of rendering, Nova 2008 has numerous features like real time shadows, reflexion and refraction, vertex and pixel shader, volumetric fog as well as a powerful physics engine.

Examination

The examination of the engines was structured as below:

- workflow (import of geometry, texturing, lighting...)
- testing (geometry, mapping, lights, features, handling...)
- problems and difficulties
- conclusion

Irrespective of the engine, the 3D model had to be prepared for the import into the engine. In our case, the 3D model was a high detailed version of a building at the RWTH Aachen University Campus. The original 3ds-file contained more than one million polygons arranged in lots of different objects, sub-objects and groups. As not every engine is able to handle models containing that many polygons (for instance Quest3D limits the polygon-amount of one simple object to 65,000 polygons) the 3D model had to be adapted to the specific requirements of each engine. While Nova 2008 is working directly in 3D Studio Max and Irrlicht is able to import 3ds-files, the two other engines (Oblivion, Quest3D) need an exporter to transform the 3ds-format into a readable file format. These exporters usually are freeware (Nif-tools for Oblivion) or come along directly with the engine (Quest3D).

The materials and textures can be either mapped in 3D Studio Max or directly in the engine. But not all engines offer this possibility. Especially the ‘real’ computer game engines like Oblivion often only have limited functionality in their Software Development Kits. In many cases it is not possible to change the geometry or mappings of imported objects.

The graphic quality and the support of the various map types and shaders have a high impact on the realistic appearance of the rendered model. Therefore the main focus regarding the mapping lay on alpha, shadow, bump and normal maps. Except for Oblivion all engines support those common maps and some even more (light map, height map). Oblivion only provides three different mapping types: color map (diffuse), normal map and glow map. For the other engines the number of mapping channels vary from four (Irrlicht) to nine (Quest3D).

The best file format for textures is .dds (Direct Draw Surface) because it is very efficient in storage consumption and supports alpha and mipmapming. Additionally, only textures of the powers of two should be used; for some engines (Quest3D) those textures even need to be quadratic.

The use of lighting can vary greatly. Some engines allow the import of external lights, some do not (Quest3D, Irrlicht, Oblivion). In that case the lights have to be set in the editor or directly in the engine itself. Especially the engines of computer games usually have an own lighting-system, which can hardly be changed. Furthermore, the Oblivion engine does not support real time shadows for static objects.

Cameras can be placed either in 3D Studio Max or in the engine/editor. The cameras define the way the user can walk through the 3D scene. Oblivion always uses the ingame camera, which is connected to
the character. It is switchable between third-person and first-person view.

For creating satisfying real-time visualization there might be the need of interactivity. Especially for architectural visualization it would be helpful if certain objects could interact with the user, e.g. animated doors or elevators. All engines, except for Oblivion, support the import of animated objects and the interaction with them.

Professional game engines (Quest3D, Nova2008) also support the possibility of adapting the graphical user interface. Own logos can be integrated as well as e.g. buttons for interactivity.

Last but not least, the engines were examined regarding their handling. A suitable engine should not only be easy to handle but provide also the possibility of a user-friendly presentation. That means an intuitive way of controlling the walk-through in the visualized world. As presentations in Oblivion are always running in the gaming environment the controlling is as easy as playing the game. The other engines can be controlled with default tools and keys or adaptations thereof.

The results of the examinations were collected in the faculty wiki (figure 5). For the purpose of comparability, the students had to make use of a document template specifically designed for the seminar to present their findings in a structured manner. Among other items, the template features an info box at the top of the page that contains a summary of the most important features of the examined engine and ratings for certain aspects of the engine (usability, time consumption, GUI, graphical quality, support etc.).

**Conclusion**

Four different types of engines have been examined: Oblivion represents the group of computer game engines, Quest3D is a program module engine, Irrlicht has no graphical user interface and stands for the program language based engines and Nova 2008 is an engine fully integrated in a common 3D software.

All these engine-types have their advantages and disadvantages depending on the user and his individual aims regarding real time visualization. This examination targets the group of users with low experience in game engines and programming languages but with high ambitions for creating high-end architectural visualization, most prominently architecture students who need a rather time-efficient tool to properly present a 3D-model in a studio context. For this target group the following conclusion should be helpful to choose the most suitable engine for their purpose.

3D engines like Irrlicht (or e.g. OGRE) that use a programming language instead of program modules are difficult to handle. Even if they provide a 3D editor, the knowledge of the specific programming language is still needed. Regarding architectural visualization this might require the collaboration of an architect and a programmer. From our point of view, those kinds of engines therefore are not recommendable for architectural visualization. Compared to the other examined engines, it took the students much more time to see acceptable visualization results because they first had to become acquainted with C++ and the language based interface of the engine (use of a compiler).
Game engines of computer games (e.g. Oblivion) usually can be used very easy for real-time visualization. Especially First-Person Shooters (Doom, Half-Life, Quake, Unreal) come along with Software Development Kits or open source code in order to allow players to adapt the game to their ideas (Mods). As this kind of modding is very popular, there are lots of tutorials, forums and communities dealing with this topic. Concerning financial aspects, those kinds of game engines are very beneficial as the user has only to pay the price of a single game and no additionally license fees.

Normally, the game’s settings (time, landscape…) and lights are unchangeable. If the game takes place in a medieval world (Oblivion) with very atmospheric lighting, this might correlate with the established idea of modern architectural visualization. Furthermore, the graphical user interface is not always adaptable; in case of Oblivion the icons (fist, energy bar) are not removable. For presentation purposes it might also be a problem that the real-time visualization runs only in the game, writing a separate .exe-file is not possible.

Those kinds of game engines can be an easy start into the topic of architectural real-time visualization. They are easy to handle and offer a complete world with full landscape and light system where the 3D model easily can be placed into.

More experienced users might, however, meet the limits regarding adaptability, animation and interaction of these engines very soon.

Professional real-time 3D engines which are based on program modules like Quest3D are very user-friendly and offer a great possibility for high-end real-time visualization. Due to its 3D Studio Max plug-in the Nova 2008 engine takes an exceptional position among the professional real-time game engines. A user already experienced in 3D Studio Max can easily create a real-time 3D-scene without any knowledge of programming or learning new software.

Usually those kinds of engines (Quest3D, Nova 2008, Virtools,…) support various map types and provide a wide range of shaders. The visualization can be very realistic, as all constraints (light, landscape, cameras…) are adaptable to the user’s requirements. For more experienced users, scripting (e.g. LUA) is normally also supported. All these advantages have their price, those engines are very expensive and sometimes even demand license fees for the created products.

Both engines (Quest3D, Nova 2008) offer demo versions, but while the Quest3D demo version has no enabled saving functions, the Nova 2008 demo version (Personal Learning Edition) always displays the Nova logo. In addition, it is possible to purchase a Quest3D Educational version (prices 2008: Educational VR version around 430 Euro).

Further works

With all four engines the students were able to achieve graphically appealing results. This examination is just the first step into real-time architectural visualization, but there are still some more engines on the market which should be examined, most prominently some of the popular first person-shooters like Unreal, Quake, Doom, Source and FarCry, but also a few open source engines. This will be done in the coming semesters.

Especially the multi user environments of some engines will be examined regarding to their usability for collaborative design. In a previous seminar, students have worked with the open source engine Sauerbraten which has excellent cooperative editing capabilities with instant visual feedback, although the import options leave much to be desired.

Also a collaboration between architectural students and students of computer science might be very interesting: a programmable 3D engine could be adapted to the needs of architects.
References


