Visualization and teaching with state-of-the-art 3D game technologies

LEHTINEN Saku

Department of Architecture, Helsinki University of Technology, Finland

This paper describes the use of the latest 3D game technologies and game editing tools in architectural modeling and visualization.

The differences to more conventional rendering techniques, the benefits and the disadvantages of using these tools in architectural teaching are discussed.

Keywords: Visualization, real-time graphics, game technologies, virtual environments, interactivity

Background

For some years now, computer gaming has been the part of IT-industry that pushes the envelope on both the fields of software and hardware. Most notably, the rapid development of 3D-acceleration hardware (way above Moore’s law) has been dictated mostly by games.

Computer games have been around for some three decades. Still they are often ignored and categorized as not being a “serious business”. In the year 2001 games’ gross turnover, for the first time, surpassed the movie industry.

Average CAD-software is no match for games in 3D-performance: modern 3D-accelerated game engines are able to render way over 1 million polygons per second with multi layered texturing and other complex surface properties. Most CAD-programs come with limited or no support for 3D-acceleration. For instance, AutoCAD and ArchiCAD are able to render some 10,000-50,000 flat shaded polygons per second in their “real-time” 3D-views.

Naturally it is questionable compare to these two: a CAD-software aims to marvel in completely different areas than a 3D game, but it is still safe to say that a typical CAD-software could be dozens or even hundreds of times faster in its 3D-speed by implementing acceleration support and some game engine technologies. The issue of mere rendering power also raises a question: what working methods would more powerful 3D-technology allow? Would a more immersive, game like design environment be the way of the future, instead of the current way of working, which carries a great burden of concepts and conventions dictated by 80’s or even 70’s technology?

Using game tools in architecture

If CADs are not able to offer good real time 3D graphics, then utilizing game tools and technology in architectural modeling becomes very tempting. The idea is not new: game engines have been tested in this purpose for at least a decade; the only limiting factor has been the ease of use.

The most popular game engine for such experiments has been Unreal technology. Unreal was one of the first games that had the official game editor (UnrealED) available to everybody who owned the game. Making new content with the full power of the engine became relatively easy and Unreal has, like many other games, created a strong subculture: Modifiers or “modders” are individuals who produce new content in their spare time with the tools provided. Most game creators endorse “mods” and allow them to be made as long as no profit is made; mods prolong the lifes-
pan of a game and create more sales. Developers might even provide their proprietary game development tools in the game CD accompanied with tutorial and help files.

Many of these game creation tools are very powerful 3D-modelers/texturizers designed to create new content very efficiently. Unlike conventional 3D-software rendering packages (3DS Max, Maya etc.) they aim for real-time graphics and also offer advanced scripting capabilities to create dynamic (interactive) content and methods for optimizing the content for maximum speed.

**HUT experimenting**

In the past three years HUT / Department of Architecture has been experimenting with teaching the use of game tools on a visualization course. Each year, a select few have tried to become acquainted with 3D-realtime worlds by using a game editor.

The game platform of choice has been Max Payne (2001) engine and it’s level creation tool MaxED. Unlike Unreal, this technology has no multiplayer support which prevents the students from creating VR or a “virtual meeting place” type of environments. On the other hand, MaxED offers a better, more realistic radiosity lighting model, making it more suitable for architectural lighting and especially interior visualization.

**Hands on**

Students show a natural interest in game technologies. The idea of games being a synonym for fun has a clear advantage and teaching with enthusiastic students can be very rewarding.

In many ways, game design tools can provide better immersion and a much more immediate response than any full-featured 3D-package. Most modern rendering and visualization programs are slow and extremely complex; the learning curve with game tools can be much less steep, as the amount of modeling commands are much smaller.

Students also seem to enjoy the immersive approach of the program. With CAD and rendering programs one usually operates much more “outside” the model. MaxED for example, does not even offer the possibility to use the traditional three-axis wireframe view. The whole modeling process is done while moving inside a textured and lit perspective 3D-world in “constant walkthrough”-manner. The movement controls are very intuitive and the students grasp them immediately. All the modeling work is done in relation to the grid plane, which can be freely rotated and aligned. Being constantly inside the environment encourages the students to think about their architecture more as a spatial experience than a schematic plan.

**Technical differences**

The actual modeling, texturing and lighting is relatively easy, but taking the geometry into the game is more difficult. In most cases, just using the editor itself and not taking the model into the game environment is enough for most architectural needs.

In teaching sense, game tools also prepare the students for any 3D-package. Most of the basic tasks remain the same no matter what the platform is. Modeling geometry requires the same logic on how to use simple primitives to create more complex forms.

Maybe the most notable difference from conventional modeling techniques is the extensive use of textures. Traditional rendering software might have good texturing capabilities, but the controls usually appear complex to beginners. Game design tools automatically force textures onto every polygon of the model. This changes the balance from creating exact polygon models to approximated forms, which use textures as suggestions of more complex forms. For students this can be a great help: the textures create a sense of scale and feel of materials right from the start, which helps spatial understanding.
Lighting is another difference: MaxED is able to render a radiosity lighting solution, which can give a very realistic feel to environment. For instance, using cardboard-like texture and radiosity lighting can, at it’s best, provide a very interactive and good “virtual cardboard models” – only lacking the tedious cutting and assembly work of their real-life counterparts [Figure 1].

Balancing geometry, texturing and lighting right from the start helps the students to quickly understand the “high and low frequencies” of the environment: low meaning the basic forms and lighting, high meaning the small surface details provided by the textures.

A clever technique for modeling was also invented during the first year of experimenting: As there is no support for any CAD drawing format; it was hard to replicate the existing environments manually. Creating large correctly scaled high-resolution textures from reference CAD-drawings provided the same usability than actual CAD lines brought into the 3D model. Once textured and correctly matched onto a large surface, the drawing allowed the students to quickly draw on top of the image [Figure 2] and “loft” 3D objects from there – a variant of the technique often used in rendering programs.

Creating only static geometry models and using them as a base of visualizations is naturally forgetting the important aspect of the technology: interactivity. In the previous years, teaching how to create interactive game has not been the goal of the course. This year, some interactive environment creation has been included, and it naturally has shifted the emphasis of the course slightly from the traditional visualization to scripting and animation issues.

Conclusion
The biggest obstacle to taking the game creation tools into general use in architectural visualization is very prosaic: all game creation tools currently lack any easy way to import from or export geometry to any common CAD-software.

In general, these technologies give a preview of things to expect in the standard 3D packages in the future: more real-time, more interactivity and even better WYSIWYG – modeling and rendering do not need to be separate processes, as the adequate quality final visualization is rendered in real-time.

They also offer an interesting, advanced and inexpensive platform to create virtual environments and visualizations, especially in architectural schools where students might find a lot of use for the technology.

Somewhat ambiguously, game editors are most useful at the beginning and in the end of a design project: The sketching phase can benefit
from very quick, interactive and less measuring-oriented design environment, which provides immediate response and a good platform for quick prototyping. On the other hand, at the very end of a project, creating an interactive walkthrough model of the project can be an option to animations or still renderings.

**Examples**

The first experiment with game technology was in the year 2000, when two students created a virtual model of a Senate Square in Helsinki.

In the spring of 2002 two students of Sibelius Academy created a surrealistic interactive model of a Kamppi metro station in Helsinki on a course that was taught by Mathias Fuchs and Sylvia Eckermann.

A group of three architecture students are creating a scene of professor Aarno Ruusuvuori’s WeeGee-house in Tapiola in the summer of 2002. The model will be placed in the lobby of an art gallery in WeeGee-house, and the visitors can play the game to find out more details about the architectural history of the house.

**References**


