Multi-Faceted Architectural Visualization

Neil Shinnors, Neville D’Cruz, and Andrew Marriott
Curtin University of Technology, Western Australia

As well as learning traditional design techniques, students in architecture courses learn how to use powerful workstations with CAD systems, color scanners and laser printers, and software for the rendering, compositing and animating of their designs.

They learn to use raytracing and radiosity rendering systems to provide visual realism, alpha-channel compositing systems to put a client in the picture (literally) or the design in situ, and keyframe animation systems to allow realistic walkthroughs.

Student presentations are now based on videos, photographic slides, slide shows or real time animation. Images (as data files) are imported into full color publishing systems for final year thesis presentation.

The architectural graphics environment at Curtin University facilitates the integration of slide and video examples of raytraced and chroma-keyed images with computer aided design techniques for architectural student presentations.

AEC

The Architectural, Engineering and Construction (AEC) industry is growing at over 12% per year. The majority of architectural firms though employ less than a dozen people. This area is a paradise for PC-based AEC software, even though a PC is so severely limited in terms of operating system, memory, disk space and graphics capabilities!

The workstation, surpassing the PC in the above areas, had a serious drawback - price. (Some may argue that a workstation's lack of AutoCAD was also a major drawback - others argue that AutoCAD for serious AEC work is a non-sequitur.) Low cost integrated work stations by Silicon Graphics, Hewlett-Packard and Sun Microsystems for example, coupled with high quality AEC software are now becoming a more appropriate platform. And with faster microprocessors and Advanced Computing Environment (ACE) systems, Personal Computers have now become Personal Workstations.

3-D designing and rendering let the architect work in a natural manner - accurate 3-D rendering presents a more realistic image to an architect or client than an artist's impression. Badly lit areas or areas with distracting reflections / lighting can be seen before the concrete has set. Techniques such as raytracing and/or radiosity can produce highly realistic images, desirable for client presentations. The realtime wireframe or shaded model capabilities available on a number of platforms have enabled architects to view and manipulate their models in 3-D. Companies such as Autodesk, Intergraph, Eagle, CAD/CAM and others now offer CAD/AEC systems with various levels of rendering performance.

Previously, the renderers integrated with CAD/ AEC systems offered poorer performance and quality than those that could be run standalone with the imported 3-D data. Companies such as Wavefront, Alias Research, Intelligent Light, Neo-Visuals and Cubicomp can import data via translators and then use their superior rendering and animating facilities to provide high quality rendered output. However, the advantage of an integrated system is that the architect simply issues a consistent set of key strokes or mouse button pushes to get an image — he/she does not have to learn a new system with a different interface and often, different aims (systems designed for animators / scientific visualizers).

Partnerships have begun between AEC and rendering companies - Autodesk with the Pixar...
Renderman system, Datacad with Velocity from Circuit Studios and Intergraph with Modelview. The aim is to provide integrated high quality rendering of AEC images for visualization, animation or client presentation. Skidmore, Owings and Merrill in collaboration with IBM has developed Skyline (or AES as marketed by IBM) - this is an entire AEC package that has been described as a computer aided design / geometric access system. It covers aspects of architectural design and construction such as site analysis, finite element analysis, scheduling, facilities management as well as the more typical aspects of design and imaging.

Paralleling the decrease in cost and increase in functionality of desktop publishing systems, the color electronic prepress systems (CEPS) have enabled architectural firms to produce draft (and sometimes final) color images for clients at low cost. CEPS allow the scanning of images for textures / initial design work, the adjusting / separating of images for gamma correction or color balancing via full color publishing software and the production of images on full color laser copiers up to A3 size. Advances in “WYSIWYG” (what you see is what you get) color on screen vs. color on paper” software by companies such as Tektronix, Barco and Fantone has meant fewer cycles in “getting it right”.

The cost of producing video walkthroughs for clients has also dropped as PC based video frame buffers / controllers have proliferated. These typically display 16 million colors at television resolution at broadcast quality. They may also directly control a 1”, Betacam or UMATIC frame by frame video recorder and typically record a frame in 20 to 30 seconds. Compression techniques enable the storing of minutes of high quality animation on a few hundred megabytes of disk. Because of the low cost, bureaus exist for taking frame by frame data in a known format and recording it to tape.

Unfortunately, in Australia, with a few exceptions [6, 14, 15], practicing architects have not embraced the new technology, even though these systems are available. To quote Jon Pittman of Wavefront: “…architects are tremendously price sensitive”. Students in architecture are using powerful workstations with CAD/AEC systems, color scanners and laser writers, and software for rendering, compositing and animating their designs. One hopes that like the waves of Computer Science graduates shouting “Pascal Structured Programming! Good Design!” who emerged from universities in the 70’s and forced employers to adopt modern methods, the new breed of architectural students will bend AEC firms into more realistic practices.

### Modern Architectural Systems

Rendering software has successfully been used in a number of non-computer related “graphic” environments:

- **Textiles** [23, 24] with commercial systems such as CDI and Modadrape,
- **Archaeology** [36, 37, 20, 45]
- **Landscaping/Botanics** [40, 17, 5, 3, 35, 22, 4]

Many of these systems were developed at universities as research or as part of a teaching program. Architectural systems have been developed along the same lines with one of the best examples of a CAD/AEC teaching and research environment being Cornell’s CAAD (computer-aided architectural design) Studio led by Professor Donald Greenberg. He has been behind many important architectural advances in the area of color [26, 31], user interfaces for architectural work [2, 44, 38, 33] and realistic rendering [8, 25, 7]. Similarly, Professor William Mitchell at the Graduate School of Design at Harvard University, has been very active in encouraging students to fully utilize modern computer graphics technology.

Students at Curtin regularly use a 600 dpi color scanner to capture “sites”. Figure 1 shows a student project that was finished in one week. The image is made up of three separate photographs (with different exposures and color corrections thanks to 1 hour processing!) which have been composited and then hue adjusted / painted [18, 19, 42, 39]. The raytraced image has been positioned [13] and then masked and touched up [1]. Figure 2 shows a composited image of a user generated house - a client provides basic parameters such as size of rooms, orientation, adjacency plus cost and a number of typical dwellings (plus an approximate cost-to-build for each) are generated [32]. The chosen CAD model is then custom adjusted, rendered and composited for the client.
Figure 1: Compositing scanned in and computer generated images

Figure 2: Composited, touched-up, user generated house in a scanned in setting

Figure 3 shows a typical interior design setting generated with the Eagle CAD/CAM software and then rendered by the Computer Animation Negus (CAN) Raytrace System[28] via Eagle macros. The scene was rendered with strict attention to camera from, to and view up information as well as field of view. A chroma-key studio was used along with chroma-keyed furniture (and even a staircase in Figure 10) for a client presentation where students sat around the table in a committee meeting scenario[11]. The camera was set up to match the rendered viewing information. The composited scene was produced with Super VHS technology plus an editing suite. The finished video was particularly impressive due to the reflections of the actors on the floor and their ability to ascend the staircase.

Figure 4 shows the same image after being manipulated for a more traditional presentation [18]. Architects expect visualization as well as presentation graphics - what happens in the room as the transparency/reflectivity of the glass doors is changed, and then, what is the "best look" to sell to the client.
Multi-Faceted Architectural Visualization

Figure 3: Interior design

Figure 4: Interior design with artistic license
The images produced may be any size but typically are television resolution for animated pieces and approximately 1024x680 for 35 mm slides. Slide images are photographed from the screen (with adjustments for screen gamma) or can be sent as digital data to companies with color laser printers.

Students use standard design techniques and then use Eagle running on Apollo workstations to create their 3-D CAD model. The CAN Raytrace System is used on the model to create images/animation.

**Raytrace rendering**

The CAN Raytracer can read in a script file which contains model data and attributes and produces an anti-aliased floating point (red, green, blue, alpha) raytraced image. The CAN system uses many techniques reported by [16, 9, 21, 43, 12] to minimize image rendering time and to improve image quality.

The Raytrace user interface is currently menu based and is rather strict (the architects do not see this interface). It allows input and output redirection from and to script files as well as input from forked off processes.

The system allows models made from: ellipsoids, rectangular prisms, triangular patches, parametric smooth curved surfaces (Bezier and Beta 2 splines decomposed into Phong shaded triangular patches), torus bends and reducers. Each object has a Lisp-like property list with attribute/value pairs. Objects can share property lists, and it is even possible to build up a tree-like hierarchy of property inheritance. Properties are color, refractive index, reflectivity, transmissivity, light color, fog color, optical depth and bump map.

Textures can be applied to any object and to any property of that object. The textures are: recursive checker, recursive dot, Mandelbrot, spot light (special effect to simulate a spot light), fuzzy light (special effect to simulate a fuzzy light), cube map, cylinder map, interpolation and flat map.

The source code is in approximately 50 files and is 1.1M bytes over 26000 lines. With comments removed it is 250K bytes and 11000 lines. All procedures are fully documented. Additionally the Raytrace System:

- provides over 70 texture maps for various effects
- provides over 45 predefined textures and script files for object attributes such as brick, grass, gravel, leather, etc.
- can use CSG expressions: objects can be intersected, subtracted or added in any number of expressions
- comes with animation utilities for keyframe parametric motion control [41, 27] as well as a farming mechanism for frames and scanlines within a single frame where multiple machines exist in a network [29]
- uses distributed raytracing to get depth of field effects through specified lenses/apertures [9, 10]
- has a wireframe rendering mode for animation preview
- comes with a set of tutorial notes (100 pages) plus example programs

The Raytracer is part of the CAN environment which includes the Utah Raster Toolkit image processing software, the Silicon Graphics image manipulation software, a model geometry translation system (MOC), a desktop publishing system, the Alias PowerAnimator system, the Wavefront Visualization system, an animation framing system and a keyframe spline interpolation system. The Computer Animation Negus environment is used by students and researchers from the schools of Computer Science, Architecture, Art and Design and Film and Television (hence the name negeus: a strong spicy wine with ingredients from many areas).

The Eagle CAD package enables architects to easily construct virtual models: it does not provide a means for viewing the rendered model in an aesthetically pleasing (or realistic) fashion. Using the CAN Raytracer is neither easy nor intuitive and hence a mechanism for combining the two packages was sought. Previously, rendering an Eagle model required a great deal of human intervention. Figures 5 and 6 show the results of hand tuning the attributes of an early Eagle generated model. Although these two images won the Technical Section of the Art competition at the Ausgraph 90 conference, most architects would not have the knowledge (nor have the time to spend) to create an image of this quality.
Figure 5: Textured, bumped image of foyer of an art gallery

Figure 6: Upstairs view of art gallery with standard CAD human silhouettes
It was decided that the converting of an Eagle model into the CAN Raytracer format must be automated. Thus, the Eagle Automated Rendering System (EARS) was developed.

EARS is comprised of several programs which handle the conversion of an Eagle model into a Raytrace model and which control the rendering process. It also contains a set of Eagle macros and a library of materials which can be used during the building of an Eagle model.

The macros let an architect use the mouse to:

- assign attributes to parts of the model
- convert a model into a Raytrace input file
- render an already created Raytrace input description
- convert and then render a Raytrace image
- display a (partially) rendered image on the screen
- query how much of an image has been generated (useful since Raytracing can take a long time)

The macros let an architect assign, via a mouse, predefined properties to parts of their model. Thus a ground plane can be rendered as grass, gravel or bitumen with a few keystrokes, a sky environment map (at various times of the day) can be chosen, building materials such as brick, glass, tiles can be assigned to the appropriate parts of the model. The material library is extensible and a user may build up his/her own predefined materials as well as using the system defined materials.

The process of conversion requires that the Eagle model be saved in ASCII format - this contains geometric information for each part as well as attribute information. The Eagle macros will also produce a file which contains the world environment data which the Raytracer uses in building the scene around the model: size of image, viewing information, world environment information, etc. Next, this file is converted into an intermediary form (known as MOG).

MOG

The MOG format is used by all our geometry manipulation software. To cater for a new geometry format we simply write a to_MOG and from_MOG program. The MOG utilities are designed around the idea of Unix "redirection" and "filters". A typical MOG filter will read the geometry from standard input, manipulate it in some way, and output the new geometry. Several MOG filters can be strung together to perform complex operations without the need for temporary files:

```
egg2smog < model.asc | mogshare | mogtri | mog2rt > model.rm
```

In the above command, egg2smog reads in the Eagle ASCI geometry file (with the ‘<’), and pipes its output to mogshare (with the ‘|’). The geometry is now in the MOG format. mogshare reads in this MOG geometry, manipulates it, and pipes it to mogtri. mogtri does exactly the same thing: it reads in the geometry, manipulates it and passes it along. The final filter in the chain is mog2rt. It reads in a MOG geometry, converts it to the CAN Raytrace format and the output is finally written to a file (with the ‘>’). It is worth noting that even though most of the operations were done on MOG geometries, not a single MOG file was actually produced. Similarly

```
mog2smog < model.asc | mogshare | mogtri | mog2rt > model.asc
```

Essentially the same can be done to completely different modeler and renderer formats with most of the same tools. The above converts from MEDUSA into Movie BYU.

MOG is essentially a language for describing geometry. A MOG geometry is made up of a header section followed by an objects section. The standard format for an object is

```
object_type [layer_number]
{
  parameters
}
```

The geometrical objects supported are:

- a polygon with an unrestricted number of vertices
- connected or meshed polygons
- a line with an unrestricted number of vertices
- connected or meshed lines
- reducers (from cones to cylinders with varying radii)
- spheres (with varying radii)
In the conversion from Eagle to CAN Raytrace format the ASCII file is split into a series of smaller MOG files based upon the value of the "material" attribute for each object within the model and the form of those objects. For example, all polygons "made of" chrome will be grouped into one file while all those made of teak will be stored in another file, and all reducers made of chrome will be stored into a third file. Concurrently, a single "material" file is created for each material used. This file contains the actual definition for the material in question as well as CAN Raytrace directives to "include" the actual objects made of that material.

Next, those MOG files containing polygons are passed through the filters mogshare, mogpunch and mogtri. This ensures the geometry conforms to the CAN Raytrace requirements. The last stage of the actual conversion process transforms each of the MOG files to an equivalent Raytrace input file. This is done (for each file) by mog2rti. During the process of building the Raytrace input files for the model, the overall Raytrace input file is generated: it simply contains the world environment data (in a slightly modified format) and a series of directives forcing the Raytracer to read the material and model input files.

To render the image, the Raytracer is simply executed with the model data, the material information and the texture maps. As is normal, complex models of 50,000 or more polygons can take several hours to render dependent upon shadow calculations, reflectance, transmittance. This time factor is strangely useful in that students learn to structure their work so that the rendering can be done overnight (if need be) and the client views the work the following day. Since clients are often very critical and may ask for numerous changes, the turnaround time can be reduced.

Figure 7 shows an image of a gold crushing plant in Africa - initial talks with the client indicated that a golden cast to the scene would be in keeping with the overall "ambiance" of the piece. Unfortunately, the General Manager of the company was alarmed at our rendering which showed (to him) a sulfurous atmosphere around the plant, and, worse, a mercury spill in front of the plant, both well known environmental hazards in gold production.

![Figure 7: Not what the client wanted!](image)
Students may view their images on 8 bit (dithered) or 24 bit (full color) screens. The images may be photographed, included in other documents, or color laser printed (via a commercial bureau). Some students use the images as part of their final year presentation as a computerized slide show or as animation.

Animated architecture

Raytracing is computationally very expensive - a single frame may take several hours to render. Animation requires 25 frames per second: assuming a television image takes 2 hours to render, then a 20 second piece of animation will take 20x25x2 hours or 40 days! It may be that your workstation is on a network and that several machines are available for work (especially at night). With 10 workstations the animation could be farmed out one frame to the first machine, one frame to the second, etc. In this manner, an animated piece would take only 4 days or 1 week of nights!

Even for a single frame, anti-aliasing, shadows, reflections, refractions may mean that a single frame takes 8 or more hours. It would be nice to have the machines on the network do individual or multiple scans to cut down the time taken. This may mean (due to communication overhead) that the image is available in 2 hours rather than 8.

The CAN Raytracer supports distributed network computation for multiple frame animation and also for single frames. The system uses a single consistent mechanism for this distributed computation - a farming program which uses a Unix shell script. This means that architecture students need not know about programming but should be familiar with concepts such as disk space and directory hierarchies.

Default global files, or individual files on the users home directory, indicate what machines are available for farming and what machines/directories are available for storing the resultant animation in.

A normal television image would require approximately 1.3 megabytes per frame. This would be 33 megabytes for 1 second and 2 gigabytes for 1 minute of animation. Luckily most formats used to store images will use some form of run length encoding - the information stored is not the rgb values but codes which indicate when runs occur [42, 39]. A typical file with no complex textures can be reduced by 25% under normal circumstances. Textures add complexity to a scene and tend to reduce runs of colors because the texture is not just a single color.

This reduction though is still not sufficient to save enough space (we might still be looking at 1 gigabyte for 1 minute of animation). To save a lot of space we need to compress between frames: temporal compression. If little changes between frames - only a few things are animated - then consecutive frames will have similar rgb values for the same scan lines. Therefore if we take a large number of consecutive frames, we can store the first and then store the differences between the first and the second, and then store differences between the second and third and so on. Our file of many frames now becomes a file with a base frame plus consecutive frame differences.

This technique is very efficient because the differences are simply partial scan lines with runs of the rgb values where changes have occurred. The efficiency is gained by the long runs where no changes occurred; these save up to 90%. Therefore the amount saved depends upon how much changes from frame to frame. Using these compression techniques, 300 frames or more have been stored in about 60 megabytes equating to about 200 K per frame.

Compositing [34] of scans in images or of static time expensive scenes which have 100,000 polygons can add visual complexity to an animated piece without adding to the rendering time. Figure 8 shows a frame from an animated piece which has the static star background overlaid by the static rendered tower, then overlaid by the static rendered Lindenmayer System trees. Between the tower and the trees, an animated rendered helicopter (not shown in this still) flies by.

In preparing for animation, students can move through their wireframe models in real time and record various from, to, view up and field of view values at keyframes. These values can then be splined and reused as input for a realtime wireframe or shaded walk through. These parametric keyframes can be graphically manipulated [30] to create the best looking
animation. For example, a camera view from a helicopter liftoff, approach to a building, hover, circle and landing in a courtyard was orchestrated using this method in less than 10 minutes. The values obtained can also be used to specify the position and orientation of the helicopter for different scenes.

The keyframe values can then be directly used in the CAN Raytrace farming system to specify the frame by frame position, view, etc. of objects within an animated piece. This automation again means that the architecture students do not need to know the inner workings of the animation system - only the mechanisms needed to produce a result.

Conclusions

The cooperation between Architecture and Computer Graphics has shown itself to be a valuable environment for generating high quality images from true 3-D CAD databases in minimal time. Rendered images (animation and single frames) have been well received by computer graphics peers in Australia. The interface to the CAD software and the farming mechanisms used in the production of animated pieces and single frames greatly reduce the time taken to complete a finished image and enable architects to use a high quality rendering system.

Although current practicing architects cannot offer these facilities, all students are an investment in the future: those graduating this year with pencil in one hand and technology in the other will be well positioned to select for employment this year's beginners when they graduate in a few years time.

The entry level cost for an AEC system is roughly $15,000 for hardware and software plus wages during the training period. Practicing architects will see these beginners graduate at a time when the AEC system will cost half that and the students will not need training but may in fact train their employers. Our preparing of the students for modern architectural practice will become, indirectly, further education for architects in the field.
Until then it is hoped that by showing practicing architects what can be done through seminars, journals, a bureau service and conferences, they will start to embrace (or at the very least not crush new employee enthusiasm for) new technology which can be of benefit to them.
References

32. A. Oliver, “Domestic Housing Design Programming,” Honours Thesis, Architecture,
Curtin University of Technology, 1991.
42. Utah Raster Toolkit, Utah Raster Toolkit, University of Utah, 1988. PD Computer Software.