Interactive Computer Rendering

Paul Richens and Simon Schofield

Martin Centre for Architectural and Urban Studies
University of Cambridge Department of Architecture
6 Chaucer Road
Cambridge CB2 2EB.

Summary

Interactive Rendering combines the geometrical precision of classical computer graphics with the representational freedom of a paint program. It is more sympathetic to the ways in which designers use images, and overcomes many of the frustrations experienced in rendering from CAD models.

The scene is generated in a standard viewing application, but saved as a specially enhanced raster image. The extra information allows the Interactive Renderer to apply brushed-on rendering effects which are sensitive to the perspective of the image. Effects can be applied locally or overall, and may be overlaid, blended and erased to create complex combinations. A huge range of treatments are obtainable, both photorealistic and not.

INTRODUCTION

If we compare the processes of making images with CAD, and making them by hand, we see that they are profoundly different. So the qualitative differences found in the respective images are hardly surprising. The potential advantage of computer rendering is in the rapid generation of images showing correct perspective, lighting, and high levels of detail, once a model has been created. It is upon this promise that CAD has gained much of its popularity. But in practice, current rendering systems pose significant problems to artists and designers, who must strive to achieve an image, tentatively held in the mind, via a complex and tedious process. This involves a lengthy cycle of altering obscure input parameters to an algorithm that provides far from instant feedback, then waiting for results and re-rendering with changed parameters. The continuous organic growth of the hand-made image is replaced by a series of disconnected and inconclusive experiments - modifications made to the parameters may result in changes which are hard to judge, because the previous rendering (and its set of parameters) may not have been kept for comparison's sake. Viewing a succession of results may lead eventually to a type of image-blindness, where the designer becomes unable to remember where the image has come from or where it is going.

Hand methods, of the sort used in painting and drawing, are immediately more responsive to the ways artists and designers want to work by providing continuous, uninterrupted methods of image construction and instantaneous feedback. The final image gains quality, clarity and conviction more from the gradual process of construction than it does from the original, isolated, conception in the artist's "mind's eye". The artist is able to make localised changes, using diverse representational devices in a single image, resulting in a richer structure than ever emerges from the "blanket" render of a CAD system. The image can be pinned to the wall for hours of gradual consideration before the next change is applied. One can come back to an image the next day in order to cast a fresh eye over it before attempting further refinements. All this makes for a relatively stress-free and far more engaging process. The subtleties and nuances eventually residing in the final hand-made image are such that no amount of formal description and parameterisation, of the sort required by a rendering algorithm, could ever reproduce the same results.

The nearest computer-based counterpart to the traditional approach to image-making is the digital paint system such as *Photoshop* (Knoll, 1993). Indeed, it is common to see CAD images moved to such a system for refinement. Unfortunately paint systems offer no assistance in generating the complex perspectives and tonal approximations produced by renderers. Computer based paint systems expect the user to handle the intricacies of perspective and tonality by eye alone.

Rendering by computer

It is not intended to dwell on the technicalities of computer rendering, but it is necessary to introduce the fundamental ideas and terminology. For a full treatment, see Foley, van Dam et al. (1990) .

A computer model of a building represents the salient points by means of three-dimensional (x, y, z) co-ordinates, linked together to define the boundaries of planes. Curved bodies are usually approximated by small flat facets before they are rendered. The process of rendering can be considered as a series of steps dealing with: perspective, lighting, visibility, smoothing of curves, shadows, and textures [Fig 1].

1 *Classical rendering algorithms. Top left: simple flat shaded facets. Top right: smoothing restores the appearance of curved surfaces. Bottom left: solid texture. Bottom right: shadow casting and bump mapping.*

Perspective, which is hard to set-up on a drawing board, is trivially easy to compute. The origin of the co-ordinates is chosen to be at the station point, with the x-axis horizontal, and y-axis vertical. The z-axis then points along the line of sight, and so measures distance away from the eye. Each (x, y, z) co-ordinate is replaced by (x/z, y/z, z/z), which has the effect of projecting objects in perspective onto a picture plane at z=1. So perspective is obtained simply by dividing by distance: an object twice as far as another will appear at half the size.

Lighting is typically calculated at each vertex of a facet, taking into account its orientation with respect to the light source, and its reflectivity. Values inside the facet are obtained by a kind of averaging. Smoothing is obtained by averaging between facets. Shadows can be computed geometrically, but it takes a long time. Light reflected from one surface onto another is generally ignored.

Deciding what is visible, which requires very little mental effort from a draftsperson, is surprisingly difficult to compute. The simplest algorithm, and the one most important for our purposes, is called the z-buffer algorithm. The image to be created is divided into tiny elements called pixels, arranged in a regular grid or raster. (A small computer or video screen provides a raster of about 640x480 pixels.) The algorithm requires memory to record the colour and brightness of each pixel - called a frame buffer - and more memory to record the distance to each pixel - called the z-

buffer. In operation, each facet is broken down into pixels, and the z-coordinate determined. Consider a point on a surface; we want to find out whether it is in front of, or hidden by, a previous point already rendered and stored in the z-buffer. If it is greater than the value in the z-buffer (further away), nothing happens. If less, then both the frame buffer and the z-buffer values are replaced with those calculated for the new pixel. When all facets of all objects have been scanned in this way, each pixel holds the result for the nearest surface at that position, which of course is the one that is visible. Finally the frame buffer is displayed on the screen - and the z-buffer is discarded.

Texture mapping deals with materials that have some sort of grain or patterning. For example a sample of brickwork can be scanned, and then mapped repeatedly onto a wall. Solid textures, which are usually calculated, are used to fake the grain of materials like wood and marble.

The goal of computer graphics research has become what is known as "photorealism". A theoretically perfect rendering from a computer model would be indistinguishable from a photograph of the realised object. In order to achieve this we need to simulate in more detail, and with greater precision, the interactions of light with matter. One strand of current research is attending to indirect lighting, colour bleeding and partial shadowing. Another studies the effect of the micro-structure of materials on reflection, and a third the effects of atmospheric haze. Each advance requires a more refined model, and a great increase in the time taken to render a scene. We begin to wonder whether it is all worthwhile.

The economics of computer rendering depend primarily on the complexity of the model, usually estimated by a polygon count, and secondarily on the sophistication of the optical simulation. Models of a few thousand polygons can be rendered in an elementary way in a few minutes on a typical desktop computer, and in a fraction of a second on a specialised graphics workstation. However photorealism demands a complex model, full of detail. Large architectural models may run into hundreds of thousands of polygons, and take months to build and days to render. Elements of entourage, such as figures and vegetation, can be disproportionately expensive: a tree needs at least 2000 polygons, yet its detailed form is of no particular consequence.

The process of producing a rendering from such a detailed model is painful and time-consuming. It involves choosing viewpoints, assigning the optical properties of surfaces, placing lights, defining and placing textures. To check the result requires a trial render, which, if photorealism is intended, may take hours. It is commonly found that it takes as many weeks to render a set of images, as it does to build the geometric model in the first place.

Objectives

Our work on architectural rendering by computer aims to break out of some of these difficulties. We do so by questioning the two fundamental assumptions underlying mainstream computer graphics. One is the single-minded pursuit of the photorealistic image: photographs do not communicate buildings very well, and architects have rarely used hyper-realistic images to promote their schemes. The second is the notion that rendering should be a deterministic algorithm - where you set everything up in advance, then press "GO" and wait for an exactly repeatable result. We would prefer considered (but unconstrained) human intervention, serendipity, and chance, to play a larger part.

The basis of our approach is to split the production of an image into two stages. The first deals with perspective projection, the determination of visibility, and possibly the casting of shadows - in other words all the strictly geometrical tasks. This is handled by conventional fast graphics techniques (the z-buffer algorithm), and will not be considered further. The second stage, which is the subject of this paper, turns this abstract geometrical solution into a fully rendered image. Both stages are highly interactive, and provide instantaneous feedback. The crucial data-structure that passes information from the first stage to the second (called EPix) is described later.

Early results provoked a good deal of interest from architects experienced with CAD. With their help, we were able to formulate some likely goals for the two-stage approach:

1. To allow for a more relevant, and economical, alternative to photorealism, sharing instead some of the qualities of painting, drawing and print-making.
2. To facilitate a "hand-held" technique, in place of the deterministic algorithm, enabling an image to be finished interactively in an hour or so. We have often noticed architects tracing over and re-rendering their computer output by hand. This is not entirely for the sake of the image: the massaging of drawings is an essential stimulus to the architectural imagination - a fact not yet acknowledged by CAD systems.
3. To allow a richer, rougher style of rendering to compensate for lack of detail in the model, and to render entourage (such as figures and trees) economically.
4. To enable montages of computer models with context photographs to be rendered in a consistent manner.
5. To allow for vagueness in presenting proposals. The photorealistic image appears too definite, worked-out, expensive and non-negotiable to be appropriate when exploring tentative ideas.
6. To allow satisfactory images to be produced at low pixel resolutions. The glossy smoothness of photorealism demands high resolution (e.g. 4000 raster lines for a slide-maker), which consumes huge resources. It should be possible to make images matched to the qualities of low-cost raster-oriented printers such as ink-jet and laser printers.

Conventional rendering is most easily applied to shiny new high-tech buildings, preferably isolated from any context. Vernacular buildings using traditional materials in a landscape setting are much harder. The ultimate challenge is to render a ruin (think of Piranesi's plates of Hadrian's Villa) which has lost its definite geometric form, whose materials are crumbling and revealing inner layers of rough stuff, surrounded by debris, dank and overgrown [Fig 2]. To render these by precise geometrical modelling, and optically exact rendering, is wrongheaded. Precision is not the issue. We hope, by overturning the conventional approach to rendering, to find a way of handling ruins.

2 *Part of Hadrian's Villa, from Piranesi's Antichità Romane. Normal CAD techniques cannot hope to produce a rendering of ruins such as these.*

Related work

The simple technique of using a pen-plotter with loose, wobbly pens was found to produce surprising and effective results (Bakergem and Obata, 1992). Architectural students, who previously fought shy of including any computer graphics in their portfolios, felt far less inhibited over including these "freehand" plots [Fig 3]. They offer several possible explanations for this; one is that each drawing is unique in that the deviations of the pen cannot be reproduced; another is that they are simply more visually engaging – they seem to have more detail, or at least contain more to look at.

3. *Squiggly pen plot shows a degree of liveliness not normally found in mechanically plotted drawings. However the congestion in the background remains a problem (courtesy W. Davis van Bakergem, Washington University, St Louis).*

Some of the fundamental ideas of Interactive Rendering (in particular the separate treatment of geometry and appearance, and the enriched pixel) were suggested by Perlin's *Pixel Stream Editor* of ten years ago (Perlin, 1985). However, it was far from interactive, and required that each pixel should be processed independently, which prohibits many desirable operations. The ideas of fractal noise, turbulence and solid texture included in the same paper have been immensely influential.

Perlin was rather vague about what data should be associated with each pixel; Saito and Takahashi (1990) by contrast define a comprehensive G-buffer that contains identifier, parametric coordinates, world coordinates, depth and direction cosines. This is used to support perspective space hatching and edge finding, with the idea of making technical illustrations "comprehensible". Many of their ideas have been developed and implemented in 5D's *KATI Renderer*, intended for mechanical engineering illustration (Glazzard, 1993). We differ in having some interest in ambiguity as well as clarity, and preferring an interactive process to one that batch treats a complete image. We have found it possible to reduce the G-buffer to a single number, without serious loss of effect.

The idea of image space rendering using synthetic paint marks was important in the genesis of our project. Similar ideas have been published by Haeberli (1990), and have found their way into commercial products such as *Fractal Design Painter* (Zimmer and Hedges, 1991). It was about this time that the term "Non Photo Realism" came into use (presumably by analogy with "Non Dairy Creamer"); we prefer to avoid such a negative characterisation.

Three-dimensional painting by means of reverse texture mapping was first described by Haeberli and Hanrahan (1990). It is beginning to appear in commercial products, but requires very high performance hardware to be interactive. It works purely in perspective space.

The recent work on stroke textures by Winkenbank and Salesin (1994) [Fig 4] is closer in spirit to ours, and is aimed specifically at architectural rendering, but is much less interactive than we would like.

The importance of indeterminate form and visual noise in allowing the viewer scope to "project" his own imagination on to an image was recognised long ago by Gombrich (1960) who points out that it has been a commonplace of painting and drawing since ancient times, and that it has been discussed theoretically by Alberti, Leonardo and many others. The modern world of digital image-making has been comprehensively surveyed by Mitchell (1993).

Early work on this project has been described by Richens (1994). A comprehensive review of non-photorealism can be found in Lansdown and Schofield (1995).

***4**. Simulated pen and ink drawing using procedurally generated textures that can respond to perspective and lighting (courtesy G Winkenbach and D Salesin, University of Washington, Seattle).*

REALISATION

Input - the enriched pixel format

Our concept of Interactive Rendering builds on the common practice of refining CAD images in a paint system, such as Photoshop, by transferring significantly more data to the finishing program.

A conventional frame buffer contains three numbers for each pixel, these being the brightness levels for the red, green and blue guns in the colour tube that will present the image. We have developed a new file format – The Enriched Pixel Format

***5**. A z-buffer drawn as a monochrome image. Brightness in the printed image corresponds to distance in the z-buffer; the darkest areas are the closest.*

(shortened to EPix) – that adds also the contents of the z-buffer, and a third material buffer, which serves to classify the source from which each pixel was derived (e.g. sky, ground, brick, glass). The z-buffer gives, for each pixel, its distance from the eye [Fig 5]. The EPix file also contains the perspective transformation used to generate the image. It is possible to use this transformation backwards (in conjunction with the z value) to recover the original model coordinate for any pixel in the image.

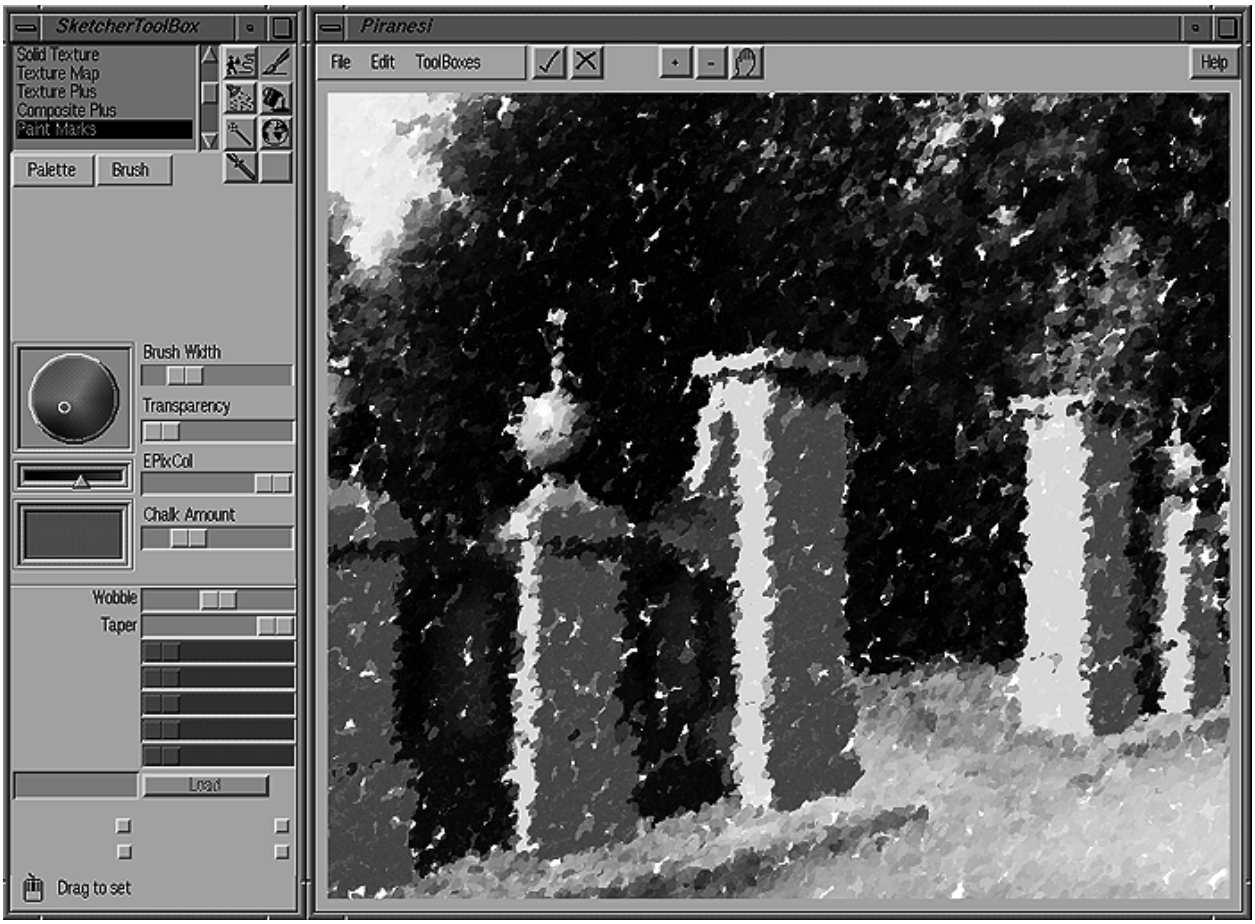
In fact, a great deal of valuable information can be deduced from the z-buffer. The z value for a single pixel tells you its distance. By inspecting the z values of adjacent pixels, you can deduce the orientation of the surface at that point. By looking a little further you can calculate curvatures and deduce the presence of edges.

We use a slightly modified z-buffer rendering algorithm to generate the EPix file. The first modification is simply to retain the z-buffer, which is usually discarded. The material buffer is computed by a simple trick; the scene is rendered without lighting, and with the colour at each vertex set equal to the material tag for the facet. The resulting frame buffer is saved as the EPix material buffer. Most viewing applications could easily be modified to provide the same data.

When the Interactive Renderer is started, it loads the EPix image, and adds a second frame buffer (called the canvas) to receive the output image. This may be initialised to the same content as the input frame buffer, or any other image, or left empty, as needed. Internally, additional buffers are used to control the application of rendering effects, and to provide for "Undo" and check-pointing of the work.

The interface

The user interface [Fig 6] consists of two parts – a painting window and a tool box. Upon loading an EPix file, a copy of the frame buffer is displayed in the window – this is the "canvas" on which the user works. A rendering effect is chosen from a scrolling list of possibilities pictured at the top left hand of the tool box.



***6**. Prototype user interface for an Interactive Renderer.*

Different effects require different controls, and the interface adapts by assigning sliders and option buttons to specific parameters. Certain controls are common to all effects, such as transparency (or degree of effect), colour, and for brushed effects, brush size. Settings can be saved in a palette (pictured in the centre of the tool box), each button of which has a pictorial thumbnail reminder of the effect – these can be saved and reloaded for later sessions. The system has been written in such a way that it is easy to author and add entirely new effects to the scrolling list.

In addition to the sliders, most tools can be "trained" using the second mouse-button, for example to control the orientation of hatching, or the length of a brush stroke.

Brushing

Almost all the effects are applied by brush tools, similar in concept to those in a conventional paint program, but with some marked differences in behaviour. They do not require the construction of elaborate stencils or masks, as they have their own ways of achieving precision. The simplest of these is the material

***7**. Brushed-on opaque colour, transparent ink and various textures. The matting functions make it easy to apply colour to a single material (as on the background), or a single plane (as in the further pier).*

matting function; at the start of a stroke, the material at the cursor point is inspected and from then on the brush will only paint onto parts of the scene with that same material type. Plane matting is similar in its function, but instead of inspecting the material, it looks at the z-buffer and restricts painting to coplanar pixels; hence surfaces of a building, for instance, can be painted in isolation from one another, and windows can be quickly in-filled with a dark colour [Fig 7]. All effects can be applied using a hard or soft edged brush, of varying size and intensity.

Paint types vary from an opaque "gouache", which obliterates whatever is painted over, to a transparent "ink", which acts like a varnish layer over the existing scene. Ink has proved to be the more useful of the two; textures and colours can be applied to the scene while preserving the lighting and shadow calculations in the original frame.

A common problem evident in paint systems is in the application of such translucent effects by a brush tool; often the user will accidentally over-paint the same area twice resulting in a doubling of the intensity of the effect in that region. We avoid the possibility of accidental over-painting through an "accumulation lock". When set, the accumulation lock monitors the amount of effect applied to each pixel and disallows any over-painting until the

accumulation lock is released. Hence parts of a wall can be painted with several different textures without the possibility of overlapping. Fig 7 shows a wall rendered with two textures – a brick and a vellum-render effect – which have been applied freehand; there is no overlapping between the two effects.

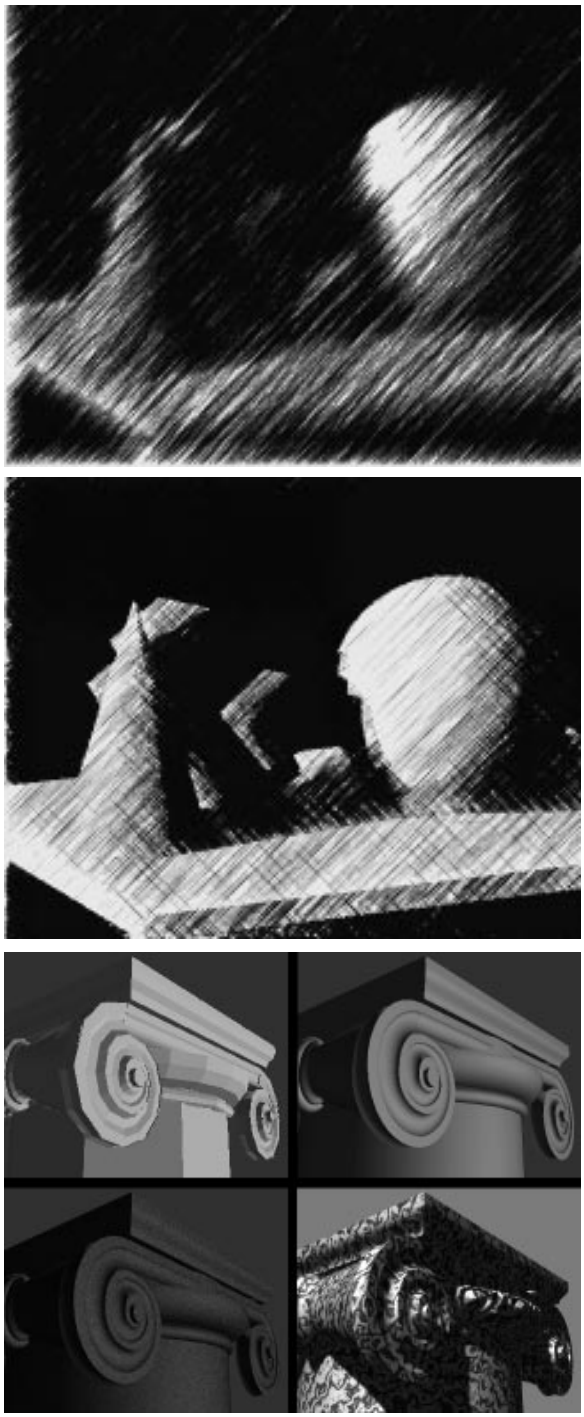
***8**. Brush strokes pick up the original tones, producing a soft image with low detail, but a certain amount of structured visual noise. Used here to present a master plan, where it is required to convey something about the landscape, and the approximate size and position of buildings, as yet undesigned. There is enough noise to encourage the viewer to imagine the details for himself.*

Image space rendering

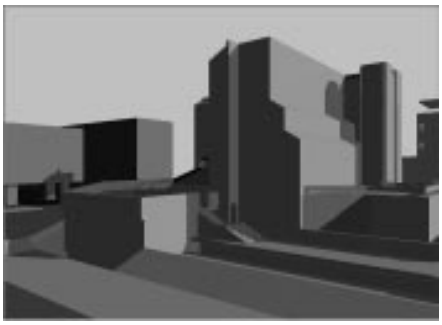
It is convenient to distinguish operations that work in the image plane from the more advanced techniques that use the z-buffer, and so can be considered as working in perspective space. The simplest of the former, in our system, is the "Restore" brush, which simply replaces whatever is in the output canvas with the content of a previously saved image. So it is always possible to recover from mistakes.

Mark-making techniques effectively repaint part of the image by rapidly placing brush strokes on the canvas [Fig 8]. Strokes are controllable in length, direction, straightness, and graininess. They may have their own colour, or "pick up" the colour of the original frame, or make a blend between the two. Marks may be precise (in that they do not run across the boundary between different materials) or loose. Marks may be scattered at random over the image, or controlled closely by the mouse. Their effect is to decrease the precision of the original image, and add a variable amount of "noise" in the form of a surface texture.

Image space hatching places more or less regular patterns of lines on the image, with controllable angle, thickness, spacing and orientation. They can respond to the underlying frame buffer by changing some of these parameters [Fig 9]. Generally they respond to the brightness of the image - for example hatching lines become thicker or closer where the image is darker. A more sophisticated response is to control the orientation of hatch lines by looking at the gradient of the brightness; this is especially valuable in hatching curved surfaces.



***9**. Experimental image space hatching techniques, where the lines respond in different ways to the lighting of the scene: **a.** Lines vary in weight. **b** Lines vary in spacing. **c** Lines vary in weight and direction. Lines are oriented at 45 degrees to the gradient (direction of most rapid change) of the lighting.*



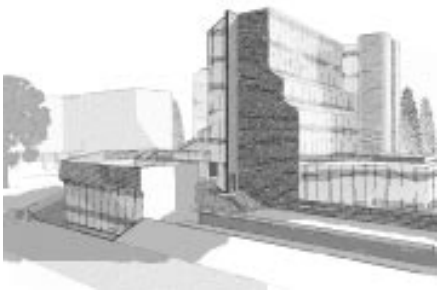
Perspective space rendering

When drawing a quick sketch one is likely to use image space hatching - for example a constant 45 degree slope used roughly to distinguish light from dark. A more careful rendering will use hatching with more control, to delineate the form and texture of objects, as well as light and shade. These techniques, which must refer to the z-buffer, we term perspective space rendering.

The simplest of them is the "Fog" brush, which places paint of variable opacity on the output image. The opacity is controlled directly by the z-buffer, so the paint is thicker where the distance is greater. The effect is of fog, smoke, haze, or aerial perspective,

depending on the colour and intensity used [Fig 10b].

A second simple, but gratifying, effect is to use an edge-detection filter to draw outlines. Edge detection is a standard image-processing technique, which responds to the rate of change of the brightness of a scene. The results tend to be the opposite of what is wanted - for example heavy edges around shadows, where the contrast is high, and light edges between parallel planes of a building. We apply the same algorithm, not to the brightness values in the frame-buffer, but to the distance values in the z-buffer. The result is a sensitive line drawing, with silhouette edges emphasised (great change in z), interior angles drawn lightly, and shadow edges ignored altogether [Fig 10c].



***10** Alternative treatments of the same scene. **a** is the original scene, with cast shadows, but otherwise modelled and rendered in the simplest possible way. **b** has been interactively rendered using several perspective space techniques including photographed textures, hand drawn cut-outs, and fog. **c** uses edge detection and hand-drawn textures which are then applied in perspective.*

The mathematical principles we use to apply textures in perspective are the same as those employed in conventional computer graphics, but the interactivity and control available is much superior. Flat textures are obtained by scanning photographs, or samples of actual materials. They are then brushed onto the image, with full control of scale, orientation and intensity. The perspective is corrected automatically, by projecting the sample orthographically onto the plane of the pixels, or alternatively by projecting from a fixed direction.

This technique can be used photorealistically, but has much richer possibilities. Textures can be fully applied, or transparent and overlaid, or combined with other kinds of hatching. Textures can be drawn by hand, then scanned and mapped into perspective space.

Solid texturing is a technique which is conventionally used to represent the grain of materials such as wood or marble. We find it equally applicable to generating more abstract hatch patterns, such as those employed in woodcut illustrations and other forms of print-making. For example, regular horizontal hatching (brick courses) can be rendered by defining a solid texture which consists of a stack of horizontal planes at the appropriate spacing. The solid texture brush converts each pixel back to the original model space, checks whether it lies on a texture plane or not, and fills in the pixel accordingly.

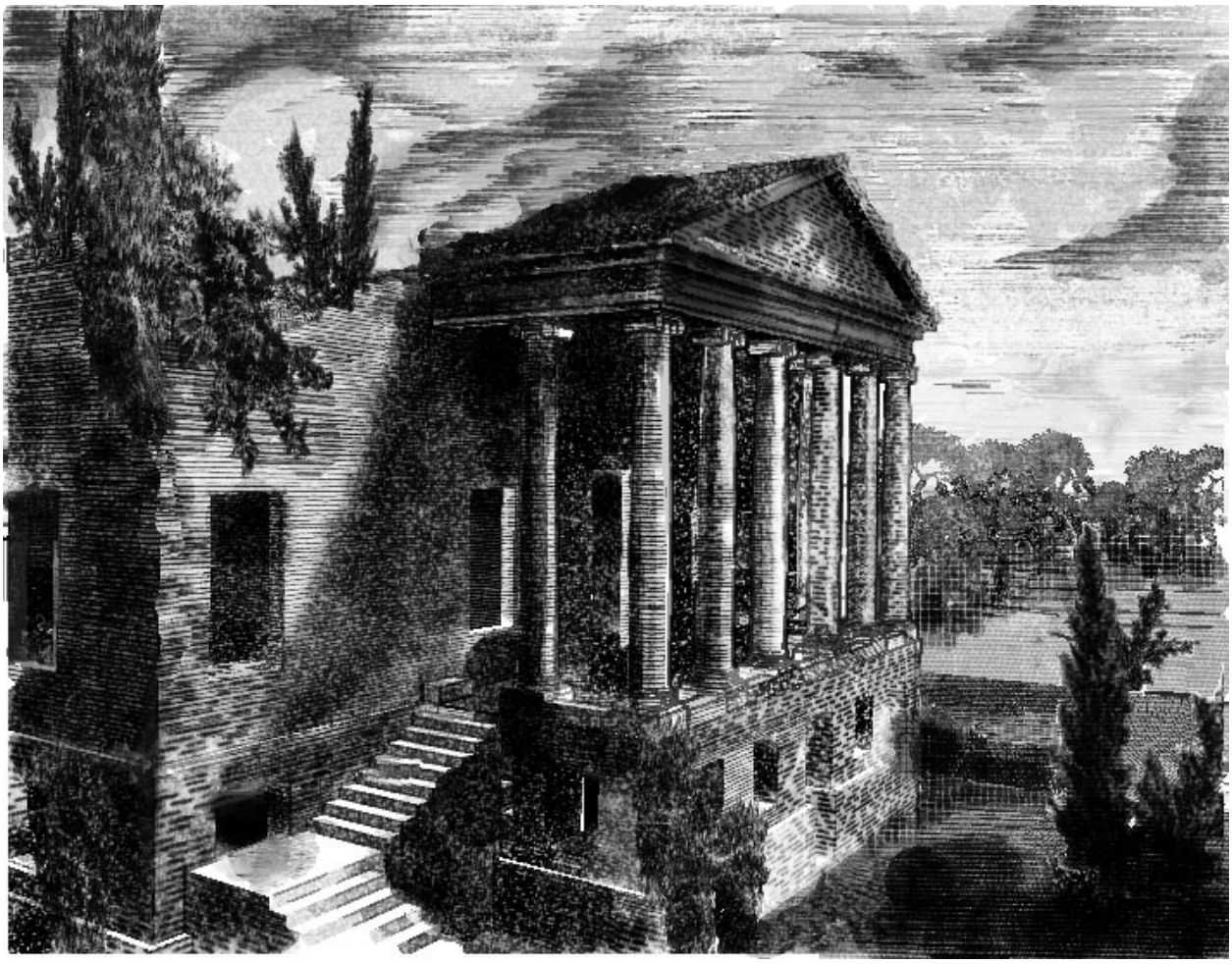
Solid textures are particularly easy to apply to curved surfaces and complex forms. They are mathematically defined, so there is no need for scanning, and it is relatively easy to make them respond to the input frame buffer or z-buffer. It is also easy to combine them with noise functions so as to obtain a controllable variation in thickness, intensity or position. In conventional renderers these effects are very difficult to control, because it takes a long time to see the result on a trial rendering and the parameters for solid textures are particularly arcain. Interactive Rendering, on the other hand, gives immediate feedback and "undo" capability meaning that effects can be arrived at through trail and error more quickly.

Compositing and modelling

Larger graphic elements, such as entire trees, vehicles and human figures can be applied wholesale to the image, after some brief preparation by the user. These are "cut-out" shapes, generated from a scanned image, and prepared by adding an alpha-mask (to define transparency) and an origin point. Like flat textures, they can be generated either from photographic or drawn material, and can be placed either to face the viewer or in the plane of a surface; the difference is that they are placed as a unit, rather than being brushed on.

Cut-outs inspect the z-buffer, and scale themselves according to perspective. If a tree is partially masked by a building, the z-buffer is used to calculate which parts are visible [Fig 10,11]. When a cut-out is applied, not only does it modify the visual aspect of the scene, it also modifies the z and material buffers of the EPix data. So a second tree can be partially masked by the first, and the surface of a tree can receive further treatment, such as brushing on a texture. Compositing can be thought of as a modelling operation; once a shape has been placed in the scene it behaves exactly as though it was part of the original model and enjoys the benefits of material and plane matting, as well as all the rendering operations.

One further effect, in the same category of modelling operations, enables a new plane to be defined in the z-buffer by painting what is effectively a "sheet of glass" into the z-buffer. In fact it is commonly used to put glazing into blank window openings, prior to using a texture brush to paint reflections. It can also be used to fill in missing surfaces, or to extend a ground plane to the horizon. Somewhat surprisingly, it can also be used in the opposite sense - to extend the sky or other background into regions where nearer surfaces are defined. The visible effect is to demolish part of the model, producing a ruin [Fig 11].



11 *By extending the sky into the upper storey, the building is effectively reduced to a ruin. Creepers are cut-outs that orient themselves to the underlying surface.*

Further work

One can conceive of an interactive renderer (based on the ray-tracing principle) which would refer constantly to the original 3D model instead of to the saved z-buffer. In some ways this would be an advance, as it provides better access to information about lighting and the geometry of surfaces, and would allow a greater mix of modelling operations with the rendering. But such an approach would be less responsive, especially with complex scenes, whereas the EPix background guarantees that responsiveness is completely unaffected by the complexity of the scene. This is a considerable advantage, and suggests that a hybrid system might be worth considering in future work.

CONCLUSIONS

Interactive Rendering succeeds in redressing the legacy of inappropriate images and the associated frustrations of using CAD systems by re-investing in the judgement and skill of the artist or designer, but without dispensing with the inherent advantages of 3-D rendering. Not only does it produce better and more varied images, it is significantly more enjoyable and rewarding to use. This is achieved by supplementing the rendering process with wholly interactive techniques, many of which pay scant regard to the demands of photorealism.

The unerring photorealism of classical 3-D rendering is often perceived as a problem by artists and designers. Interactive Rendering enables the application of many alternative treatments to the image. While photorealism of the sorts witnessed in conventional computer images is entirely possible, it is with an eye on the softer, more expressive, freer forms of representation seen in traditional imagery that the system has been developed.

The output image is constructed gradually; techniques can be easily overlaid and applied highly locally in ways difficult (or impossible) to describe to an automated renderer. Applied techniques can be erased and over-rendered. Residual traces of past explorations and decision-making may, in fact, add richness to the final image. The process can be interrupted and restarted without difficulty. The result can even become the input for a further round of rendering.

Interactive Rendering has really to be experienced to be appreciated. It "looks" much like a paint program, but "feels" quite substantially different. Initially this is because of the surprising but agreeable helpfulness of the material and plane matting actions. The difference deepens when the perspective rendering effects are called into action, such as fog, texture brushes, or the placing of cut-outs in perspective. These feel more like modelling, but are far more visually oriented. The paradigm is new and distinct from what has gone before; naming it in terms of the familiar is rather problematical. "Three dimensional painting" describes pretty well what it feels like. "Interactive rendering" describes what it does.

The result of splitting the rendering process into two parts, one dealing with geometry, the other with surface appearance, while passing a sufficient amount of data between them, enables both processes to become highly interactive, even on computers of modest power. High interaction gives immediate feedback; the consequence is that intention and realisation converge rapidly and pleasurably [Fig 12].

While most of our initial objectives have been realised to a satisfactory extent, somewhat elusive have been the pursuit of "noisy" rendering techniques to compensate for lack of detail, and to allow the viewer to project (in Gombrich's sense) what he hopes

12. *A finished rendering using multiple overlapping effects (courtesy J Rollo, University of Cambridge).*

to see, and the related issues of rendering incomplete and tentative proposals. We have had occasional successes, but lack a recipe for reliably reproducing them. Problems of montage and entourage are convincingly solved, as is the introduction of a much wider range of graphical techniques, and adaptability to many forms of printing. One thing, though, remains irksome. Despite our reservations about classical photorealism, we have to admit that Interactive Rendering turns out to be a very effective way of producing it.

Acknowledgements

Brian Logan, John Rees, John Rollo and Tim Wiegand at the Martin Centre have made valuable contributions, as have many visitors, but especially Richard Coyne, Iain Fraser and Bill Mitchell. Architects who have contributed ideas, and projects to try them on, include Nicholas Ray, John Hare, Max Fawcett and Roger Matthews. We are deeply grateful to Informatix Inc, for financial support, and to Masanori Nagashima for his enthusiasm and far-sightedness.

References

Bakergem, W.D. van and Obata, G. (1992). Free-hand plotting – Is it live or is it digital? In (ed) G.N. Schmitt, *CAAD Futures '91.* Vieweg, Wiesbaden.

Foley, J.D., van Dam, A. et al. (1990). *Computer Graphics Principles and Practice,* 2nd ed., Addison Wesley, Reading MA.

Glazzard, N. (1993). *The KATI Renderer* (software). 5D Inc, London.

Gombrich, E.H. (1960). *Art and Illusion: A study in the psychology of pictorial representation.* Phaidon, London.

Haeberli, P. (1990). Painting by numbers: abstract image representation, *Computer Graphics,* vol. 24, no. 4, ACM SIGGRAPH, pp. 207-214.

Haeberli, P. and Hanrahan, P. (1990). Direct WYSIWYG painting and texturing of 3-D shapes, *Computer Graphics*, vol. 24, no. 4, ACM SIGGRAPH, pp. 215-224.

Knoll, T. et al. (1993). *Photoshop* (software). Adobe Systems Inc., Mount View CA.

Mitchell, W.T. (1993). *The Reconfigured Eye.* MIT Press, Cambridge MA.

Perlin, K. (1985). An image synthesiser*, Computer Graphics,* vol. 19, no. 3, ACM SIGGRAPH, pp 287-296.

Richens, P. (1994). Does knowledge really help? In (eds) G. Carrara and Y.E. Kalay, *Knowledge-Based Computer-Aided Design, pp. 305-325.* Elsevier, New York.

Saito, T. and Takahashi, T. (1990). Comprehensible rendering of 3-D shapes, *Computer Graphics,* vol. 24, no. 4, ACM SIGGRAPH, pp. 197-206.

Lansdown, J. and Schofield, S. (1995). Expressive rendering: an assessment and review of non-photorealistic techniques, *Computer Graphics and Applications* Vol 15, (3) , IEEE, pp29-37.

Winkenbank, G. and Salesin, D.H. (1994). Computer generated pen-and-ink illustration. In *Computer Graphics,* Proceedings, Annual Conference Series, 1994. ACM SIGGRAPH, pp. 91-100.

Zimmer, M. and Hedges, T. (1991). *Fractal Design Painter* (software). Fractal Design Corporation, Aptos CA.