Does Knowledge really help? – CAD Research at the Martin Centre.

Paul Richens

Director

Martin Centre for Architectural and Urban Studies

University of Cambridge Department of Architecture

Summary

The Martin Centre CADLAB has recently been established to investigate software techniques that could be of practical importance to architects within the next five years. In common with most CAD researchers, we are interested in the earlier, conceptual, stages of design, where commercial CAD systems have had little impact. Our approach is not Knowledge-Based, but rather focuses on using the computer as a medium for design and communication. This leads to a concentration on apparently superficial aspects such as visual appearance, the dynamics of interaction, immediate feedback, plasticity. We try to avoid building-in theoretical attitudes, and to reduce the semantic content of our systems to a low level on the basis that flexibility and intelligence are inversely related; and that flexibility is more important. The CADLAB became operational in January 1992. First year work in three areas – building models, experiencing architecture, and making drawings – is discussed.

Introduction

The Martin Centre had a very early involvement with architectural CAD, more than twenty years ago, which led quite rapidly to commercial exploitation, and the formation of Applied Research of Cambridge Ltd. As the commercial work succeeded, the academic interest dwindled, and has only recently been resuscitated. Most of my career has been in commercial development of software for architects, originally at Applied Research of Cambridge, and latterly at McDonnell Douglas Information Systems. I made two major developments - the OXSYS system in the seventies[1], and GDS in the eighties. Several other projects never left the drawing board.

A few years ago, having joined the research staff at the Martin Centre, I was invited by Masanori Nagashima, President of Informatix, to establish a laboratory for CAD research at the Martin Centre. Informatix is the Japanese distributor for GDS, MOSS and several other packages of importance in the construction industry. We decided quite early on to avoid the “thinking machine” paradigm for architectural CAD, and focus rather on using the computer as a “medium for design”. The purpose of this paper is to discuss the reasons for rejecting an approach that is widely adopted by academic researchers (though rarely by those in the industry), and to give some definition to the “design medium” alternative by describing results obtained in the first year of research.

While the agenda for the newly established CADLAB was still under discussion, we both attended the conference organised by Gerhard Schmitt in Zürich (CAAD Futures 1991). This was a valuable opportunity to take stock of what was going on in academic research. What we heard there has had quite important effects, both positive and negative.

There were a number of practitioners among the academics in the audience, and it was interesting to watch their reaction to the papers. Classic papers from William Mitchell[2](on a shape grammar for structurally stable variants of Laugier's primitive hut), and Charles Eastman[3] (a database for design concepts, for example those concerning a chair) could hardly fail to interest, but what about the the general run of KBS and Shape Grammar papers – were they intelligible or relevant , or even potentially relevant? Or were they more impressed by Durisch and Anderheggen's[4] work on 3D montage, John Danahy's[5] stunning system for the 3d realisation of landscape projects (from the University of Toronto) or (best of all) Bakergem and Obata's[6] analysis of the far-reaching consequences of loosening the the pen-holder on a plotter, so that the lines produced become squiggly? All three seem to exemplify the computer as a graphical medium rather than a thinking machine.

Richard Coyne[7] of the University of Sydney attacked the fundamental notions of computational design by dividing the world into the computable (rational, objective, scholarly and methodological) and the non-computable (romantic, subjective, mysterious and advancing by the creative leap) – and asking which had the greater affinity to architecture. He did not answer the question, but left it hanging in the air. One had the uneasy feeling that the researchers answered it one way, the practitioners the other. Aart Bijl[8] reminded us that computers do not think, but are better regarded as a medium of communication.

State of the Art

Architectural CAD has been possible for over twenty years. What has been achieved? Nowadays all large offices and a sizeable minority of small ones have some CAD capacity. Nine times out of ten this means AutoCAD. It is used primarily for production drawings, in much the same way as a word processor is used for letters and reports. Its impact on the quality of architecture coming out of an office is about the same as that of the secretary's word processor – which is very little. Its contribution to efficiency also matches that of the word processor – ie a little, rising to quite a lot when a document has to be reissued with revisions. Its contribution to the quality of life in the office is probably negative, as it distances the architect from the drawings, and demands the employment of a peculiar new kind of technician.

A secondary use of CAD is for the building of 3D models which are then rendered as presentation drawings. The process is laborious, and requires operators with special skills, who translate preliminary sketches or drawings into models. This is arduous, but fits quite naturally into office life, because the production of perspectives and presentation models has long been in the hands of specialists.

A third use of some importance in the bigger offices is the coordination of the work of a multidisciplinary team. AutoCAD is not well-conceived for this task (MicroStation or GDS would be better) Used in this way, CAD enters into the structure of the work, and offers a real benefit, primarily in Quality Control.

So CAD is offering something to the efficiency and production quality of an office, which is worthwhile. But the galling thing is that, after 20 years, CAD contributes little to design, and computers are not used by designers, at least not when they are designing.

This situation is widely recognised, by both practitioners and researchers, in both industry and academia. There is less agreement about the solution. Is intelligence the answer, or something else?

Theories of Architecture

One might hope that architectural theory would offer something solid on which to develop an approach to CAD. Unfortunately there is no Unified Theory of Architecture, but a plethora of theories, and even one or two meta-theories to help us classify the theories. Each generation of architects has its favourites, which oscillate in and out of favour over the centuries. For example, Sir Henry Wotton wrote in 1624

“The end (of architecture) is to build well. Well-building hath three Conditions: Commodity, Firmness, and Delight” [9]

This is a direct translation of Vitruvius[10] (about 60BC), and re-emerged to be a favourite of English architects of the post-war generation. Is this because of the satisfying rotundity of its 17th Century phraseology, or the simple memorability of a rule in three parts, or because it puts Functionalism first? To me it looks uncannily like the start of a goal-driven rule-base for architecture. But it is hardly sufficient. What about decorum (Alberti), power (Neitzsche), truth (Ruskin and the Prince of Wales), significance, meaning, economy, extravagance or even the “the masterly, correct and magnificent play of masses ...” (you know who)? Do you doubt that architecture means something, that it carries a message? If you look at an unfamiliar building, you may not be able to discern the precise message, though you can undoubtedly tell how loud it was shouted. The message is not (usually) from the architect, but from the person or institution that caused the building to be erected. The architect’s role is like that of the political speech writer, to put it across with a proper degree of force or erudition or humour, and a skilful choice of words.

These kinds of things are outside Sir Henry’s three rules, are not explicit in a design brief, and are probably not directly verbalised by either client or architect. Yet they have a dominant effect on the design, and on its acceptability to the client. They influence, perhaps determine, the mass of a building, its location, the way it addresses it neighbours and the landscape, its planning, detailing, materials and craftsmanship. In other words, everything visible. They even determine how much attention is to be paid (if any) to each of the Vitruvian concerns: commodity, firmness and delight.

I conclude from this that architectural CAD should be predominantly visual. Following Rasmussen[11], it should be able to manipulate or simulate solid and void and plane; scale proportion and rhythm; light, colour and texture; and ultimately acoustics.

These things are of prime importance to all architects. Individuals may have theories or preoccupations in favour of functional planning, energy conservation, structural optimisation and so on. There is certainly room for these things. But a CAD system (at least if it is to gain wide acceptance) had better, on first approach, be visual, and theory-free. I think this implies rule-free and knowledge-free as well.

Theories of Design

I am concerned that KBS community bases its theory on an over-optimistic simplification of the nature of design. To pick a convenient example, Carrara, Kalay and Novembri[12] characterise design as:

“1. Defining a set of functional objectives that ought to be achieved by the design artefact.

2. Constructing design ‘solutions’ which, in the opinion of the designer, are (or should) be capable of achieving the predetermined objectives.

3. Verifying that these solutions are internally consistent, and that they achieve the objectives.”

Now this is just about completely unrecognisable as a description of what really goes on in architectural design. The trouble is in the first step. Architectural objectives usually include functional ones, but are dominated by less definable intentions, and are never collected, in any complete sense, before design starts. They are evolved and discovered as the work proceeds.

I do not think this is due to sloppiness on the part of architects. It is the nature of the problem. Or rather, it is because we are not dealing with a problem at all, but an act of creation. A problem is something you find in a mathematics text book. All the data is given, someone has worked it out beforehand, there is a definite right answer. Architecture is not like this. The data is not given, nobody has done it before, and you cannot tell if you have got it right (you do not win every competition, do you?).

I do not believe that architecture is unique in failing to conform to the define-construct-test idea. Software design is just the same, despite the intense efforts of the gurus of software engineering to tell us differently. The so called “waterfall” method proceeds in this way. It may succeed with absolutely routine tasks (yet another pay-roll system), but it only produces innovative results if the programmers are bright enough to cheat. So nowadays we prefer “rapid prototyping”.

The same applies in architecture, except that prototypes are too costly. Computer modelling and simulation is the next best thing. And as in the software case, the evaluation of a prototype is made first by the designer, then by the rest of the team, and lastly by the jury or client or users. Evaluation is predominantly done by inspecting the “look and feel”, rather than by formal evaluation against predefined objectives.

Production and Creation

It is worthwhile to distinguish two kinds of design, routine production and innovative creation. Production is the design of a familiar kind of object, having small and identifiable differences from its precedents - for example the detailing of a utilitarian fire-escape staircase. In this case both the objectives and the design procedures are identifiable, and readily reduced to an algorithm. In the case of creative design (let us say, for contrast, the grand entrance to an opera house) neither the objectives nor the procedure can be identified in advance.

Knowledge-based systems are clearly adapted to production. Architects aspire to creation.

Of course, plenty of buildings, and parts of all buildings, are routine . Retail chains tend to specify their outlets so completely that fully automatic design is conceivable. Gas stations are an extreme example. “Design and build” contractors who employ a standardised proprietary method of construction can benefit substantially from knowledge based systems for working out the constructional details of their buildings, cost estimation, and documentation. Hospitals are a more interesting case, as the functional programme tends to be very fully specified at the outset, with a mass of detailed guidance about planning and the layout, servicing and equipment of particular spaces.

Bespoke Knowledge-based systems have a value for these production oriented tasks, when a series of buildings of similar type and construction are to be built for a single client. There is a danger that they will focus attention too strongly on the functional, economic and constructional aspects of the design. Creativity shifts from the project architect, to those who write the standards, the knowledge bases, and the engines for exploiting them. The opportunity for a sensitive architectural response to a particular brief and site is diminished.

Data modelling

A few years ago Earl Mark observed that “intelligence” in a CAD system was inversely related to flexibility. Here flexibility means the ability to represent a wide range of buildings; intelligence refers to the amount of design assistance offered. Mark’s observation is very widely true; perhaps it should be enshrined as the First Law of Data Dynamics. I suspect the reason has to do with a central problem in data modelling. To make any progress, you have to pin down a “Universe of Discourse”. The more detailed you wish to make the data model, the more circumscribed is the Universe - and so flexibility is lost. It is also worth observing that an intelligent system is inherently more complicated to understand than a flexible one, simply because there are more entities in its data model.Whenever we set out to make a CAD system, we have to choose a point on the continuum between intelligence and flexibility. At the CADLAB we are aiming low, believing that simplicity and flexibility are much more likely to gain acceptance than complexity and intelligence.

CADLAB Agenda

Our brief is to investigate and develop CAD techniques that might be of practical importance within five years. We must take into account trends in hardware and software technology, the weak points of current commercial CAD, and the aspirations of users. We work by building prototypes, which are demonstrated to our numerous visitors. We are guided by their reactions. If something seems irrelevant, incomprehensible or unworkable, we drop it. If our visitors want to buy a copy, or ask if we will help on a current project, then we are encouraged to continue.

We want to appeal to architects, especially the talented front-end designers who have so far seen little advantage in CAD. We want to make a positive contribution to the practice of architecture. And for the reasons outlined above, we think the “medium of design” paradigm is going to be more productive than the “thinking machine”.

So rather than deep intelligence, we are focussing on the “superficial” – visual appearance, the dynamics of interaction, immediate feedback, plasticity. We utilise clean mathematical abstractions (such as solid modelling), rather than ones that attempt to capture cognitive categories. We believe in simulation, provided that it happens in real time, and the result is presented visually. We do not ask our users to do anything text-based (such as writing rules, shape grammars, programming, understanding numbers), or anything that requires elaborate planning ahead to make it work.

Technology

Interactive 3-dimensional design requires fast graphics. We are aiming a few years ahead, so need to consider what the typical graphics-equipped PC of 1985 might be like. Memory costs are diving, so a z-buffer is certain, plus a very high rate of coordinate transformation , Gouraud shading, texture mapping and so on. The Silicon Graphics Indigo of today is a reasonable approximation. So one theme is the exploitation of this category of hardware, for example by elaborations of the standard z-buffer algorithm.

Another theme is Object Oriented programming (using C++), where we aim to establish a methodology for large scale multi-person development . We find object orientation entirely natural for CAD system design. We aim to import as much software as possible from other sources. So far we have integrated X-Designer (for building Motif interfaces), the ACIS solid modeller kernel (distributed by Spatial Technology), an object oriented geometrical modelling framework called Inventor (from Silicon Graphics), and a scan-line rendering package called LightWorks. We are on the look-out for an Object Database.

None of this has been especially hard to do, though C++ is an inelegant vehicle, rather hard to learn, and (at least in the Silicon Graphics environment) not very good for prototyping. Importing these substantial libraries leads to an inconveniently large program. Part of the reason is that each package we import brings with it its own classes for collections, vectors, elementary geometry and so on. C++ badly needs some internationally standardised object libraries.

Model Building

Elaborate three-dimensional computer models are often made for presentation purposes. The process is nearly as time-consuming and costly as making physical models. The design has to be worked out in detail before modelling starts. The model builder is usually a highly trained specialist, the software functional and precise, but hard to use without a good grasp of three-dimensional coordinate systems, working planes and axes rotations. The design architect may employ someone to use this software for him, but is rarely tempted to use it himself. If he uses anything, it will be a basic application like ModelShop which is simple, speedy and direct, but with restricted geometry, poor reliability, and shabby presentation.

Our aim is to produce something that is geometrically rich, but simple, visual and direct, attractive for design “in the round”. The starting point is boolean solid modelling, which is straightforward in concept (join bits together, make holes), and free from petty geometric limitations. Solid modellers are very difficult to write, but that is no drawback in these days of re-usable Object programming. We import the ACIS modelling kernel, which is a fast-evaluating analytic boundary-representation modeller incorporating NURB surfaces.

The primary problem in 3D work is how to achieve 3D input using a 2D pointer on the screen. Our solution is quite simple: you get what you are pointing at. We present the model fully surfaced (using fast z-buffer rendering) – no wire frames – and have a cursor which is at all times sensitive to the surface it is crossing. It shows this sensitivity by visibly orientating itself onto the surface, giving an almost tactile quality to the interaction. It is easy to draw a line on a surface, or move two surfaces into contact (both in position and orientation). It becomes quite natural to drag a table across a floor (no danger of it rising into the air), or slide a picture across a wall. Most model building can proceed in this way, by placing one thing on another. Where, rarely, it is necessary to locate something in free space - the answer is to erect some scaffolding. In simple cases this happens automatically.

The cursor is also sensitive to edges, vertices, extrapolations from these. and a few other salient points in the model. So the exact alignment of objects is the norm; user action would be required to nudge a position out of alignment. This reverses the usual “snapping” protocol, where the default behaviour is approximate location, and special action has to be taken to achieve exact alignment. If numerical precision is required, you “eyeball” the move first, then type in the correct dimension. Of course, everything is UNDO-able.

We place great emphasis on direct manipulation. Objects are constructed, moved and rotated by dragging with a mouse, and change visibly as you do so. While it is easy to drag a solid object in real-time using z-buffer rendering, one of our more surprising innovations is a technique for dragging holes around, so that you can peer into a closed box to see what is inside, or drag a window around a wall, seeing what is outside. This is done by a modified z-buffer algorithm, not by continuously re-evaluating the solid model.

Another z-buffer algorithm allows a section plane to be dragged through a model in real time, to reveal its internal detail. This is a quite simple exercise in computer graphics, but a radically new and effective way of communicating architectural arrangement.

Experiencing Architecture

Immersive visualisation technology, using wide-angle head-mounted displays, has enormous potential in architecture. However, it will be a little while before this potential is realised. Current displays have inadequate resolution, and affordable graphics engines cannot render fast enough to deal with a realistically elaborate architectural model. In the meantime, we have to be content with conventional screen display, where a somewhat jerky presentation of movement is more tolerable. There are plenty of research problems that can be investigated while we wait for the hardware to catch up.

The issue we are tackling, is how to convey the maximum amount of information about a model building, so that the degree of surprise that you have when you enter the real one is minimized. It is well known to anyone who has studied architecture that photographs do not convey very much about what it feels to enter the real building. Good buildings exceed our expectations, poor ones look worse in real life than in their photograph.

Immersive displays should convey much more than photographs, if only because they let you explore a model, but the average demonstration is not very impressive. Movement and velocity control is difficult, you crash through walls and floors in a thoroughly disorientating way, there is no sense of scale, of how big things are, or how far away. The experience is thoroughly un-architectural, and it takes a long time to learn to walk.

We have been looking at techniques to help establish a more realistic movement through buildings, leading to a sense of "body image" – how big spaces and features are compared to oneself. First of all, speed of movement is controlled to about 1m/s – or a footstep for each click of the mouse button. Then a collision detection algorithm prevents you from passing through surfaces in the model. Finally, an elaboration of the collision detector keeps your feet on the floor. No more flying. If you want to go upstairs, you have to find some stairs to go up.

In combination these developments make it possible for most first-time users to negotiate his way around a building. Sense of scale is improved, but still not very good. Texture mapping is helpful here. It is hard, with a screen display, to see exactly where you are. For example, if you ask someone to walk to a doorway, and stand at the threshold, they do not judge it accurately, as there is no peripheral vision. This will have to wait for the head-mounted display, or something equivalent. Try walking downstairs while looking through the viewfinder of your camera, and you will get some idea of how distancing (rather than immersive) the screen display is. The best technique is to lean against the bannister, and feel for the steps with your feet, which is roughly what the collision-detector does for you.

As an aid to debugging the collision detector, we implemented an alternative mode of display, where the perspective viewpoint is a metre behind your head, and the body volume is drawn as a translucent blue box. You can see yourself moving around. The curious thing about this non-realistic display is that it is easier to control, and gives a better sense of where you are, and how big things are, than does the accurate view centred at your eye position. The displaced viewpoint compensates for the lack of peripheral vision, and the blue box helps you to see your feet, and measure things against your body.

One interesting point about the collision detector is that it works purely from the surface geometry of the model. There is no semantic labelling – no differentiation of walls, floors, ramps and staircases. No intelligence. If you walk through a window you will fall to the ground, if there is any. If not, you fall forever.

Making drawings

When presentation drawings are made by computer, the first step is to build a three dimensional model, and the second is to render it to get a more-or-less photo-realistic image. This can take nearly as long, as view points have to be found, materials and textures applied, modelling defects eliminated, and the lighting set-up refined. Each change requires a trial rendering, which may, towards the end, take hours to produce.

This commonplace scenario provokes a couple of thoughts. First of all, when photos are notoriously bad at communicating buildings (and traditional architectural renderings have never striven to look like them), why do we place so much emphasis on photo-realism? Secondly, what is the operator actually doing during this long process? He appears to be striving to achieve an image that is more or less defined in his mind, by a complex and tedious process of altering the input parameters to an algorithm which provides far from instant feedback. Could he not achieve what he wants more immediately, by directly manipulating the image? Sometimes he does; it is not unusual to see architects post-processing their computer renderings, sometimes by drawing over them, sometimes by compositing and filtering in an image processor such as PhotoShop. Architects have a definite need to spend time "massaging" their images. It is part of their process (see for example Ian Fraser's [13]account of the behaviour of Frank Lloyd Wright).

Our work on architectural drawing applies some lateral thinking to these issues. We have drawn some inspiration from Bakergen and Obata's Wiggly Pen Plotting, Perlin's[14] proposed pixel processor, and the work of Paul Haeberli[15]. At its simplest, the technique is to grab a conventional, simple rendering, such as the walk-through software produces in a fraction of a second, and reprocess it. The grabbed image is sampled at random pixel positions, the RGB value inspected, and some sort of mark put down in the output image. The mark is usually bigger than a pixel, such as a stippling dot, a hatch line, or some larger "brush stroke". The mark may have a good deal of internal variability, in its width, length, straightness, colour, transparency, and so on. It is a bit like an automatic paint program, and indeed similar techniques have emerged in recent commercial applications, particularly FractalPaint.

Early results were quite encouraging. First of all, the images were interesting, even attractive, to the architects we showed them to. Secondly, it was apparent that the same technique could be applied to a scanned photograph. If applied to a montage, it unified its components immediately by redrawing them in the same "style". Thirdly, the amount of structured "noise" in the image compensated for lack of detail in the original model; rich images could be obtained by simple means. Fourthly, the images could have a soft, indefinite quality, which is much more appropriate when presenting a design concept which has not been worked out in any great detail.

The next stage was to add attribute flags to each pixel, and use different "sketchers" to process different materials, such as sky, grass, foliage, and various building materials. This produced rather indigestible images, and showed up some problems when hatching came into conflict with the perspective of an image.

We broke off at this stage to conduct some systematic investigations of individual aspects of drawing. We used a simple ray-traced scene containing flat, curved, and doubly curved surfaces, and rendered it using many different techniques. The techniques were conceived as a matrix, one dimension being the type of mark, the other the way that it responded to light. Marks included stipples, blobs, and various kinds of hatch lines. Response to light included change of mark colour, transparency, thickness, length and spacing.

The problems we had with hatching earlier, encouraged us to experiment with using the gradient of the light field to orientate lines. This tends to emphasise the form of curved surfaces quite powerfully. Different results can be obtained by orienting hatch lines along the gradient, at right-angles to it, or at some intermediate angle.

The next step, which had been long anticipated, was to grab the z-buffer along with the pixel brightness data. So for each pixel, we know how distant it is. With a little extra information about the perspective projection, it is now possible to generate hatch lines receding to a vanishing point, and to alter technique with distance. Edge-detection applied to the z-buffer produces sensitive line drawings, with emphasis of contours, and lightly drawn interior edges.

These experiments have produced a wealth of usable techniques with qualities ranging from steel engraving through lithograph to delicate pencil drawing. (It is not particularly our intention to imitate traditional media, but we can hardly avoid using them as a point of reference). The technique can be entirely automatic, or accept any degree of manual intervention. We are currently assembling a user interface that will allow an architect to massage his image, drawing, redrawing, overdrawing until he is satisfied with what he has discovered. We imagine that he will spend anything between a few minutes and an hour or two, depending on the closeness with which he interacts with the process.

Conclusion

There is a general consensus that CAD has not fulfilled its potential in architecture, and in particular is not used, or even usable, in the early conceptual stages of design, when creativity has its greatest scope, and the crucial decisions are made. What must CAD offer in order to surpass the traditional media - pencil and tracing paper, and modelling materials? We doubt that the answer lies in manipulating text-book stuff like knowledge of buildings, precedents, rules, and design intentions. We need to work with the visual world of diagrams and drawings, solids and spaces, but not their significance and purpose, which are better left to the mind of the beholder. The medium does not have to understand the message. The primary qualities we need are simplicity, informality and readiness-to-hand. Beyond that we need to develop qualities that engage the hand and the imagination, that promote exploration and discovery. Again these are the qualities of a medium, like tactility, flexibility, plasticity, and transformability.. Only then will we begin to support the "thinking with the hands" and "deep imagining" which (at least here in Cambridge) we strive to develop in our architects.

Acknowledgements

The work summarised here has been carried out by Simon Schofield (Making drawings), John Rees (Experiencing architecture), and Tim Wiegand (Model building, and making everything else hang together). Numerous visitors to the Martin Centre have helped, but particular thanks are due to Richard Coyne and Ian Fraser for their stimulation in depth. We are very grateful to the architects who have contributed real projects, especially Nicholas Ray, John Hare and Roger Matthews. And finally, thanks to Masanori Nagashima and Informatix Inc, for their enthusiasm and long-term view.

Figure Captions

Note to editor: Figures are supplied as a strip of 35mm transparencies. They are expected to be reproduced in monochrome, and quite small, eg 6 to a page of "Automation in Construction"

Frame	Fig	Caption
1	1	Watching yourself go up stairs. The transparent box represents your body, the clash finding algorithm ensures that is does not penetrate surfaces, but does rest on one. Forward motion results in it climbing the stairs without difficulty.
2	2	A dynamic sectioning plane reveals the internal arrangement of a building.
3	3	The 3d cursor picks up the underlying surface, and responds to its curvature by orienting itself according to the outward normal. This kind of responsiveness gives an almost tactile quality to the interaction.
4	4	Direct manipulation in three dimensions.

5	5	Dragging a cylindrical hole through a solid. This novel form of direct manipulation leads to all sorts of unexpected geometrical discoveries.
6	6	Shadow volumes rendered by a modified z-buffer algorithm
7	7	Conventional rendering (by Wavefront), with natural elements (tree and sky) re-rendered with marks.
8	8	Exploratory interface for the re-drawing system. If the mouse is used, marks are made at the cursor position, otherwise at random positions following a "drunkards walk". The orientation of marks is "taught" by drawing an example with the mouse; other qualities are controlled by the sliders.
9	9	Wavefront image re-drawn using the interface of Fig 8. It takes about five minutes to make a lightly-detailed image like this.
10	10	Portrait of the artist, re-drawn from a video frame.
11	11	Each pixel is tagged to indicate whether it is building, ground, sky or vegetation, and then redrawn with an appropriate algorithm. The image is a failure for a number of reasons, of which the most obvious is the conflict between the cross-hatching and the perspective of the building.
12-19	12a to 12h	A series of redrawings of the same ray-traced image, using different kinds of marks, and different ways of responding to the brightness of the original, such as by changing orientation, spacing, size, darkness and transparency.
20	13a	Hatching oriented perpendicularly to the direction of greatest change of brightness tends to trace out contours of equal brightness, and so form rings around a highlight on a curved surface.
21	13b	By orienting hatching at 45 degrees to the gradient, the sphere appears "hairy", whereas a smooth spiral is formed on the cone.
22	14	A z-map. which records for each pixel how far away it is from the viewpoint. Here it is drawn as a greyscale image; dark areas are close, and light ones distant.
23	15	Hatching by depth. The technique is basically the same as that in Figure 13a, but using the rate of change of depth rather than of brightness to control the orientation of lines.
24	16	Hatching responding to surface orientation.
25	17	Noisy hatching adds complexity to an image with scant detail.
26	18	Edge detection applied to the z-map yields a sensitive line drawing. Contours are emphasised, while arrises and mouldings are drawn more lightly.

27	19	Warsaw, derived from a ModelShop block model.
28	20	Villa Stein, by Le Corbusier.

[1] Richens, Paul. The OXSYS system for the design of buildings. *Proc CAD78.* Pergamon (1978)

[2] Mitchell , William et al. Integrating Shape Grammars and Design Analysis. *Proc CAAD futures 91.* Vieweg (1991)

[3]Eastman, Charles..Use of Data Modelling in the Conceptual Structuring of Design Problems. ibid

[4]Durisch, Peter and Anderheggen, Edoardo.. Leaving the Planar Universe ibid

[5]Danahy, John. The Computer-Aided Studio Critic: Gaining Control of What We Look At. ibid

[6]Bakergem, W Davis van, and Obata, Gen. Free Hand Plotting - Is It Live or Is It Digital. ibid.

[7]Coyne, Richard.. The Impact of Computer Use on Design Practice ibid.

[8]Bijl, Aart. On Knowing - Feeling and Expression ibid

[9]Wotton, Sir Henry. *The Elements of Architecture.* London (1624).

[10]Vitruvius..*The Ten Books on Architecture* tran M H Morgan. Harvard (1914).I (iii)2

[11] Rasmussen, Steen Eiler.. *Experiencing Architecture.* MIT Press(1959)

[12]Carrara, Gianfranco, Kalay, Yehuda E. and Novembri, Gabriele. *Knowledge-based computational support for architectural design.* In the present volume.

[13]Fraser, Ian. *The Architecture of Drawing.* in press

[14]Perlin, Ken.. An Image Synthesizer. *Computer Graphics* 19(3) ACM SIGGRAPH (1985)

[15]Haeberli, Paul. Paint By Numbers: Abstract Image Representations..*Computer Graphics* 24(4).ACM SIGGRAPH (1990)