EXPERIMENTS IN COMPUTER AIDED DESIGN

Report from the Department
of Architecture
Massachusetts Institute of
Technology

SPACE ARRANGEMENT

Tim Johnson
A. G. H. Dietz
Guy Weinzapfel
Richard Krauss
David Morris

Sponsor: National Science Foundation

It is our goal to develop a machine-aided design system that will become a natural part of the design process.

Since a machine will only solve problems stated in quantitative terms, it is necessary to restate architectural considerations in such terms; we are concerned with finding those architectural relationships which can be given a mathematical description.

To begin with, we have taken the attitude that architecture is primarily concerned with the uses of space, and have chosen to take as the basic abstract element of design a unit allotment of space for a particular activity.

At the outset we will be working with a TV screen that will only show plan displays, and so the unit allotment of space will appear in plan, and is called a 'use surface'. As such it represents a bounded activity in two dimensions. To the machine the 'use surface' is a set of mathematical variables which describe the dimensions, orientation and location of a space, a volume, not just a surface. The use surfaces may be drawn, proportioned and moved on the display screen with a hand held light-pen.[1]

Those relationships between use surfaces which can be given a mathematical description will be formulated as constraints on the dimension, location or orientation of use surfaces in three dimensions. A series of constraints applied to several use surfaces will be the machine description of the total architectural problem. Once such use surfaces and constraints have been described, the machine can generate a set of feasible solutions, through routines that will seek the optimal one that best satisfies the series of constraints.

Since design problems, and even design methods, cannot be anticipated, the program will permit the designer to tailor the system to his needs, without re-programming the machine in the ordinary sense. This is accomplished by programming a suitable set of *atomic* constraints that can be combined into an application dependent *molecule* by the designer. For example, visual access, physical access, orientation, linear ratioing, adjacency and proximity are included in the set of atomic constraints. The architect may make up a 'path' molecule (using his light-pen) that is composed of physical access, proximity and visual access, if he prefers circulation paths that meet these three criteria. If the designer prefers a modular arrangement of spaces because of a given structural system, he may combine the linear ratioing constraints into a 'bay' molecule to meet this criterion. Thereafter, the architect may deal with his personal set of molecular constraints that satisfy his methods of design.

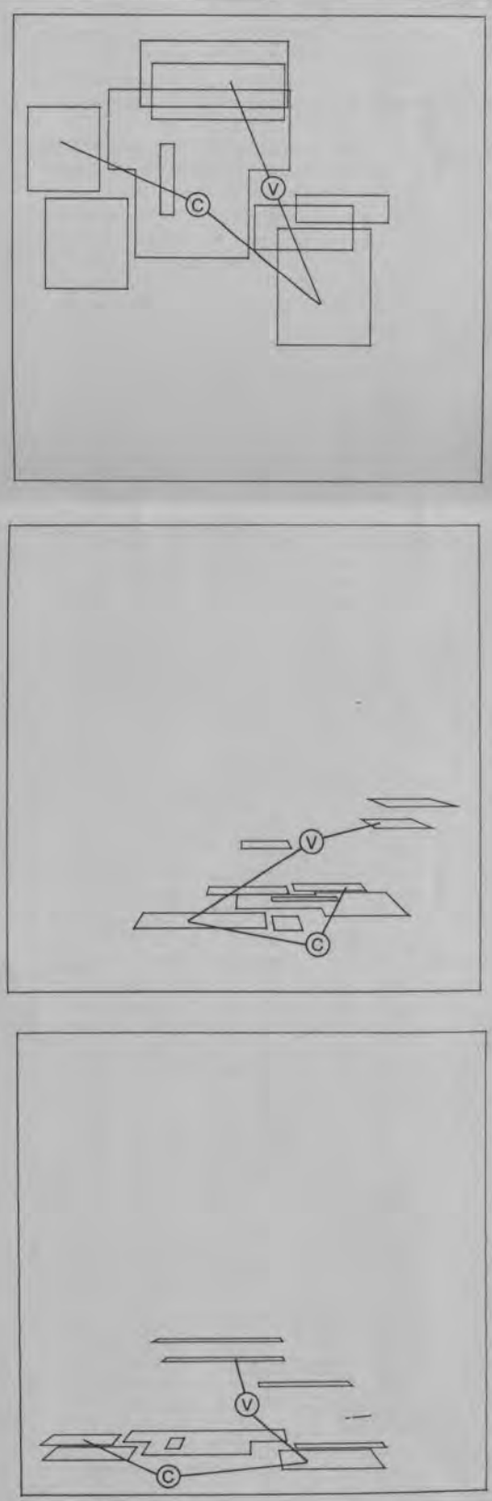
In order that the designer can make comparisons, and other judgments about these constraints with respect to each other, each constraint can be weighted. Thus to what extent one constraint will be given priority over another, or applied before another, will be determined by the weighting.

The system will be an interactive one. It will act as a kind of educational device for the designer; as he requests it to perform tasks, he will become acquainted with the ramifications of the problem as he has set it up in the machine by seeing what is and what is not possible in accord with his instructions and by seeing what their consequences are. As the designer learns more about the problem he may add more constraints as he sees fit, or remove some that have been applied, or discard the problem as originally stated and start again. When it is doubtful that a constraint really had application to the problem, it can be modified or eliminated. In this way the designer learns more about the problem.

Architects are primarily concerned about the evolution of form in response to requirements, so the computations and the results of computer analysis will be displayed graphically. As the machine progresses toward a solution, a gradually changing image will be displayed. Care will be taken to make all operations visible: the aim is to have the designer understand the reason behind every change, and to know what the dynamic effects are of the judgments he has made. The designer will be able to control directly the evolution of his design; he will be able to reject or select any of the solutions displayed, or will be able to change any of the parameters (or their values) which affect the design.

The forms the machine displays will be directed only by instructions activated by the designer; any aesthetic judgments the machine can make will be those that the designer can articulate sufficiently for use in this system. □

[1] 'Sketchpad III, a computer program for drawing in three dimensions', T. E. Johnson, Architecture Department, MIT, Cambridge.



Typical cathode-ray-tube presentation of the generalized space allocation program using the Sketchpad III program for graphic communication. Shown are a plan (top) and two perspectives of a use-surface arrangement. All the constraints acting on a use surface may be displayed by pointing at the surface in question and pressing a button. In this illustration, the connection (C) and visual access (V) constraints have been made visible.

DISCOURSE AND CHOICE

Discourse: William Porter

Katharine J. Lloyd
Wren McMains
Aaron Fleisher

Choice: John Boorn

Stanley Hoderowski
Aaron Fleisher

Sponsor: Urban Systems Laboratory at MIT and IBM

DISCOURSE and CHOICE are compatible computer systems which are intended to serve city planners and designers and others concerned with the urban physical environment. The DISCOURSE system is primarily for generating and testing environmental design ideas, whereas CHOICE is primarily for evaluation.

The two major criteria governing the design and development of the two systems have been:

1. That they accommodate the kinds of data and manipulation designers use, or might use, and that they accommodate changes and additions quickly and easily; and
2. that they permit the designer to develop his own language of design as he is working.

Neither criterion would have been possible to achieve without the existence at MIT of computer 'time sharing' systems in which a central computer is connected to user consoles (which resemble typewriters) and from which the user can expect nearly instant response.

The first criterion we have attempted to achieve in a way compatible with the other three by using relatively simple data structures. For example, in DISCOURSE, the designer could put the *attribute* 'industry' at the locations '21,33; 22,33; and 23,33' (referring to the row and column numbers of a grid he had constructed and laid on top of a map of the environment he was working on).

He could further specify that there were 400 employees in industry at that place, a number which might vary with different locations, and that those who worked in that type of industry earned an average of $8000 per year—a number which is constant over all *locations*. All elements of DISCOURSE have both names and values which the designer can establish and change himself. The description will vary with the fineness of the user's grid of *locations* and with the amount of information he puts in each *location*.

The user can display his information by showing the names and values of the elements and by mapping them on the geographic grid. Presently the 'maps' are typed by the console, but work is under way to utilize cathode ray tubes for continuous and special displays as well as printers of faster and more elaborate hard copy.

CHOICE provides three kinds of matrices in which the data can be stored: primary, value and order matrices. The results of tests of several design alternatives can be entered into the primary matrix. The values which various groups place on the test results can be placed in a value matrix, a new set of values obtained by multiplying the two matrices together, and from the results the preference of each group could be computed. The design alternatives can then be ordered according to the preference of each group.

The specialized data structures ease the bookkeeping problems for the designer so that his attention can be focused on the things he wants to do. In CHOICE the designer may perform complex cost/benefit analysis with respect to more than one group or individual. The characteristics of the designs to be evaluated may be directly specified by the designer or assessed from other simulation or design models. The designer may input these consequences in either an ordinal or cardinal form, thus allowing him to represent judgments as well as measured parameters. He may combine cardinal and ordinal representations to obtain a variety of measurement scales. The use of multiple accounts allows the designer to represent time-series outcomes. Other operations in CHOICE permit computing discounted present values, statistics of mean and standard deviation, and expected values. The flexibility of the system permits the designer to rapidly alter evaluation arguments such as weighting assumptions, probabilities, interest rates, and definitions of the accounts.

In DISCOURSE, the user can manipulate names and values and their locations and can add new ones or transform those he already has described. Furthermore he can implement rules about how the environment might develop through the use of such features of the language as statements which branch to other statements and statements which are executed conditional on the truth of other statements. He can store the rules he has written and call them with a single name. And he can store the latest design scheme he has worked out either to work on some other time or to evaluate with the CHOICE system, perhaps against other schemes he has previously worked out.

For example, a designer might wish to study possible locations for industry in the outlying areas of a large city by generating alternative locations according to different decision rules, and then testing the locations against other criteria. The results of certain tests such as the increase in the tax base for the community can be coupled with the importance which various population groups attach to those results in order to evaluate the proposals. The designer can implement any of the operations involved in this example by using DISCOURSE and CHOICE. He may find, over time, that he builds up a library of generators, tests and evaluators which are particularly useful.

Rather than being substitutes for the hard thinking which is involved in describing, changing and evaluating environments, DISCOURSE and CHOICE require a complete understanding of every aspect of what is done with them. We argue that what is describable is programmable; and we argue the converse of the contrapositive as well: that what is not describable is not programmable. Let us hasten to say that it cannot be inferred from the above that the computer therefore can yield neither counter intuitive nor non-trivial results. Clearly one can describe procedures which man could only carry out in a very long period of time and which might or might not yield surprises.

Our insistence that the designer be able to create his own ways of working poses a fundamental issue for us and for the user: if we enrich the systems to the extent demanded by the subtle user, we make the systems difficult to use, and we ask of the user that he think through design issues which, if he were to design in the traditional manner, he might not have to face. The results are that the systems now will slow down the user, and that they will force him to work in an unfamiliar way. A second issue we face is how to determine whether there has been any improvement over what the designer might have done without the systems. A third issue is whether we are able in the design of computer systems to give sufficient play to the non-verbal component of design: how much of design involves formal ideas which cannot be described in any other than visual terms?

And, finally, what capabilities of design should we be stressing in the development of the system? Which should man do and which the machine? And, can the last question be answered identically for all designers, or even for any designer over time?□

[1] *Discourse: A language and system for computer assisted city design*, Ph.D. Dissertation by W. Porter in preparation in the Department of Urban Studies and Planning at MIT.

[2] *A choice system for environmental design and development*, J. P. Boorn, Ph.D. Dissertation, Department of Urban Studies and Planning, MIT.

ARCHITECTURE MACHINE

Nicholas Negroponte

Leon Groisser
Anthony Platt
Richard Hessdorfer
Steven Gregory
Paul Mockapetris
Mark Drazen
Stephen Peters
Paul Linsay

Sponsors: Ford Foundation and Interdata, through the Urban Systems Laboratory of MIT

What makes a human designer a 'good' architect? We know that he must somehow contribute physical environments that both house and stimulate the good life. But we do not know much about the good life (it has no 'utility function' and cannot be optimized). We know that he must have an understanding of, and ease with, physical form. But we do not know how our own cognitive processes visualize shape and geometry. We know that he must interpret human needs and desires. But we do not know how to describe these needs and desires.

What probably distinguishes a competent designer is his ability to provide missing information. Any environmental design task is characterized by an astounding amount of unavailable or undeterminable information. Part of the design process is, in effect, the procurement of this information. Some is gathered by doing research in the preliminary design stages. Some is obtained through experience. Other chunks of information are gained through prediction, induction and guesswork. Finally some information is handled randomly, playfully or personally.

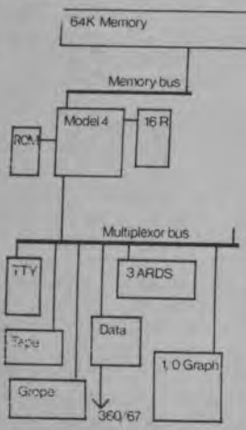
It is reasonable to assume that the presence of machines, of automation in general, will provide for some of the omitted and difficult-to-acquire information. However, it would appear foolish to suppose that, when machines know how to design, there will be no missing information or that a single designer can give the machine all that it needs. Consequently, the Architecture Machine Group at MIT, are embarking on the construction of a machine that can work with missing information. To do this, an Architecture Machine must understand our metaphors, must solicit information on its own, must acquire experiences, must talk to a wide variety of people, must improve over time, and must be intelligent. It must recognize context, particularly changes in meaning brought about by changes in context.

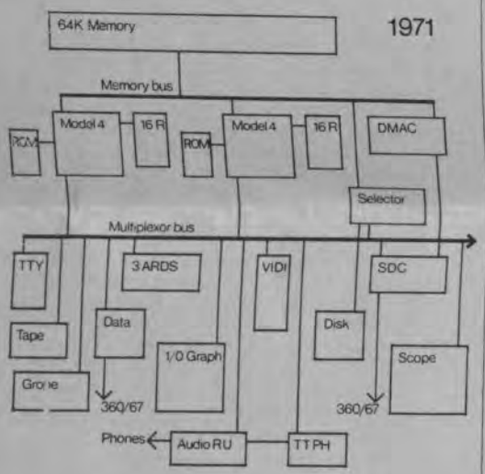
We should take advantage of the professional iconoclasms that exist in our day—a day of evolutionary revolution. We should build machines equipped with at least those devices that humans employ to design. I suggest that we build machines that can learn, can grope and can fumble; that we build machines that help architects, not replace them.[1]▷

[1] From N. Negroponte, *The architecture machine*, forthcoming, MIT Press.

PROGRAM

◁Following a year of experiments and exercises in 'artificial design intelligence', the Architecture Machine Group will start constructing a satellite machine which will be the beginnings of an Architecture Machine. This device will be primarily composed of Interdata processors and Interdata memory. Its task will be to learn about architecture.
The following illustrations represent three particular experiments that have in common the specific goal of providing Architecture Machines with the ability to solicit information on their own.
The two diagrams are estimates of the first two years of growth.□





TOWARDS A HUMANISM THROUGH MACHINES

Nicholas Negroponte

Given that the physical environment is not in perfect harmony with Everyman's life style; given that architecture is not the faultless response to human needs; given that the architect is not the consummate manager of form and use; let us consider the evolution of physical environments. In particular let us consider an evolution aided by the use of a specific class of machines, as Warren McCulloch calls them, ethical robots. In the context of architecture, we shall call them Architecture Machines.

There are three possible ways of having machines assist the design process:
1. Current procedures can be automated, thus speeding up and reducing the cost of existing practices;
2. Existing methods can be altered to fit within the specifications and constitution of a machine, where only those issues are considered that are supposedly machine-compatible;
3. The process, considered as being evolutionary, can be introduced to a mechanism (also considered as evolutionary), and a mutual training, resilience, and growth can be developed.

We shall consider only the third alternative and shall treat the problem as the intimate association of two dissimilar species—man and machine—and two dissimilar processes—design and computation. We shall further define our concern as the acquaintanceship of two intelligent systems, the architect and the Architecture Machine. By virtue of ascribing intelligence to an artefact or the artificial, the partnership is not one of master (smart, leader) and slave (dumb, follower), but rather of two associates which each have the potential for self-improvement.

Imagine a machine which could respond to local situations in the physical environment (a family that moves, a residence that is expanded, income that decreases). Such a device could report on and concern itself specifically with the unique and the exceptional. In effect, it would concentrate on the particulars. The human designer cannot do this. He cannot accommodate the particular; he obliges the general. Britton Harris suggests, 'He is forced to proceed in this way because the effectuation of planning requires rules of general applicability and because watching each sparrow is too troublesome for any but God.' The reader surely needs little reminder of the results.

Prelude to an architect-machine dialogue
Consider that you are in a foreign country, do not know the language, and are in desperate need of help. At first your hand movements and facial expressions carry most of your meaning to the silent observer. Your behaviour uses a language of gestures and strange (to the observer) utterances to communicate your purpose. The puzzled listener searches for bits of content he can understand and link to his own language. You react to his reactions and a language of pantomime begins to unfold. This new language has evolved from the mutual effort to communicate. Returning to the same person a second time, with a new need, you will find that the roots of a dialogue already exist. But this second conversation might be gibberish to a third party observing the exchange for the first time.

A designer-to-machine introduction should be a similar linguistic evolution. Each should track the other's design manoeuvres, evoking a rhetoric that cannot be anticipated. The event is circular since the designer-machine unity provokes a dialogue and the dialogue promotes a stronger unity. This progressively intimate association of two dissimilar species is the symbiosis. It evolves through mutual training—in this case, through the dialogue.

A man-machine dialogue has no history. The historical and presently antagonized mismatch (non-dialogue) between man and machine has generated a great deal of preoccupation for it. In less than a decade, the term 'man-machine communication' has passed from concept to cliché to platitude.

The theory, however, is important and straightforward: in order to have a cooperative interaction between a designer of a certain expertise and a machine of some scholarship, the two must be congenial. They must, for example, share the labour of establishing a common language. Thus a designer, when addressing a machine, must not be forced to resort to machine-oriented codes. And in spite of computational efficiency, the paradigm for fruitful conversations must be machines that can speak and respond to natural language.

With direct, fluid, and natural man-machine discourse, two former barriers between architects and computing machines could be removed. First, the designers using computer-aided design hardware would not have to be specialists. With natural communication, the 'this is what I want to do' and 'can you do it' gap can be bridged. The design task would no longer be described to a *knobs and dials* person to be executed in his secret vernacular. Instead, the job would be formulated and executed in the designer's own idom, with simple negotiations. A vibrant stream of ideas could be directly channelled from the designer to the machine (and back).

The second obstruction overcome by such close communion would be the potential of re-evaluating the procedures themselves. In a direct exchange, the designer could exercise his proverbial capriciousness. At first, a designer might have only a meagre understanding of his specific problem and thus require machine tolerance and compatability in his search for the consistency among criteria and form and method, between intent and purpose. The progression from visceral to intellectual could be articulated in provisional statements that converge on both design and methods.

But the tête-à-tête must be more direct and fluid; it is gestures, smiles, and frowns which turn a conversation into a dialogue. In an intimate human-to-human dialogue, hand-waving often carries as much meaning as text. The 'manner' carries cultural information: the Arabs use their noses, the Japanese nod their heads. Customarily, in machine communication studies, such 'manners' are ignored and frequently are referred to as 'noise'. But such silent languages are not noise; Warren Brody and Nilo Lindgren submit that a dialogue is composed of 'whole body involvement—with hands, eyes, mouth,

URBAN 5, a primitive effort at a man-machine dialogue useful in architectural design.

facial expressions—using many channels simultaneously, but rhythmized into a harmoniously simple exchange.'

Imagine a machine that could follow your design methodology and at the same time discern and assimilate your conversational idiosyncrasies. This same machine, after observing your behaviour, could build a predictive model of your conversational performance. Such a machine could then reinforce the dialogue by using the predictive model to respond to you in a manner that is in rhythm with your personal behaviour. This dialogue would be so intimate (even exclusive) that only mutual persuasion and compromise would bring about perceptions and ideas—ideas, in fact, unrealizable by either converser alone. In such a symbiosis, it would not be solely the designer who would decide when the machine is relevant.

The overlaying of a specific design character upon a generalized machine is not fanciful; computer programs exist that illustrate some primitive attempts. An anonymous machine, after identifying a speaker, can transform itself into an exclusive apparatus that reflects previous encounters with that speaker. The extent of the metamorphosis depends on the length of acquaintance. At the onset of the partnership, the machine gathers gross features; later it avails itself of subleties.

It might be argued that we are proposing the creation of a design machine that is an extension of and in the image of a designer who, as he stands, has already enough error and fault. However, we have indicated that the maturation would be a reciprocal ripening of ideas and ways. At first, jobs in which the man is particularly inept would stimulate a non-trivial need for cooperation. Subsequently, each interlocutor would opt out of situations notably clumsy for his constitution, while at the same time he would pry into issues which were originally outside his scope of concern (or the concern of his profession). Eventually, a separation of the parts could not occur; the entire 'symbiotic' system would be, as Gordon Pask described, 'an artificial intelligence that cannot be partitioned.' It would be computer-aided design.

Computer-aided versus computerized

'Computerized' operations are too often misnamed 'computer-aided.' The computerized/computer-aided distinction is too often confused with or solely embodied in the mode of machine usage.

The conventional mode of computer usage for the past 20 years, 'batch processing', entails a computation centre to which a user delivers a 'program' (a deck of cards, magnetic tape, paper tape) to be 'run'. Then several hours or days later the user returns to receive his 'output'. More recently, real-time computation depending upon 'time-sharing' techniques allows the user a prompt machine response and permits terminals (usually teletypes) to reside in the office or at home. These terminals are connected to the large central machine, and they can be interconnected with each other. The rapid switching of users' programs in and out of the large machine provides each user with the illusion of a dedicated machine and permits him continual use of his terminal.

It is commonly suggested that the on-line nature of the interaction in a time-sharing system is in itself a dialogue and transforms computerized procedures into computer-aided ones. This is simply not true. For example, let us suppose you desire the average apartment-to-parking-space distance for some design project. In a batch processing mode (presuming the program exists), you supply as data the description of your design and the answer returns hours later—indeed a computerized procedure. On the other hand, in a 'real-time' environment, the project description resides in the machine, and you simply type on your teletype terminal the apartment-to-parking distance command.

But just because the answer comes back in three seconds rather than three days, 'computerized' does not become 'computer-aided'. It simply becomes more convenient. Computer-aidedness demands a dialogue; events cannot be merely a fast-time manifestation of cause and effect.

On-line communication, therefore, is not a sufficient (though necessary) condition for a computer-aided environment. Computer-aided design requires three further features: mutual interruptibility for man and for machine, local and dedicated computing power within the terminal, and a machine intelligence.

Interruptibility gives a dimension of interaction that allows the process, as well as the product, to be manipulated. In a computer-aided system, the machine may interrupt the user and present unsolicited information—for example, that the cost of his low-income housing project is $38 per square foot. The apparent high cost might have been due to inadequacies in the original estimating routine provided for the computer; for instance, substantial indirect savings might have been overlooked. In this case, the designer could tamper with the estimating routine and incorporate hitherto neglected parameters. Thus the architect might welcome the remark, ignore it, or take offence and request that such interludes of finance be restricted.

Unfortunately, the present time-sharing philosophy fosters a cause-and-effect conversation. Time-sharing assumes that a designer's explicit manipulations will occupy between 1 and 10 per cent of any sitting; the remaining time represents his deliberations and distractions. Each user's moments of contemplation are in effect another user's instants of computation. A designer can interrupt his own program, but a routine cannot easily interrupt its partner-in-thought. This is because, in order to leave the computational utility available for other users, each routine resides in the machine only when explicitly called into service by its particular user. In other words, the user's machines—are encouraged to listen, but not to interrupt.

To retain assets of 'time-sharing', while avoiding the anathema of batch processing, and to acquire mutual interruptibility, we must adjust the allocation of computing power. Some information processing power and a certain manipulative and storage capacity must be transferred to the terminal that was originally a teletype transmission and receptive device. This semi-autonomous terminal (possibly portable) would become a small computer that would be a 'machine in residence'. An Architecture Machine would be such a machine. The designer would speak directly to this satellite machine. In turn, this small remote computer would interactively converse with larger parent machines. (Sending out work to a central mechanism would not be apparent to the designer; his Architecture Machine would elect this course for reasons of speed, memory, information, or all three.)

It is the Architecture Machine in residence, located in habitation with the designer, which would undergo the personalization. It would be composed of additive and subtractive pieces of hardware as determined by the discipline of its partner. The local consortium of parts would do the interrupting, the dialoguing, and the evolving. Observe that the interrupting and the re-interrupting would be dependent on the nature of the designer's activities, on the context of his efforts. Through familiarity with the specific designer's idiosyncrasies, the appropriateness of the machine's interruptions can be suitably reinforced by context; this is the inception of an intelligent act.

A mechanical partner, as we have suggested, must have intelligence. Customarily, computer-aided design studies and intelligent automata studies have been antipodal efforts. On the one hand, we are told to render unto each their respective design functions and talents: man (intelligent) thinks and the machine (dumb, fast) calculates. On the other hand, we are told that, 'Anything you can do, a machine can do better.' The two outlooks are not necessarily contradictory. There is a real issue whether machine intelligence can be independent of human intelligence. In computer-aided design, only the combination of mechanical amplification and mechanical imitation will validate the dialogue. The dialogue will evolve an intelligence, this intelligence will stimulate a more profound dialogue, which in turn will promote further intelligence, and so on. Furthermore, the concurrence of 'extended designer' and 'artificial designer' will force a design redundancy and an overlapping of tasks which are necessary for the understanding of intricate design couplings. Perpetual cross-examination of ideas by both the man and the machine will encourage creative thought that would otherwise be extinguished by the lack of an antagonistic (thus challenging) environment. Computer-aided design concerns an ecology of mutual design complementation, augmentation, and substitution.

Vis-à-vis machine intelligence

Intelligence is a particularly difficult behaviour to emulate in machines because of its extreme dependence on context—time, locality, culture, etc. A sophisticated set of sensors, effectors, and processors is needed to view the real world (directly or indirectly) and to discern changes in meaning brought about by changes in context—in other words, to be intelligent.

For example, the meaning of a metaphor (in the physical environment, in a story, in a painting) is conveyed through context, and the assessment of such meaning is an example of an intelligent act. (Note that a literary metaphor characterizes the era and the culture in which it was written.) One might judge a machine's intelligence (not necessarily its maturity, wisdom, or knowledge) by its ability to appreciate a joke —a joke being a funny story with a punch line that is an about-face in context. As humans we exhibit an intelligence by tracing back through previous metaphors of the tale and deriving pleasure from the new meanings revealed and brought on by a shift in context. (Note that people of different cultures have difficulty in understanding each other's jokes.)

Some architects might propose that machines cannot design unless they can think, cannot think unless they want, and cannot want unless they have bodies and, since they do not have bodies, they therefore cannot want, thus cannot think, thus cannot design—*quod erat demonstrandum*. The argument, however, is usually an emotional issue rather than a logical conclusion. Nonetheless, the reader must recognize (if he is a 'machine-intelligence' enthusiast) that theories on machine intelligence can, at this time, best be supported with such examples as computers playing a superb game of checkers and a mediocre game of chess. And furthermore, architecture, unlike checkers (with fixed rules and a fixed number of pieces) and much like a joke (determined by context), is the croquet game in *Alice in Wonderland*, where the Queen of Hearts (society, technology, economics) keeps changing the rules.

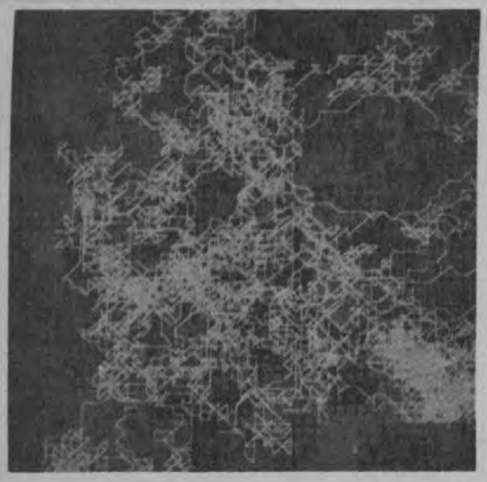
Let us not be misled. We are not interested in a machine that will simply parrot a human designer, nor are we interested in a machine that will have an autonomous existence by which to mimic and replace an architect. An Architecture Machine will feature a dependence. An artificial intelligence is in fact an interdependence.

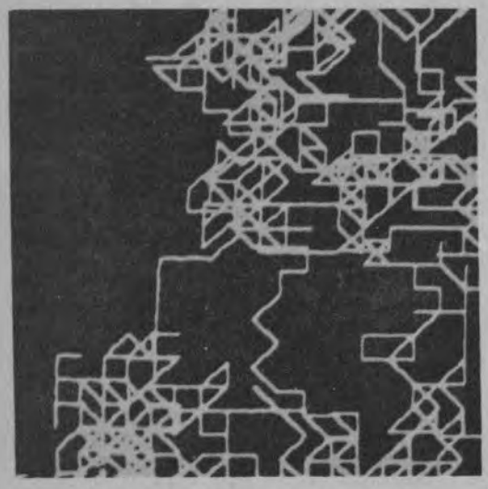
You might summarize the proposition here as being: For machines to contribute to an environmental humanism, they must have a 'natural dialogue' with a human designer, natural because they need his metaphors and natural because they need his ideas unmutated. The dialogue, in turn, must reside within a computer-aided (not computerized) system. The system, consequently, must include a 'resident processor', some real-world sensors and effectors, and an intelligence.□

Reprinted from *Technology Review*, Vol. 71, No. 6, April 1969.

GROPE

A second example of interfacing with the real world is Steven Gregory's GROPE. *GROPE is a small mobile unit that crawls over maps, in this case, Passonneau and Wurman's* Urban Atlas (*MIT Press, 1966*) *maps. It employs a low resolution seeing mechanism constructed with simple photocells that register only states of on or off, I see light or I don't see light. In contrast to the Platt experiment,* GROPE *knows nothing about images, it deploys a controller which must be furnished with a context and a role (as opposed to a goal).* GROPE*'s role is to seek out 'interesting things'. To do this, the little robot compares where he has been to where he is, compares the past to the present (plus occasional random numbers to avoid ruts) to determine future moves. The onlooking human or Architecture Machine observes what is 'interesting' by observing* GROPE*'s behaviour rather than receiving the testimony that this or that is 'interesting'. At present, aspects of* GROPE *are simulated on scopes and other aspects use the local computing power on* GROPE*'s back (for relays).*□



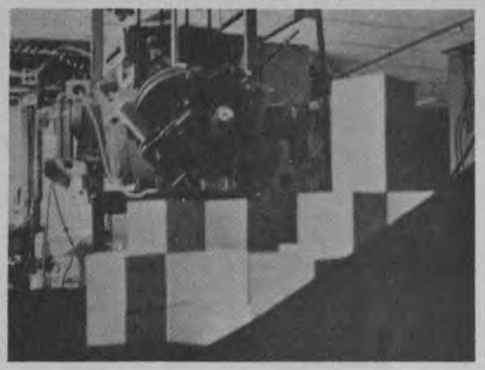


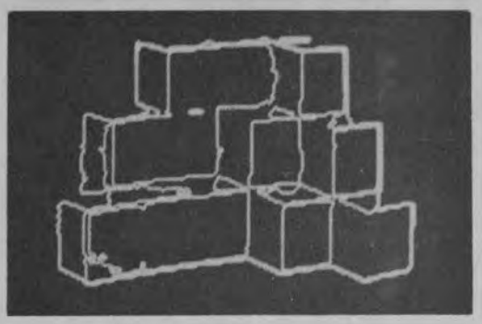
INTERACT

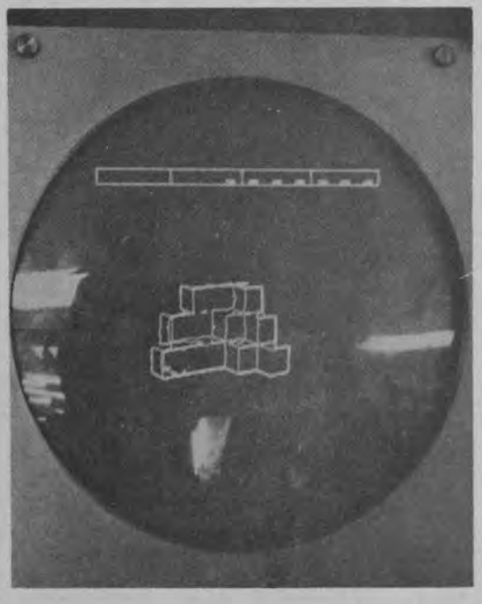
INTERACT faces the problem of soliciting information about the environment, about needs and desires, from the inhibitants themselves. Using English input, Richard Hessdorfer is developing a 'consumer' item that could initiate a dialogue with inhabitants, build a model of needs and desires (particular to the speaker) and report back to Architecture Machines.
As a part of the Hessdorfer experiment, a teletype-writing device was brought into the South End, Boston's ghetto area. Three inhabitants of the neighbourhood were asked to converse with this machine about their local environment. Though the conversation was hampered by the necessity to type English sentences, the chat was smooth enough to reveal two important results. First, the three user-inhabitants said things to this machine they would probably not have said to a human, particularly a white planner or politician: to them the machine was not black, was not white, and surely had no prejudices. Second, the three residents had no qualms or suspicions about talking with a machine (in English about personal desires); they did not type uncalled for remarks, instead they immediately entered a discourse about slum landlords, highways, schools, and the like. (The reader should know, as the three users did not, that this experiment was conducted over telephone lines with teletypes, with a human at the other end, not a machine. The same experiment will be rerun shortly—this time with a machine at the other end of the telephone line.)□

SEE

SEE is an exercise in machine vision, giving Architecture Machines eyes. Anthony Platt is presently applying the vidisector (the seeing mechanism) of MIT's robot project, asking it to look at simple physical models. The interim goal is to observe, recognize and determine the 'intents' of several models, built from plastic blocks. The machine looks at the models and extraplates certain characteristics and criteria that might have led to it. Following a certain number of witnessings, the machine is asked to generate its own solution from that which it learned (and which is visually representable).







COMMENTARY

Stanford Anderson

An outsider to these MIT projects can best raise only certain general questions. 'What do architectural designers wish to ask of the computer sciences?' is primarily a question in design theory. 'How should design problems be put to the computer?' is a question that combines issues in design and in logic. Such questions should remain open to all concerned students and designers.

'How can computer science meet the problems set by architectural design?' is a question for computer specialists. The three questions interact, of course. One is simply observing that a computer is not being used for its own sake, and that non-specialists should continue to engage certain aspects of computer-aided design.

What the designer wishes to ask of the computer sciences depends on our understanding of architectural design. A possible formulation,[1] emphasizing the open-ended, exploratory character of design, is the following. Architecture structures man's environment to facilitate the achievement of human purposes (intellectual, psychological and utilitarian) where those purposes are incompletely known and cannot be extrapolated from what is given in the situation. Rather, human purposes are altered by the very environment that is created to facilitate them. The structuring of the environment must be accomplished, then, through the exercise of tentative foresight and the critical examination of that foresight and the actions to which it leads. According to this description, neither the human purposes nor the architect's methods are fully known in advance. Consequently, if this interpretation of the architectural problem situation is accepted, any problem-solving technique that relies on explicit problem definition, on distinct goal-orientation, on data collection, or even on non-adaptive algorithms will distort the design process and the human purposes involved.

Any such characterization of the design process raises a variant of the classical 'problem of induction', and thus challenges how we can reasonably put design problems to the computer.

In logical terms, universal theories cannot be derived from reports of a finite number of observations.[2] 'All swans are white' is not a safe generalization from any number of observations of white swans. Put crudely, but more to our point: a finite number of observations cannot yield a design hypothesis that can reliably be expected to embrace any subsequent observation. A related issue that can be stated even more pointedly is: one false observation transmits its falsity to any generalization that encompasses it. Incompleteness of information can, and inaccurate information does, lead to false inductive generalizations.

In the light of this logical problem, the first of the three MIT projects presents the greatest difficulties. That space arrangement project relies on a finite (for now, very small) number of observations and certain combinational rules. Those observations will always be incomplete: some important design considerations may not be consciously articulated; others will only be known in the future; certain observations, excluded because formerly considered of negligible significance, may assume importance in a new context. This incompleteness, plus the likelihood of inaccurate observations, make it likely that the optimization routines will conduct searches in areas that bear little relation to the actual problem. The designer may or may not be able to recognize that discrepancy at the time of its appearance on the scope. Even assuming he does, his next effort will be plagued with the same problems.

Such an enterprise presents a further difficulty. It tends to obscure the fact that the selections of observations and of combinatorial rules are theory-impregnated. That is, it is on the basis of some general notion that these selections are made. Working from the level of detailed characterization and with several programmers and designers giving inputs, it is even likely that the system is impregnated with several, quite possibly conflicting, design hypotheses.

I would argue that such design hypotheses should be brought to the forefont of attention and operate as the guide for detail decisions. Only in this deductive manner can each observation and choice be made in concert with; or intentionally in contrast to, the design hypothesis; only in this way will one realize when the design hypothesis is inadequate, incapable of satisfactorily subsuming detail decisions.

This last analysis implies that the designer must enter the computer-aided design system with a general formulation and progressively use the critical tools of the designer and machine to move towards an end state: either a satisfactory design or a recognition that the general formulation is inappropriate for the problem. Within this approach the designer maintains constant awareness of the general position being advanced and he is constantly learning the detail considerations that must be met—considerations that might never have occurred to him had he not had the general formulation before him both as idea and as form.

The Architecture Machine Group at MIT has undertaken a highly speculative project, the results of which can only be intuited. However, this group incorporates two notions that are important from the point of view of this commentary. The man/machine interaction begins with a form generalization by the designer. A battery of critical tools then impinges upon that form hypothesis: the designer's own critical ability can alter both his form proposal and the machine programme; designer-programmed criticism can perform repeated reviews of the changing form proposal; and, it is proposed, machine-generated criticism can provide an antagonistic environment that challenges the designer on grounds other than those he has established.□

[1] Offered, as part of a broader analysis, in my paper 'Problem-solving and problem-worrying', Architectural Association, London, 1966; also the Cranbrook conference, American Collegiate Schools of Architecture, 1966.
[2] The problem of induction and related issues are discussed by W. W. Bartley III, 'How is the house of science built' *Architectural Association Journal* (Feb. 1965), pp. 212–18.

Contributors (pages 478 to 514)

Abel, Chris: Born 1941. Studied at Bristol CAT 1958–1960, Berlin 1960–62, and 1966–68. Now doing research in urban planning.

Anderson, Stanford: Born 1943 in Minnesota. Studied at Berkeley and Columbia Universities, now teaching architectural history at MIT.

Brodey, Warren: Previously at MIT, has recently established the Environmental Ecology Laboratory in Boston, Mass.

Chalk, Warren: Born 1927, London. Studied Manchester College of Art, worked at LCC 1954–63. Joined Archigram in 1961. Currently writing.

Churchman, C. West: Born 1913. Associate Director and Professor of Business Administration, Space Sciences Laboratory, Berkeley.

Cowan, Peter: Reader in Architecture at University College, London and Director of the Joint Unit for Planning Research, London.

Drew, Robert Lionel: Born 1925. Studied Bristol. Has been concerned with the teams to sponsor, commission and operate complex technical projects. Author of the Solway project, *Tn. Plann. Inst. J.* 1965.

Greene, David: Born 1937, Nottingham. Studied Nottingham. Started *Archigram* magazine with Peter Cook in 1961. Teaching at Leicester and working with Archigram.

Greene, Judith: Born 1936. Studied Oxford and London. Lecturer, Birkbeck College, London since 1966. Researching in psycho-linguistics.

Johnson, Tim: Born 1939. Studied University of Michigan. Now lecturer at MIT.

Kamnitzer, Peter: Studied Harvard and Columbia. in the architectural firm Kamnitzer/Marles. Assoc Prof at School of Architecture and Planning at the Univ of California, LA.

Landau, Royston: Born 1927. Studied University College and AA, London. Worked in architects offices in the USA for five years. Now in private practice in London. Has recently published *New directions in British Architecture* (1968).

Lakatos, Imre: Born 1922, Hungary. Studied Budapest, Moscow and Cambridge. Professor of Logic, LSE, University of London.

Musgrave, Alan E.: Born 1940. Studied LSE; now lecturer in Logic, Philosophy and Scientific Method at the LSE.

Negroponte, Nicholas: Born 1943. Studied architecture at MIT, now Assistant Professor in the Department of Architecture, MIT.

Pask, Gordon: Professor in the Institute of Cybernetics, Brunel University, Research Director of System Research Ltd, Surrey.

Popper, Karl Raimund: Born Vienna 1902. Studied University of Vienna. Professor Emeritus of Logic and Scientific Method, University of London. Author of *The logic of scientific discovery* (1957), *The open society and its enemies* (1945), *The poverty of historicism* (1957), *Conjectures and refutations* (1963).

Porter, William: Born 1934. Studied Yale Now Assist Professor of Architecture, MIT.

Price, Cedric: Studied at Cambridge and the Architectural Association, London. Taught part-time at the AA 1958–61. In private practice,

Rabeneck, Andrew: Born 1942. Studied Regent Street Poly 1958–65. Worked on Herts schools 1965–67. At UC, Berkeley, 1967–68. With Building Systems Development 1968–69.

Law-Whyte, Lancelot: Physicist. Born Edinburgh 1896. Senior Scholar Trinity College, Cambridge. Late '20's in Berlin as a Rockefeller Fellow in Physics. 1930's associated with investment bank as scientific consultant on financing new inventions. Later managing director Power Jets Ltd, which developed the Whittle jet engine. 1941–1945, Director of Statistical Enquiries, Ministry of Supplies. Has since devoted his time to writing and lecturing.

Wilson, Alan G.: Assistant Director at the Centre for Environmental Studies in London, formerly Head of Mathematical Advisory Unit in Economic Planning Directorate of Ministry of Transport.