Concept design games

N. John Habraken and Mark D. Gross

School of Architecture and Planning, MIT, Cambridge, Mass. 02139, USA

This paper describes our work on using games as a tool for research in design theory and methods. Games offer a means of isolating certain aspects, or concepts, of designing for purposes of scrutiny. A game provides an environment for a group of players, acting with individual goals and a shared program, to make and transform complex configurations, free of functional requirements. Adjusting game parameters emphasizes different concepts. We have developed nine games that explore a variety of concepts of general interest to those concerned with organizing physical configurations. Beyond these particular concepts, we argue that games are a useful way to couch studies in design theory and methods.

Keywords: games, design theory, design methods

This report describes our work on using games as a tool for research in design theory and methods. Using games as research tools follows naturally from our previous work in design methodology and theory as applied to the design of built environments[1–3]. By way of introducing our current efforts, therefore, we review a few key aspects of this earlier work.

Human settlements are complex artifacts. They are often large and may extend over vast areas. They differ considerably from one culture to another. They also are used over long periods of time, during which they can be subject to dramatic transformations. Today even a single building is a fairly complex thing. It embodies an array of subsystems: the structure, various systems of partitioning, envelopes to shield it from the elements, and systems for heating, ventilation, electricity, water, gas and communications. The building must house many interrelated human activities that may vary considerably over its lifetime.

In studying the designing of buildings and urban environments we decided not to limit out attention to what a single designer does to a static artifact. In our investigations we observed two important principles.

- There are always many designers. The artifact to be made is designed in a process of cooperation and negotiation among many actors. The participating designers have different expertise and their responsibilities in the larger design task can be distributed in many ways.
- The artifact changes all the time. Human settlements are never finished and we keep designing them. Though each designer can finish his individual task, urban environments and also individual buildings continue to be designed upon throughout their lifetime. Most design work in architecture and urban design relates to artifacts already in place that must be added to or changed. Even a new building usually adds to an existing urban environment; hence the design of the building alters a larger object – the environment – of which it becomes a part.

Corresponding to these two principles, we introduced two new issues in our thinking about design theory and methods.

- Designing is a social activity that takes place among people who negotiate, make proposals, set rules for their conduct and for the work to be done, and follow such rules. In short, to a large extent, designing involves agreement-making and rule-making.
- Designing is about morphological change. Designers must understand the transformations of complex physical organizations. We believe that complex physical organizations have certain transformational properties in common, regardless of their function.

These two issues must be seen in relation to each other. The interaction between designers and the configurations they organize and shape is at the core of designing. This brings us to study how designers manipulate and transform complex configurations, while making agreements and rules as to how to go about their work. Much of our work results from this enlarged perspective. The methods we have developed in the past seek to improve the performance of design communities working on buildings and urban environments that change and adapt.

Concept design games

When designing involves the interaction of many actors, all involved with transforming an artifact, we see similarities with board games. In the games we describe, several players work with a configuration on a board. As when designing, players must fit pieces into an existing field; rules, conventions and principles limit how they may move; and they make flexible, negotiable arrangements about what conventions and rules to use in a given situation. Last but not least, players make projections for configurations to be constructed. In these ways our games resemble real-life design situations. Yet, in contrast to real-life experience, the games enable us to study design actions by providing an environment that is manipulable and well bounded.

One of the most difficult aspects of understanding designing has always been that too many divergent acts occur simultaneously, defying simple description. We found it useful to develop a set of games that each isolates and focuses a single aspect, each giving a clearer picture of what just some of designing is about. Seen this way, our games may appear remarkably one-dimensional, stressing one aspect while suppressing, or even leaving out entirely, other aspects of designing. However, our games always preserve the basic unity of the design experience in that a variety of actors work together in transforming an artifact. Thus our games incorporate the two issues of designing mentioned above.

We found that in a game we can bring forward aspects that represent concepts guiding our designing. When we communicate as designers we share certain concepts that cause us to act in certain ways; through these same concepts we also 'understand' each other's actions. Developing and playing games, we learn about the concepts we hold. That is why we call our games 'Concept Design Games'.

Figure 1 shows a simple example of using a game to illustrate a concept. Here we recognize the concept of 'dominance' by observing the players' moves. (Player A is said to dominate player B if A's moving a piece requires B to respond, but not vice versa. See Appendix II.) This is not a design game, obviously, but it may be the beginning of one. Rules creating a situation of dominance between players can provide a context for a game that also involves designing a configuration on the board to meet certain programmatic demands.

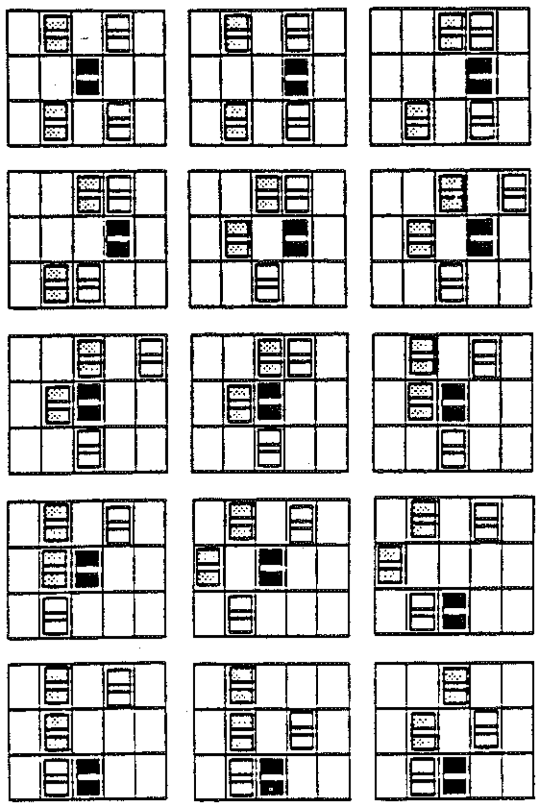

Figure 1. Use of a game to illustrate a concept

Understanding form, not function

The action-oriented approach we find in games stresses the role of the physical artifact. The configurations we engage with 'behave' in certain ways, and this behaviour shapes, to a significant extent, the way we organize ourselves as designers. When we use the term 'behaviour' in relation to the thing being designed, we mean the way the artifact allows or resists manipulation and transformation. Through studying change in the built environment we have become aware of how such form-behaviour influences our actions as designers.

As with all investigations of physical phenomena, observing change is the key to understanding inner structure. Because our subject – the artifact designed – is of our own making, the changes we observe are the result of design decisions. We find a dialectic situation: we shape the artifact, but in turn the artifact shapes the way we organize our design processes. These dynamics reveal hierarchical structure in the artifact and territorial structure in the organization of design responsibilities. The concepts of hierarchy and territory drive several of our games. These concepts are relevant to all parts and at all sizes of the built environment; to the room as well as the city, and to the construction detail as well as to the larger built framework. We do not know whether these principles also apply to other complex artifacts outside our domain, but other investigators may well find interesting similarities.

What we can learn in this way about physical configurations differs from the functionally oriented knowledge that designers usually express about their subject. Buildings and urban environments are designed by many specialists of markedly different interests and backgrounds. Functional knowledge is what separates them. The artifact that brings them together can only do so by means of properties that are not functional. In real-life designing, we rightly pay a good deal of attention to functional matters, but in our games we play with configurations that do not refer to any real-life artifact. Thus the games eliminate the functional knowledge designers usually bring to their work. This limitation, though severe, may help us understand the extent to which designing can be seen as a general activity.

A research tool, not a design tool

The concept games we have developed do not resemble reality the way simulation games do. The latter are based on functional knowledge and they succeed to the extent that they preserve real-life facts and real-life disciplines. In contrast, concept games extract an aspect from reality and enlarge it.

Concept design games are research tools intended to help us better understand designing. They do this by opening to scrutiny the concepts we use as designers, as well as the structures of the complex artifacts we manipulate. They are not meant to be tools for designing, nor are they made to help teach designing.

A game box

One product of our work is a 'game box' that will allow readers to develop their own games, shaped around concepts that interest them. The game box is our way of making operational the idea of design games as a research tool.

The game box contains:

- A basic terminology for discussing the structure of games.
- An explanation about how to create a 'technical universe' to play with, for each game, from a larger set of pieces. We also present a set of pieces of different kinds, out of which many such technical universes can be composed.
- An account of what we learn about game development, including a discussion of five strategic questions that game developers must address.
- A computer-based method to record game play.
- Nine games to serve as examples, dealing with a variety of concepts in which we were interested.

We found, not surprisingly, that by developing a game based on a particular design concept, we can learn much about the concept. Game development in itself is a powerful form of research. In our work, game development went hand-in-hand with game playing. As we developed each game we played it, modified the rules, and played it again. The most interesting games may never be finished because we keep changing them, probing new ways to see the subject, finding new ways to deal with it. The nine games we submit are therefore open-ended. We present a playable version of each game but also suggest variations on the game structure that might prove interesting. We found that each game opens a whole field of possible inquiry that we had no time to enter into. In this way the game box is a research tool.

While game development is a form of research, a finished game can serve as a research tool as well. The game is a stage for exploring design behaviour; it establishes a certain context. By playing the game we can learn more about what we can do in that explicitly constrained context. We can use games this way to study how designers negotiate, come to agreements and follow conventions. We may learn about how design teams approach problems, or we may study alternative design processes. We can compare different ways designers can organize themselves relative to a design context set up by the game.

The scope of our project, however, did not enable us to study the extent to which concept games can serve these uses. Nor did we submit the games to players outside our development team in order to gauge the games' accessibility. Much less could we explore the development of research programs in which the game would serve as a tool. However, from our experience with playing the games, we feel confident that they can be used as vehicles in researching design behaviour. Thus the idea of the finished game as a tool remains untested although we believe it to be most promising.

The game box is a tool for demonstrating and testing concepts about designing. Based on our previous experience in design theory and methods, we selected concepts that we knew to be of particular interest. The reader need not share our interest in these particular concepts to find the games useful. We hope indeed that others will use the same game box in quest of very different ideas. But to follow our work the reader must become acquainted with the concepts we chose to work with. Not all these concepts are generally familiar among designers. The next part of this paper presents several concepts in our games in summary form.

PROPERTIES OF CONCEPT DESIGN GAMES

The nine games we developed are very different both because the concepts that drive them are different and because we sought to make as wide a range of game forms as possible. It will not do to give a single game as an example because to understand the common properties of the games, one must see several different games. However, rather than presenting detailed descriptions of several games, it seemed better, by way of introduction, to discuss a few aspects characteristic of all of them.

Variable physical organizations

All our games are board games and thus are played with pieces on a board. We played with the parts that wooden clothes pins are made from (we call them 'pegs'), nails of different sizes and washers. We also used self-sticking labels bought in a stationery store. These pieces offered the variety of size, shape, material and colour that we needed and they have the advantage that they are readily available. Although we also made a more sophisticated set of pieces in acrylic, all our games can be played with the simple versions.

Each game uses only a selection of these available pieces and they are used differently in each game. An important part of developing a game is therefore making an appropriate 'Technical Universe' for it. The Technical Universe is the physical system out of which we design something. It is the sum of the constraints embodied by the pieces themselves and the allowed deployment relations, independent of the board they may be deployed on.

Not only can Technical Universes differ from game to game, several very different Technical Universes can serve the same game. This is because most games rest primarily on certain relational properties of the Technical Universe posing constraints to the players. Though we may play with configurations of different shapes, we may observe similar constraints.

Playing a game in different physical circumstances can clarify the concept under scrutiny. Figure 2 shows two configurations from different Technical Universes. Each can be interpreted as having two systems (black and white). Seen in the context of transformation rules, the systems operate on different levels in a 'dependency hierarchy' where the higher level system (black) dominates the lower level (white). It is possible to make the same hierarchical structure in each of the Technical Universes. In the first the relation between the levels is connection; in the second the relation is enclosure. To the extent that the hierarchical structure drives the playing, these two alternatives are equivalent.

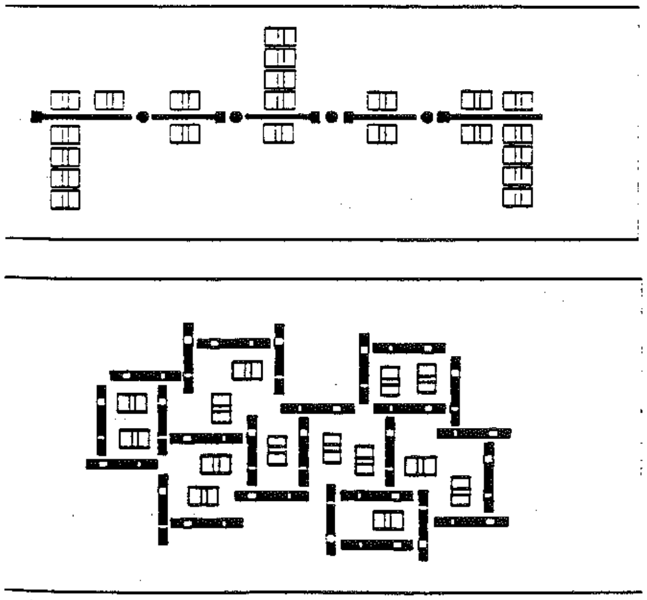
Figure 2. Two configurations from different Technical Universes

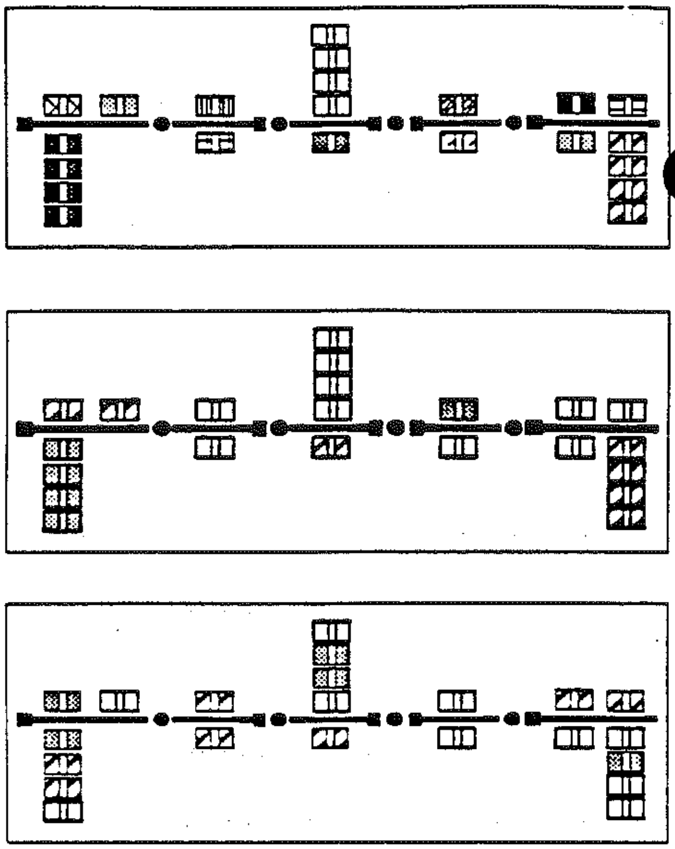
Figure 3. Alternative control distributions for the lower level of a configuration

Control distribution

The control of the players over a certain configuration on the board can usually be distributed in many ways. This contrasts with board games we are familiar with, such as Chess and Go, where the control distribution is inflexible; one party plays White and the other plays Black. Figure 3 shows alternative control distributions for the lower level of a configuration.

Control patterns over a given Technical Universe are an important variable in design games because they reflect the way we organize ourselves to work on a complex configuration. Depending on the purpose of the game, they may be set by the game developer or left open for the players to determine.

Territorial organization

For a player to manipulate part of the configuration, be it a local part or a subsystem distributed across the whole board, he must have space to work in. Parties operating simultaneously within a restricted space operate within a territorial organization. There are 'private' spaces a party can work in without interference from others and 'public' spaces where parties must share space.

As distinct from the control of *pieces* the territorial organization has to do with control of *parts of the board*. Understanding the territorial distribution in a given field for purposes of design can be important for design efficiency. Multi-levelled territorial hierarchies may occur. We can learn about territorial organization in complex design situations by observing processes of urban settlement and change. We translated two basic urban organizational patterns into two territory games.

The territorial organization adopted for a particular instance of play cannot be fully fixed because usually we do not know exactly how much space we need to do our job. The territorial boundaries must shift within a given hierarchy of territorial organization. In territorial organization, we find another variable of Concept Design Games. This variable too, may be set by the game developer, or left open, allowing players to choose alternative patterns of territorial distribution.

Figure 4 shows a fragment of one of the games in which territorial organization is part of the Technical Universe. The boundaries formed by nails have gates and enclose clusters of pegs. The territorial hierarchy is indicated by the letters at the gates, A being a higher level territory than B. Note that the physical organization is not one-to-one with the territorial organization: the same kinds of clusters can occur on levels B and C, depending on the number of gate crossings needed to reach them. In Figure 4, the clusters with gates B are in 'public' space relative to the 'private' space containing clusters C. In this example, territory is formalized as part of the game by the use of nails to mark boundaries. In most games territorial organization occurs informally and boundaries are invisible; nevertheless territorial organization can be used to subdivide and coordinate work on the board.

Program

Control over pieces and over parts of the board is common to all board games. As we have seen, in our

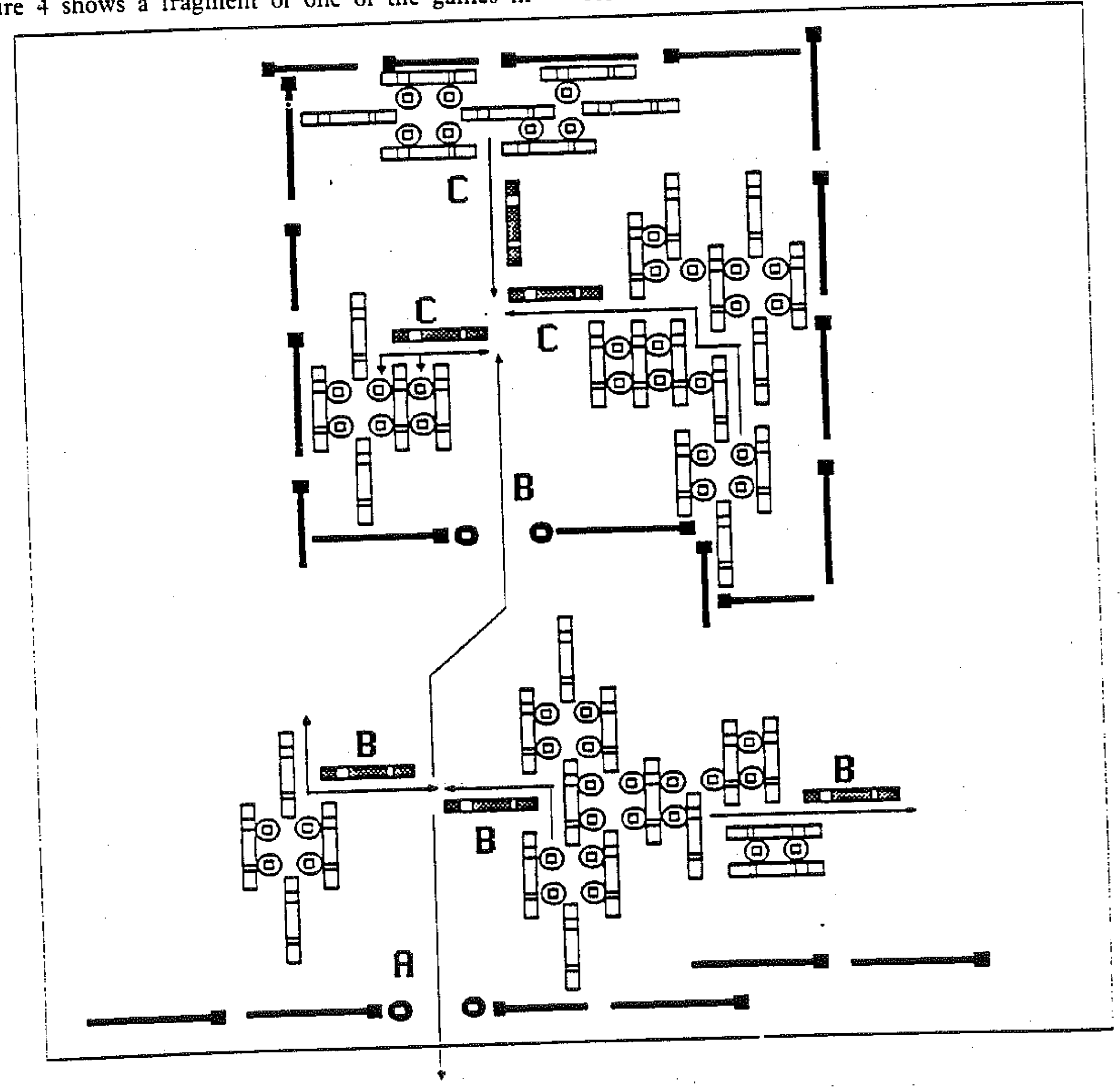

Figure 4. Fragment of one of the games in which territorial organization is part of the Technical Universe

design games these variables become active, subject to change in development and play. The idea of a 'program', however, is particular to design games. The program is a task common to all players. It is distinct from the 'goals' that individual players have in all board games, our design games included. The program describes certain board conditions that must be fulfilled to complete the game. Without a program the individual goals of the players make for a competitive game. The program leads players to also cooperate and plan together.

Because design games have a program they can never be purely competitive. Individual roles played in the games may compete with each other for reasons of scarcity of resources (pieces or space), or because of conflicting goals. But these must be reconciled with the common task and herein lies the source for negotiation and ultimate agreement and shared rulemaking. The pursuit of a common program can also bring tensions were players interpret the program differently or advance different ways to achieve it. The program brings into the game many interactions we associate with designing.

In-play development

Another aspect particular to design games is that the game may be developed as much by the players as by the initial creator of the game. The variables mentioned so far – control distribution, territorial organization, and program interpretation – can be available to the players and indeed these represent typical variables to design teams in all design tasks.

We can therefore say that the work done by the game developer is continued by the team playing the game. This is only natural because both engage in a design process; the developer-designer setting the stage for the player-designers. It is the game developer, however, who determines the degree of freedom given to players to engage in these activities.

Many players, many pieces

Although some of our games can best be played with just two players, it is in the nature of design games that tasks and organizations can proliferate. In most games we can bring a larger number of players and increase the size of the board and the number of pieces accordingly.

Even when only two players play they may find it of interest to work with large and complex configurations on the board. We are interested in interactive processes among designers because many design tasks today raise questions about controlling complexity. It is important that configurations on the board can, when needed, become sufficiently complex to reflect such conditions. To find the right balance between order and complexity, and between complexity and playability makes the game development a design challenge by itself.

RECORDING DESIGN GAME PLAY

Our main objective in developing a recording scheme was to assist game developers in analysing game play. In the course of developing a game we would play it several times, then find ourselves unable to remember in detail how the game went. After playing, we would propose and discuss modifying the rules to improve the game, but we needed to know what actually had occurred. We tried recording play by photographing the board after each move using an instant camera, but this proved difficult and distracting. We considered using a videotape recorder but this, also, seemed to involve too much extra effort, both in recording the play, and again later, in viewing the tapes. Videotape might be a good way to record the games if one wanted to observe *all* events: including those not directly pertaining to game play, but we merely wanted to record the moves made.

Games that forbid players from moving or removing pieces once placed build, in effect, a partial record of game play. The board at the end of play contains every move that was made. Tic-tac-toe is a simple example. In such games it may be possible to reconstruct the sequence of play, or at least plausible sequences of play, just by looking at the board at the end of the game. But in our concept design games it is usually impossible to reconstruct the sequence of moves from the final state of the board. Hence our need for a means to record moves as the game is played.

From the start, we thought to construct the game recorder as a computer program that could accept descriptions of games moves and display games states and histories on the screen. It was logical, therefore, to consider using the computer to develop and play the games as well as to record game play. Initially, however, we decided to develop our games outside of the computer. We wanted to play with physical pieces on a physical site. We wanted to ensure that our games could also be played without a computer. We thought of the game recorder as a tool in a larger game development process.

Recording and reviewing

The game-recording task is minimally construed as keeping track of pieces placed: added, removed or moved. This includes recording:

- what piece
- where it is placed
- who placed it
- when it was placed

Recording just this information would be useful for post-game analysis. In an extended version of the game recorder we may also want to record why the piece was placed, check whether the move is legal, and perhaps to keep score. Further, we may want the game recorder to keep track of rules, agreements and negotiations made among the players. Eventually we would develop the game recorder towards an environment for game play and game development.

The purpose of recording histories of game play are:

- to view and review previous game states, sequences of states, and entire histories of game play
- to restore previous game states
- to replay games from previous states

A history of a single instance of play is simply a sequence of moves and board states. When a game is replayed from an intermediate state a branch is added from that state: the sequence becomes a tree. The history now represents a set of alternative ways the game was played; with every replay, the history tree becomes bushier.

Notation

Chess notation offers a model for our recording scheme. Three column-inches of newsprint suffices to record all the moves in a game. Chess notation describes a process, the sequence of moves from opening to endgame. The two players' moves are listed in order. Each move names a piece, and describes its final board position: P–QB4, or, for a capture, it names the two pieces that engaged: P × P. Chess notation is sparse, as measured by the number of characters needed to record a move, or a game. It conveys each move using a small number of symbols.

In part, the elegance of Chess notation derives from assuming the reader understands the game rules. For example, to interpret the notation P × P the reader must know how pawns capture in order to determine which two pawns might be involved. If the notation did not presume this knowledge, it would need to describe the pawns' initial locations, and this would require more symbols. Thus if we devise a different game using the Chess board and pieces, it is most likely that we cannot use Chess notation to record play in our new game.

This last observation relates directly to our problem. We wanted to devise a means to record play across a variety of different games. As mentioned above, in some of our games the rules change, are left unstated, or are invented as a part of play. Unlike Chess notation, our recording scheme cannot rely on knowledge of the game rules.

Layout languages

From notation we came, therefore, to the idea of a layout or picture language. A layout language offers a means to describe and manipulate a two- or three-dimensional configuration of physical elements as a symbolic code. Once we have represented the layout in symbolic form, we can apply standard list-permuting and pattern-matching techniques, either to transform the layout, or to recognize, parse and classify its features. By using a layout language to describe the physical configuration on the game board we avoid integrating the notation system with the game rules.

Layout, or picture languages have been widely applied. Physicists devised picture languages to describe patterns of atomic particle scatter[4]. Layout languages are also used by VLSI designers in the layout phase of automated integrated circuit design[5,6]. The formal properties of layout languages have been studied[7]. A picture language can be recursive and purely functional[8]. The shape grammar formalism used in some studies of the built environment[9,10] to represent thematic variation in built environments is also related.

Writing form

We call our layout language 'Writing Form', or 'WF' for short[11]. The game recorder uses WF to represent board configurations. It is implemented as an embedded language in an object-oriented dialect of Common Lisp[12]. WF nouns refer to material and space elements; WF verbs refer to operations for arranging these elements in space. WF's primitive arrangement operation strings elements along imaginary lines. Each WF sentence denotes a configuration of material and space elements; the most elemental description of configurations are given in terms of edges and distances.

For example, the letter 'E' can be described in WF as a horizontal string of two elements: a vertical edge (e1) and a vertical string of three horizontal edges: two identical top and bottom edges (e2) and a middle one half as long (e3), with a distance (d1) between each pair of horizontal edges (see Figure 5). First we define these edges and the distance, then we string them together to make the 'E' configuration:

```
(define   e1 (edge vertical 30))
(define   e2 (edge horizontal 20))
(define   e3 (edge horizontal 10))
(define   d1 (distance 10))

(define   E
 (string  horizontal
          e1
          (string vertical e2 d1 e3 d1 e2)))
```

The names of the simplest elements – the edges and

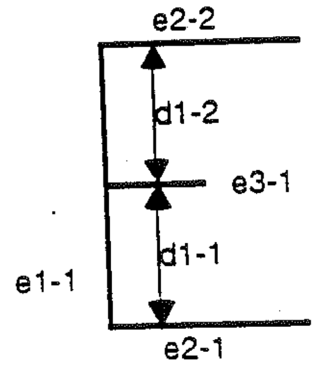

Figure 5. Description of the letter 'E' in WF

distances – are shown in the diagram. Note that each element is named uniquely, even the two instances of edge e2 that form the top and bottom horizontal edges of the E.

Writing Form users can extend the language in two ways. First, WF may be extended by defining *new element types*. Element types defined within WF are first-class WF objects: they may be composed and configured in the same ways as the built-in edge and distance elements. Second, users can define *new operations* for positioning elements relative to one another. These operations are independent of the elements they may be applied to. For example, players may define alignment, centring, equidistant spacing as well as more complex means and routines for positioning elements and configurations using parameters, conditionals and iteration.

We have built two interpreters for the WF language. The first – we call it the WF interpreter – translates WF expressions to layouts for display. The second – the WF reader – translates layouts entered with a graphic editor into WF expressions. For our present purposes it is sufficient for the WF reader to generate any WF expression that describes the layout. (Here we are not concerned with simplest or canonical forms.)

A game recorder

The game recorder appears to the players as a graphic editor. Players record moves by directly manipulating pieces; they need not learn the WF language that the game recorder uses internally to represent board states. The game recorder screen is divided into three parts (see Figure 6). One part contains a supply of elements of various types. Each available element type appears as an icon; selecting the icon generates a new element instance that can be placed on the board. Another part of the screen contains the game board and the pieces in play. Most recently placed pieces are specially marked. A third part of the screen displays the game history.

Players take turns in making moves: selecting elements from the supply and placing them on the game board, moving elements already on the board or removing elements. The game recorder, using the WF reader described above, translates these moves into its internal representation and stores them in its history. Players can view game states by selecting items on the history and also explore the consequences of making alternative moves by playing games over again, beginning at intermediate states.

The game board at any time is described, internally, by a WF expression. As we have seen, the physical configuration and the symbolic WF representation correspond directly. It follows that their transformations also correspond directly. Adding, deleting and moving pieces around on the board corresponds to (lexically) adding, deleting and replacing symbols and subexpressions in the WF expressions. The 'current board state' expression may also be viewed and edited in a text buffer. Changes made textually will alter the layout and this change will be recorded as a move.

Extending the game recorder

So far the game recorder only records players' moves: the

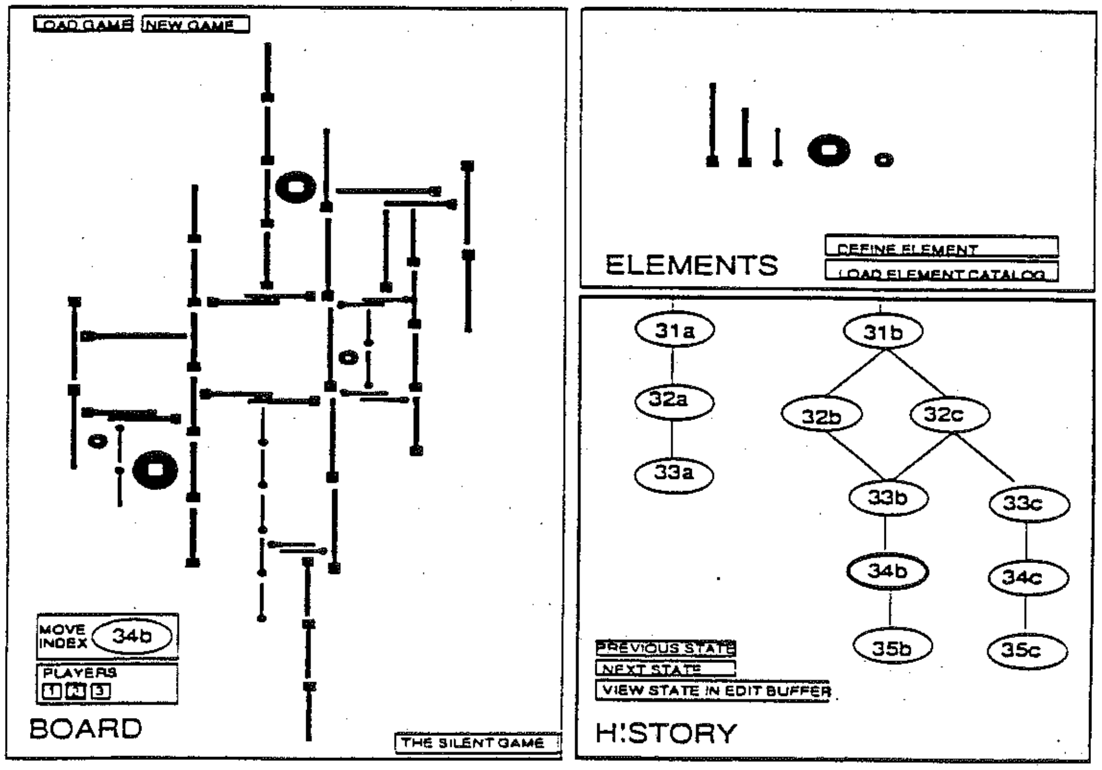

Figure 6. Game recorder screen

sequence of board states and transformations. Now we want to extend the game recorder to take account of rules. In particular we want it to know about constraints on the selection, placement, and dimensions of game pieces. Such constraints determine the 'behaviour' of the game pieces. It must be possible for players as well as the game developer to write these constraints. Here are some examples of constraints we want the game recorder to be able to handle:

'washers only on grid crossings'
'pegs always perpendicular to nails'
'a washer must occur wherever a nail and a peg meet'

In games where a well-defined technical universe dictates allowed and forbidden arrangements of games pieces, we want the game recorder to check players' moves for validity. In games where players introduce rules during play, the game recorder should keep track of when rules are entered, rescinded, applied or violated. Once the game recorder can be taught rules, perhaps it could also suggest possible moves.

To achieve these goals, we are recasting the game recorder and the WF layout language within a constraint-based computing environment. Following well known work on constraints[13-15], we have developed a constraint language[16,17] and connected it with a package of interactive graphic routines. The constraint language interpreter maintains, manages, enforces, and solves relations among variables that describe attributes of a system whose behaviour we wish to simulate. We can use constraints to describe both the geometry of our pieces and the rules about piece placement.

For example, we can define 'peg' as a special class of 'rectangular physical object', with fixed dimensions, and certain position constraints with respect to other similarly defined objects, for instance, nails. We might require of pegs that they must always be connected and perpendicular to a nail. If we place a peg parallel to a nail, the game recorder can (1) report the violation and reject the move; (2) report the violation but accept the move; (3) rotate the peg 90 degrees in order to satisfy the constraint, or (4) offer alternative ways to satisfy the constraint (rotate the nail instead, turn the peg the other way, turn both beg and nail 45 degrees, etc.).

The constraint-based version of the game recorder is still under development. Initial experiments seem promising. In games, as in real-life designing, rules and constraints play an important role. Therefore, constraint-based computing seems to offer an appropriate technology for our research in design theory and methods. However, much work remains to be done, both to advance constraint-based computing techniques and to demonstrate that design rules can be expressed in these terms. We have found our concept design games a useful vehicle for both these aims.

REFERENCES

1 **Habraken, N J** *Supports: An alternative to mass housing* Design and Housing Group, MIT Laboratory for Architecture and Planning, Cambridge, Mass. (1972)

2 **Habraken, N J, Boekholt, J T, Thijssen, A P and Dinjens, P** *Variations – The systematic design of supports* MIT Laboratory for Architecture and Planning/MIT Press, Cambridge, Mass. (1976)

3 **Habraken, N J** *Transformations of the site* Design and Housing Group, MIT Laboratory for Architecture and Planning, Cambridge, Mass. (1983)

4 **Shaw, A** 'Picture graphs, grammars and parsing' In *Frontiers of pattern recognition* Academic Press, New York (1972)

5 **Williams, J D** 'Sticks – A New Approach to LSI Design' *Master's thesis*, Massachusetts Institute of Technology (1977)

6 **Batali, J and Hartheimer, A** 'Design Procedure Language Reference Manual' *MIT Artificial Intelligence Laboratory Memo #598* (1982)

7 **Rosenfeld, A** 'Multidimensional formal languages' In *Basic questions of design theory* (Ed. W Spillers) North-Holland/American Elsevier, New York (1974)

8 **Henderson, P** 'Functional geometry' In *Proc. 1982 ACM Conference on Lisp and Functional Programming* (1982)

9 **Stiny, G** 'Introduction to shape grammars', In *Environment and Planning B* (July 1980) pp 343–351

10 **Flemming, U** 'The role of shape grammars in the analysis and creation of designs' In *Proc. 1986 SUNY Buffalo Symposium on CAD – The Computabiity of Design* John Wiley, New York (forthcoming)

11 **Habraken, N J** 'Writing Form' Unpublished Working Paper, Design Theory and Methods Group, Department of Architecture, MIT (1983)

12 **Coral Software** 'Coral Common Lisp for the Macintosh (beta test version)' Cambridge, Mass. (1986)

13 **Sutherland, L** Sketchpad – A man-machine graphical communication system' *PhD Thesis* MIT

14 **Steele, G L and Sussman, G J** 'CONSTRAINTS – A language for expressing almost hierarchical descriptions' *Artificial Intelligence* Vol 14, pp 1–39 (1980)

15 **Borning, A** 'Thinglab – an object-oriented system for building simulations using constraints' In *Proc. Fifth International Joint Conference on Artificial Intelligence* (1977) pp 497–498

16 **Gross, M** 'Design as exploring constraints' *PhD thesis* MIT 1986

17 **Gross, M, Ervin, S, Anderson, J and Fleisher, A** 'Designing with constraints' In *Proc. 1986 SUNY Buffalo Symposium on CAD – The Computability of Design* John Wiley, New York (forthcoming)

DESIGN STUDIES

Volume 9 Number 3 July 1988

Published in co-operation with the Design Research Society

Guest Editor: Donald A Schön

CONTENTS