Logic and Time-Based Art Practice

Ernest Edmonds

ABSTRACT

This paper is concerned with art practice in the tradition of systematic construction. In particular, the underlying structures of work are seen to be logical, although no particular logic is emphasized. The problems of time-based work are discussed in relation to computer-generated video. It is argued that, in this context, logic programming is important and that images may be generated by such systems in an interesting way. Of particular significance is the fact that the inference system in a logic programming implementation can be seen to define the underlying structure of the time base of a work. An example is described and the implications discussed.

Systematic Construction

The work described in this paper falls within a tradition that focuses on the underlying structure of the artwork, often referred to as the "underlying mathematical structure" [1]. Mathematics is a broad term applying to many different activities, but in this context it is formal systems, rather than all mathematical studies, that are of interest. They are covered within specific domains of mathematics. For the purposes of this paper we will consider the structure to be an underlying *logical* structure. We refer to mathematical logic but not to any particular logical formulation, any of which might be appropriate.

Of particular interest are time-based works, which certainly can be treated within the framework of systematic construction. Formal descriptions of time-based processes can be considered in different ways and using different logics [2]. The author has shown a fragment of a computer-generated video work which explores some of these issues [3]. This paper discusses the formal considerations relating to such art practice and illustrates a new video work.

Logic Programming and Image Handling

As we have suggested, logics play a significant role in the domain of formal systems and, hence, in art practice involving systematic construction. In the context of electronic art, therefore, we are bound to consider the branch of computer science known as logic programming particularly carefully [4]. In effect, logic programming provides an executable, problem-solving interpretation of mathematical logic. We are able to define an underlying structure in a logic programming language and then ask the computer to find solutions to set goals automatically that conform to that structure. The details of this need not detain us here. Rather, we should simply note that underlying logical structures may be specified to computers in this way and that instances, or realizations, may be constructed as a result.

It has been shown that an image on a computer screen can be treated as an object within a logic programming system [5]. It is important to note that as a logical statement concerning the image object is applied, the image may be changed. In this way it is possible to define underlying logical structures for images in a computer and to use those structures to produce specific realizations in the class of images implicitly defined.

Ernest Edmonds, LUTCHI Research Centre, Loughborough University of Technology, Loughborough, Leicestershire, U.K.

Received 15 April 1988.

Pergamon Press plc. Printed in Great Britain.
0024-094X/88 $3.00+0.00

The Inference System and Structures in Time

At the heart of a logic programming language implementation lies what is known as an inference system. This is a piece of software that interprets the logic in order to attempt to achieve whatever goal has been set. Its particular organization determines the way in which the logic is applied to the problem at hand. Put simply, it determines the order in which the logical assertions provided are applied.

An important consequence of this is that the time-based structure of a work may be determined by the definition of the inference system's strategy. By setting forth a particular goal to a computer system that has specified within it both a defined underlying logical structure for the image and the particular inference system to be used, one is systematically constructing a time-based work. The pace of the work will be determined, of course, by the processing method used rather than by the inferencing strategy as such. However, it is quite simple to annotate either the description of the underlying logical structure of the image or the definition of the inference system's strategy, or both, with specifications of the pace at which particular acts should take place.

Jasper: A Sample Work

A computer-generated video work, *Jasper*, can be taken as an example of the ideas described above put into practice [6]. Some illustrations of stills from this video are shown in Fig. 1. The images of the work may be thought of as being constructed on a grid in which locations may be specified in normal rectangular coordinates. They are in shades of grey which can be identified by a number between 0 and 255, where 0 is black and 255 is white.

The structure of the image, in this work, was defined by the following simple rules (expressed here in English):

1. Square X is at position (X,X), has sides of length X and is of tone X *or* tone 255 – X.

2. Picture X is satisfied when square X is drawn *and* X is less than 250 *and* picture Y is satisfied, where Y = X + 10.

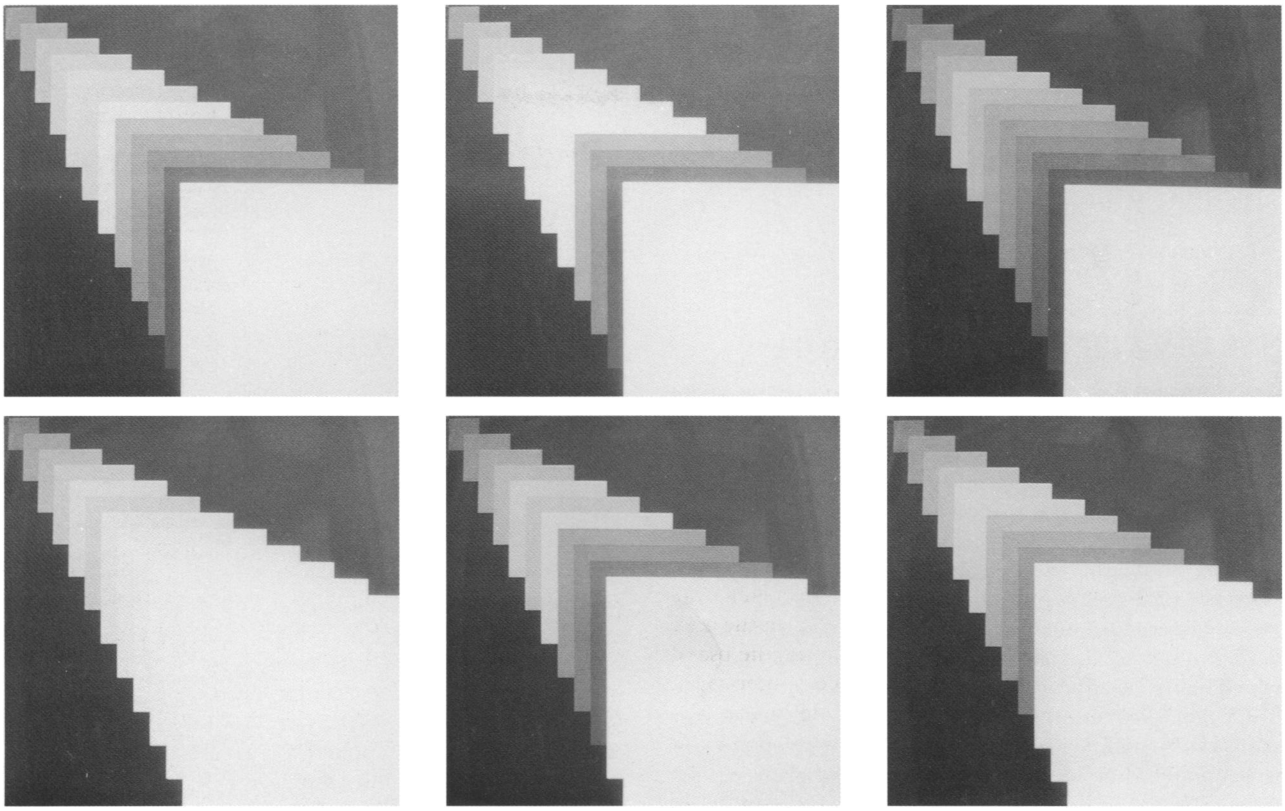

Fig. 1. Stills from *Jasper*, computer-generated video, 1988.

The image is generated by setting for the system the goal of producing picture 10.

Note that the second rule activates itself and hence produces a sequence of actions on the image. Such rules are known as 'recursive' and are very important in logic programming as well as in many other branches of computing and mathematics.

There are, clearly, many ways in which attempts may be made to satisfy these rules. The key point is in the choice of inference system to be used. In the case of *Jasper*, the rules were expressed in the logic programming language PROLOG and its standard inferencing system was applied to them [7]. As part of this strategy the system tries different ways of satisfying its prime goal whenever an attempt fails.

In our example, all attempts fail because satisfying picture 10 will always, in the end, rely upon satisfying picture 250, which is not possible. Thus, the standard inferencing system of PROLOG is used to generate a time-based work that can perhaps be thought of as a relentless attempt to satisfy these very simple, but, in fact, unsatisfiable rules. It might be noted, as an aside, that the negative aspect of this view, the inevitable failure, can be avoided if one sees the moment when X = 250 not as a failure of rule 2, but as the state at which the next alternative solution should be sought.

FURTHER OPPORTUNITIES

No annotation was used in *Jasper* in relation to pace and so the work has a regular driving rhythm. A different, related work, *Jasper Sighs* [8], has also been produced. In this work, rule 1 is annotated in order to specify the time it should take to draw the square, and that time changes in a fixed rhythm of its own.

I am currently exploring different inferencing strategies, ones that involve interaction with a human [9]. This last possibility is clearly addressing the central issue identified for electronic art by Cornock and Edmonds [10] and elaborated by Edmonds [11]: the exploration of the possibilities of art systems with which participants interact.

References

1. S. Bann, "Introduction", *Systems* (U.K.: Arts Council catalogue, 1972) pp. 4–14.
2. E.A. Edmonds, "Discussion 2: Time and Formal Systems", *Exhibiting Space 1985 Autumn Programme* (London: Conspectus, 1985) pp. 142–153.
3. E.A. Edmonds, "Exhibition 4: Duality and Co-Existence", *Exhibiting Space 1985 Autumn Programme* (London: Conspectus, 1985) pp. 23–25.
4. R. Kowalski, *Logic for Problem Solving* (North-Holland, 1979).
5. A. Schappo and E.A. Edmonds, "Support for Tentative Design: Incorporating the Screen Image, as an Object, into PROLOG", *IJMMS* **24** (1986) pp. 601–610.
6. E.A. Edmonds, *Jasper* (computer generated video) (Loughborough, U.K.: LUTCHI Research Centre, 1988).
7. W.F. Clocksin and C.S. Mellish, *Programming in Prolog* (Springer-Verlag, 1981).
8. E.A. Edmonds, *Jasper Sighs* (computer generated video) (Loughborough, U.K.: LUTCHI Research Centre, 1988).
9. E.A. Edmonds, "Negative Knowledge Towards a Strategy for Asking in Logic Programming", *IJMMS* **24** (1986) pp. 597–600.
10. S. Cornock and E.A. Edmonds, "The Creative Process Where the Artist is Amplified or Superseded by the Computer", *Proc. CG70*, Brunel University, U.K. Reprinted in *Leonardo* **6** (1973) pp. 11–15.
11. E. A. Edmonds, *Beyond Computable Numbers* (Loughborough, U. K.: Loughborough University of Technology, 1987).