Def Prog / abst. CAD.

A definitive programming approach to the implementation of CAD software

(extended abstract)

Meurig Beynon
Department of Computer Science,
University of Warwick, Coventry CV4 7AL
Tel: (0203) 523089
E-mail: ...!ukc!warwick!wmb

Keywords: computer-aided design, human-computer interaction, user-interface management, geometric modelling

This paper - a sequel to [2] - outlines an approach to the implementation of CAD systems that makes use of a programming paradigm based upon definitions ("definitive programming"). It includes interpretations for "design" and "intelligent CAD". Since the primary emphasis is upon a methodology that can be applied to developing CAD software, these interpretations are at a much lower level of abstraction than those that can be described from an AI perspective (cf [10]). They are nonetheless consistent with the view of "dialogue over a definitive notation" as an appropriate intermediate code for user-computer interaction.

The concept of using definitions as a basis for interactive software was advocated in [1], and is illustrated in the design of the ARCA and DoNaLD graphics systems [3,4]. The merits of using definitive principles as a basis for software for interactive graphics are described in detail in [2]. In particular, the advantages in respect of data representation and abstraction over traditional procedural or purely declarative programming paradigms are explained.

The framework described in [2] is limited in several respects however. A key idea behind definitive notations is that the current status of a user-computer interaction ("the state of dialogue") is effectively represented by a system of definitions resembling in essence the system of functional relationships between scalar quantities underlying a naive spreadsheet. Such a view of interaction is oriented towards a passive use of the computer in which the responsibility for changing the state of dialogue rests upon the user. In practice, there are many important issues in the design and implementation of CAD systems that demand a more general framework. Valuable though the passive use of sophisticated data description is in the early stages of the design process [12], there is also a very significant role for interaction in which the computer plays an active part. Traditional user-interface management issues come to mind in this context [7,8]: the animation of a dialogue through appropriate use of windows, menus, graphical displays, and the use of analogue rather than textual input. Of equal importance are issues such as the monitoring and maintenance of constraints [11,13]; an activity in which the computer must itself participate in changing the state of dialogue.

The purpose of this paper is to argue that the essential concept of definitive programming (viz the representation of a state of dialogue by means of a set of definitions that is transparent to the user) can be developed to support the broader concerns raised by the implementation of sophisticated CAD systems. Some justification for regarding CAD systems based upon definitive principles as well-suited for the implementation of "intelligent" CAD software is also given. Indeed, the thesis that computer-aided design systems can be very effectively implemented using definitive principles in such a way that the user has a powerful abstract view of the stages of the interactive design process itself suggests a possible interpretation of "intelligent" computer aided "design".

The paper is divided into 5 sections.

In section 1, some of the principal ideas discussed in [2] are briefly reviewed. A brief outline of a definitive notation suitable for geometric modelling - combining some of the characteristics of both the ARCA and DoNaLD notations - is described. The aim is to illustrate explicitly how the ideas in [2] can be applied to the description of geometric objects in many semantically different ways: as abstract complexes of simplices, as frames comprising finite configurations of "points", "lines" and "planes", and as realisations of such frames as sets of points in space (cf [6,14]). In effect, the state of dialogue over such a notation is regarded as a formal representation of the user's current model of a geometric object, and the design process is viewed as a process of refinement of this model associated with a sequence of transitions through different dialogue states.

Section 2 focusses on some limitations of this "pure definitive notation paradigm" for design as a basis for developing CAD system software. Two principal issues are considered

1) the need in general for a framework of constraints within which the design dialogue must operate,
2) the nature of the relationship between the semantics of the geometric design dialogue and the mechanics of the interaction between the user and the computer.

(These are respectively considered in detail in Sections 3 and 4.)

Where 2) is concerned, it may be seen that no provision is made within the pure definitive notation design paradigm for representation of the current state of the device supporting the user-computer interaction. This is in some sense appropriate, in that the abstract design process has meaning independent of the ephemeral state of the screen during some specific design transaction; on the other hand, there is in general a complex interrelationship between the underlying semantics and the current state of the display interface. It is proposed to deal with this problem by introducing an auxiliary definitive notation within which to specify the screen display at any stage. In effect, the form of the user interface can then be specified as a set of definitions, either by the system or by the user, and subsequently driven by dialogue actions initiated by the computer.

An underlying principle of definitive programming, relevant to both 1) and 2), is that the way in which a transition between dialogue states is effected is not of primary significance; what is crucially important is that the current state of dialogue can be subtly represented, and is at all times transparent to the user. To exploit this principle fully, a richer programming paradigm is required, in which - in the conceptually simplest framework - several processes may participate in the design dialogue (see [5], and compare [9]). To this end, an extended form of definitive programming is described, incorporating definitions and user-defined functions (as in a pure definitive notation) together with actions that - in an appropriate dialogue state - are invoked to change this state. Sections 3 and 4 are concerned with the way in which such a computational framework may be used to handle constraints and aspects of the user-interface management in a unified manner, and include some illustrative examples based on a rudimentary prototype implementation.

The final section of the paper discusses the potential advantages of a definitive programming approach to the development of an "intelligent" CAD system. As a useful focal point for future research and experiment in this area, it is suggested that the functional relationships that are explicit in a definition based model can profitably be regarded as constituting intelligent knowledge, and the inference of such functional relationships as a characteristic feature of intelligent CAD systems. As a footnote, it might be added that the interesting inferences would appear to involve more than simply logical extrapolation from functional relationships specified by the user, and presumably entail some monitoring of the user's models to support inductive inference.

Acknowledgements

The author is indebted to Alan Cartwright and Edward Yung for helpful ideas and programming support.

References

1: W M Beynon, Definitive notations for interaction, Proc hci'85, CUP 85
2: W M Beynon, Definitive principles for interactive graphics, Proc NATO ASI: Theoretical Foundations of Computer Graphics and CAD, Il Ciocco, July 1987
3: W M Beynon, ARCA: a notation for displaying an manipulating combinatorial diagrams, University of Warwick RR#78, 1986
4: W M Beynon, D Angier, T Bissell, S Hunt, DoNaLD: a line drawing system based on definitive principles, University of Warwick RR#86, 1986
5: W M Beynon, The LSD notation for communicating systems, University of Warwick RR#87, 1986
6: U Cugini, The role of different levels of modelling in CAD systems, Proc NATO ASI: Theoretical

Foundations of Computer Graphics and CAD, Il Ciocco, July 1987
7: P ten Hagen, R van Liere, A model for graphical interaction, Proc NATO ASI: Theoretical Foundations of Computer Graphics and CAD, Il Ciocco, July 1987
8: J Foley, C Gibbs, W C Kim, S Kovacevic, Formal specification and transformation of user computer interfaces, Report GWU-IIST-87-10, Dept of Electrical Engineering and Computer Science, George Washington University, 1987
9: M L Kersten, F H Schippers, Towards an object-centered database language, Report CS-R8630, CWI Amsterdam 1986
10: J Lansdown, Graphics, Design and Artificial Intelligence, Proc NATO ASI: Theoretical Foundations of Computer Graphics and CAD, Il Ciocco, July 1987
11: G Nelson, Juno, a constraint-based graphics system, SIGGRAPH '85, p235-243
12: R F Riesenfeld et al, Computer aided design, UUCS-84-003, Computer Science Dept, Univ of Utah 1984
13: T Takala, C D Woodward, Industrial design based on geometric intentions, Proc NATO ASI: Theoretical Foundations of Computer Graphics and CAD, Il Ciocco, July 1987
14: K Weiler, Edge-based data structures for solid modelling in curved surface environments, IEEE Computer Graphics and Applications, 1986, p21-40