Eliminating the Dichotomy Between Scripting and Interaction

John F. Schlag

Three Dimensional Animation Systems Group
Computer Graphics Laboratory
New York Institute of Technology

Abstract

Throughout the computer graphics literature, but in modeling and animation lore in particular, the prevalent attitude seems to be that scripting and interaction are irreconcilably different types of user interface. This paper propounds the belief that this dichotomy is a myth and gives examples of systems which encourage this belief. Two of the systems described are a keyframe animation program and a geometric modeling program developed at the NYIT Computer Graphics Laboratory. Both of these systems are used on a daily basis for production, development and research.

1. Introduction

Scripting and interaction are widely used interface styles for modeling and animation programs. The advantages and disadvantages of each are well known. Scripting systems allow command sequences to be incrementally modified and reexecuted, but allow no interactive input of data and operations. Interactive systems allow this input, but lose all record of commands previously executed, leaving the user to start over if some intermediate parameter affecting the final result needs to be changed. In almost every publication describing a new modeling or animation system, there is a paragraph or two which notes these properties and proceeds to ballyhoo the advantages of the method chosen. As with many other such apparent dichotomies, most designers accept without question the division, pick one technique, implement it and resign themselves to writing the obligatory text. From this lamentable situation we may conclude that the production of a user interface which integrates scripting and interaction is *difficult*, but certainly not that it is impossible.

The integration of scripting and interaction is worthwhile for several reasons. The first is that a more general conceptualization of user interfaces results. This has the potential for expanding the set of problems that can be solved with a computer. On a more practical note, interactive graphics workstations are often expensive and therefore scarce. In many graphics houses, the time available on such resources is a limiting factor in the amount and/or complexity of animation producible. It helps greatly if users can work, albeit more slowly, at ordinary terminals, using scripting instead of interaction.

Resource limits are not the only reason to use scripting at one workstation and interaction at another. The production of computer animation requires many diverse activities, most of which are accomplished more efficiently with one technique than the other. Some establishments may boast interactive *and* scripting tools for particular activities, but rarely do these tools interface to a common database format. More often, the systems are incompatible "competitors", perhaps having been written by different individuals, and any attempt to use both to solve a single problem is frustrated by the need to reorganize data, shuffle files about and translate between formats. A system which integrates scripting and interaction can be used for conceptually diverse activities in whatever mode is appropriate. A banner example of the type of production which needs this is the now famous robot ant animation by Lundin [15], in which the basic 3d path of the model is supplied interactively and the dynamics are supplied by an explicit computational model.

For perspective, the more general problem here is the development of *editing theory*. A general theory of editing should be independent of the data to be edited, so it should be applicable not only to ordinary text editing, but to other activities such as geometric modeling (shape editing), animation and robot programming (motion editing), music production (sound editing), painting (image editing) and computer programming (algorithm editing). What we would like for any given data format is a logical and complete set of functions for manipulating that format, and both interactive and scripting interfaces to these functions.

When differentiating between interaction and scripting, it is natural at some point to quibble about exactly what *interactive* means. Some would argue that if the time around the typical edit-process-view loop of a scripting system is short enough, the system is interactive. Is there some threshold on this loop time which must be met? "Real" time? Historically, the term has been used to mark a contrast with *batch* systems, where the sizes of input and output data sets are usually large. Systems that produce visible results after small amounts of input