EXPLOITING CLASSES IN MODELING AND DISPLAY SOFTWARE

Turner Whitted and Eric Grant

Department of Computer Science
The University of North Carolina
Chapel Hill, North Carolina

Abstract

The class concept is one component of object-oriented programming systems which has proven useful in organizing complex software. In experimenting with the use of classes for geometric modeling applications, we have devised a class hierarchy that yields some conceptual order in the midst of diverse representations of shapes. Rather than searching for a uniform primitive representation, we accept the diversity and build a framework in which dissimilar models are combined in an orderly manner.

KEYWORDS: geometric modeling, procedure models, object-oriented programming

Introduction

Geometric modeling systems can become extremely complex when design applications demand flexibility in the representation of shapes. A major challenge for programmers who create such systems is to preserve order in spite of this complexity. Trends in this direction include moves toward uniform data representation and very general mathematical representations of shapes. However, the advent of specialized procedural modeling techniques is a step away from uniformity which strains programmers' abilities to cope with the diversity that it presents. We feel that using an object-oriented programming methodology helps to solve this problem.

In spite of the recent interest in object-oriented programming, we have seen only a few published examples of 3-D graphics systems built in an object-oriented environment [Hedelman, Lorenson]. For the most part, the examples that we have seen emphasize the message passing aspects of Smalltalk-like languages and do not pay much attention to the effective specification of classes.

In this paper we describe the class hierarchy of an experimental modeling and display system which we have assembled in order to study the problems of constructing extremely complex geometric objects. We emphasize identifying common elements of geometric procedures which can be shared among representations. Our guiding principle is that methods should be shared by as many classes as possible and that they should belong to classes as high up in the class hierarchy as possible. In a following section we describe our attempt to define a class hierarchy that meets this criterion.

Diversity of Geometric Representations

Geometric models used in computer graphics have become so diverse that they don't seem to fit any rational scheme of classification. There has been a substantial amount of development of modelers that manipulate polygonal meshes, models that produce parametric surfaces or algebraic surfaces, and unified modeling systems that handle all three representations. Some of these have gone so far as to devise a common representation to ease the difficulty of storing and manipulating the diverse representations.

The widespread use of procedure models [Newell] has complicated the issue even further since the procedures are often restricted to such narrow purposes as creating trees [Bloomenthal], terrain [Fournier], or grass [Reeves]. To be sure, there are general purpose procedures such as sweeping, and there are generalizations such as graftals [Smith] which provide a common framework for a variety of individual geometric procedures.

The common simplification of reducing all complex shapes to polygonal approximations is no longer feasible when the number of primitive elements exceeds a few tens of thousands. Modification of such complex collections by users is nearly impossible. Common operations such as interference checking and display are confronted with massive amounts of data in this case.

Classes

One of the more successful mechanisms for coping with the complexity of programming is the use of classes. Originally a feature of the Simula language, classes are a central feature of Smalltalk [Goldberg]. Their value is more apparent when one considers that mature languages such as Lisp [Cannon] and C [Stroustrup] have been extended to include classes.