6. The CAD*I processors for Matra Datavision's Euclid

Principal author: U. Gengenbach

6.1 System description

Euclid is a solid modeller developed and sold by Matra Datavision, France. Its data structure captures both CSG and Polyhedron information. A number of application packages are based on the kernel modeller such as

- AEC
- 2D drafting
- Schematics
- Sculptured surfaces
- Sheet metal bending
- Finite element calculations
- NC programme generation

All of these application modules are attached to the kernel system via the so-called language interface, a library of subroutine calls for creating, manipulating, deleting and evaluating objects in the Euclid data structure. Euclid version 4.2 has been made available during the project by Matra Datavision. It was running on a DEC MicroVAX II minicomputer under operating system VMS at KfK.

6.2 Existing interfaces

Among the possible interfacing techniques to the system as outlined in "Access to CAD system data bases" on page 33 the following were available for Euclid:

Processors for IGES and VDAFS
Euclid provides processors for both neutral file formats. Their source code, however, was not available during the project.

Log file
Euclid writes a log file of each session. It is used to recall a session after the system has been aborted by the user or some external reason. Since it is written in binary format and its structure is not documented it is not suitable for interfacing purposes.

Euclid language interface
The Euclid language interface is the main interface to the modeller [57], [58]. Its primary purpose is to make application programmes, both those supplied by Matra Datavision and those written by the user, independent of the particular data structure. It goes along with the Euclid kernel system as a set of over 200 Fortran 4 subroutine calls. Programmes based on these subroutine calls may be linked into the system as application programmes or run interactively by means of an interpreter. The subroutine calls of the language interface may be roughly classified into the following categories.

- creation of basic geometric elements (points, curves, surfaces, solids)
- geometric transformations
- modelling operations (intersections between curves or surfaces, Boolean operations)
- visualisation
- creation of entities for the definition of drawings (e.g dimensions).
- setting of system parameters (e.g. initialisations, diagnostic messages)

E. G. Schlechtendahl (ed.), *CAD Data Transfer for Solid Models*
© ECSC — EEC — EAEC, Brussels — Luxembourg 1989

- administration of the data structure (e.g. pointer handling)
- dialogue handling
- decoding of the data structure.

In addition to that language interface Euclid provides a utility to dump the content of the data structure, large commons, in table format onto a file. Most of this information is in hexadecimal code. Its semantics is documented only for versions up to Euclid version 4.1. So this is no reliable basis for processor development. It is the policy of Matra Datavision to prevent the user from mingling with the details of the data structure. The language interface provides sufficient functional capability for virtually every application, keeping investments in software independent of changes in the data structure.

The consequence of these considerations was that both pre- and post-processors are based on the Euclid language interface.

6.3 Internal representation of CAD models

The following chapter presents a formal description of that part of the Euclid data structure that was relevant for CAD*I processor development. It does not include free-form geometry and nongeometric entity types like annotations for drafting applications.

The structure of the interactive Euclid database is a stack. The objects on the stack are sorted according to the LIFO principle. The active object is on top of the stack. Objects can be made active by

- graphical identification on the display,
- by searching for their user-defined name, or
- by browsing through the stack.

Objects on the stack may be of type point, curve, surface, face, figure or solid.

The Euclid scope
The Euclid data structure supports the concept of scope in the following sense:

The entity types

- E_FIGURE,
- E_REVOL,
- E_PRISM and
- E_CONSTRUCT

have a hierarchical data structure. That means a constituent, unless it is of type E_INSTANCE, can only be accessed by starting at the root node and traversing the structure. Hence those entities may be regarded as being in the scope of the enclosing entity. The analogy is most striking with the E_FIGURE entity in EUCLID, where the user has to issue the command:

<OPEN FIGURE>

in order to create, evaluate, modify or delete its constituents. Applying transformations on an E_FIGURE implies that its constituents follow that transformation.

The other three entity types show the same behaviour though it is not that transparently supported with operations on the user interface.