Chapter 4.
Identifying Model Objects

Topic: *Entity, *Model Object

A *model object* is any object that can be saved to and restored from a save file (.sat or .sab). ACIS model objects are implemented in C++ using a hierarchy of classes derived from the ENTITY class. Model objects are described in the *3D ACIS Fundamental Concepts Guide*.

This chapter describes mechanisms for identifying the type of any class derived from ENTITY, as well as its level of derivation.

Using is_entityclass Functions

Topic: *Entity, *Model Object

Applications may use functions named is_<*entityclass*> (where <*entityclass*> is the name of a class derived from ENTITY) to determine if a given object is a specific type of entity (i.e., a specific class derived from ENTITY). For example the is_ELLIPSE function takes an ENTITY object and returns a logical that indicates if the ENTITY is an ELLIPSE or not.

Derivation from ENTITY

Topic: *Entity, *Model Object

Although the ENTITY class itself does not represent objects within the modeler, any class that represents a permanent model object in ACIS must be derived from ENTITY in order to inherit the common data and functionality that is mandatory in all permanent objects.

An ENTITY pointer can point to an object of class ENTITY, or to an object of any class derived from ENTITY (directly or indirectly). Because it is often necessary to know the specific class of the object (FACE, LUMP, SHELL, SURFACE, STRAIGHT, etc.), functionality is built into these classes to support this determination.

Figure 4-1 illustrates the derivation of the classes from ENTITY. The individual classes are described in detail in reference templates in online help.

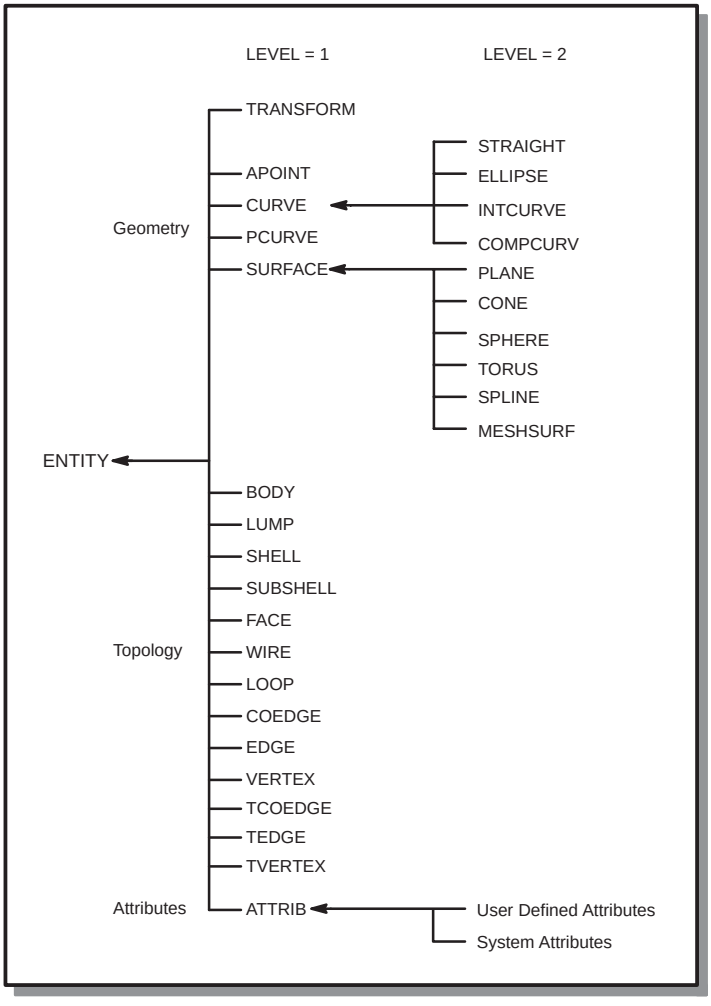

Figure 4-1. Derivation From ENTITY

Object Type

Topic: *Entity, *Model Object

Each ACIS class derived from ENTITY has a *type*. Each type is assigned a unique constant integer value associated with the class at compile time. No two type variables have the same value.

The actual integer value of a type should never be directly referenced in code, since it may change. Instead, the type's name should be referenced. Types are named <CLASS>_TYPE.

The following table lists the most common classes and type names:

Class	Type Name	Class	Type Name
BODY	BODY_TYPE	POINT	POINT_TYPE
LUMP	LUMP_TYPE	CURVE	CURVE_TYPE
WIRE	WIRE_TYPE	PCURVE	PCURVE_TYPE
SHELL	SHELL_TYPE	SURFACE	SRUFACE_TYPE
SUBSHELL	SUBSHELL_TYPE	ELLIPSE	ELLIPSE_TYPE
FACE	FACE_TYPE	STRAIGHT	STRAIGHT_TYPE
LOOP	LOOP_TYPE	SPHERE	SPHERE_TYPE
COEDGE	COEDGE_TYPE	CONE	CONE_TYPE
EDGE	EDGE_TYPE	PLANE	PLANE_TYPE
VERTEX	VERTEX_TYPE	TORUS	TORUS_TYPE
		SPLINE	SPLINE_TYPE
TRANSFORM	TRANSFORM_TYPE	INTCURVE	INTCURVE_TYPE
		MESHSURF	MESHSURF_TYPE
ATTRIB	ATTRIB_TYPE	COMPCURV	COMPCURVE_TYPE

Object Level

Topic: *Entity, *Model Object

Each ACIS class derived from ENTITY has a *level*, which is an integer representing the level of derivation of the class from ENTITY. For example, BODY is directly derived from ENTITY and has a level of 1. SPHERE is derived from SURFACE, which is derived from ENTITY, so SPHERE has a level of 2.

The actual integer value of the level should never directly referenced, since it could change (although this is unlikely). Instead, the level's name should be referenced. Levels for each class are named <CLASS>_LEVEL.

The following table lists the most common classes and their levels of derivation from the ENTITY class:

Class	Level Name and Value	Class	Level Name and Value
BODY	BODY_LEVEL = 1	POINT	POINT_LEVEL = 1
LUMP	LUMP_LEVEL = 1	CURVE	CURVE_LEVEL = 1

Class	Level Name and Value	Class	Level Name and Value
WIRE	WIRE_LEVEL = 1	PCURVE	PCURVE_LEVEL = 1
SHELL	SHELL_LEVEL = 1	SURFACE	SRUFACE_LEVEL = 1
SUBSHELL	SUBSHELL_LEVEL = 1	ELLIPSE	ELLIPSE_LEVEL = 2
FACE	FACE_LEVEL = 1	STRAIGHT	STRAIGHT_LEVEL = 2
LOOP	LOOP_LEVEL = 1	SPHERE	SPHERE_LEVEL = 2
COEDGE	COEDGE_LEVEL = 1	CONE	CONE_LEVEL = 2
EDGE	EDGE_LEVEL = 1	PLANE	PLANE_LEVEL = 2
VERTEX	VERTEX_LEVEL = 1	TORUS	TORUS_LEVEL = 2
		SPLINE	SPLINE_LEVEL = 2
TRANSFORM	TRANSFORM_LEVEL = 1	INTCURVE	INTCURVE_LEVEL = 2
		MESHSURF	MESHSURF_LEVEL = 2
ATTRIB	ATTRIB_LEVEL = 1	COMPCURV	CO2PCURVE_LEVEL = 1

Inquiring the Type of an Object

Topic: *Entity, *Model Object

Each ACIS class derived from ENTITY has an identity method to allow applications and ACIS to inquire the type of of an object, given an ENTITY pointer to that object.

The identity method accepts a single integer argument for the derivation level desired and returns an integer specifying the type of the object. The prototype for the identity method is:

```
int identity(int level) const;
```

The level argument is optional. If no level is specified, the most specific type of the object (greatest level of derivation) is returned.

For example, if an object is defined of a class whose derivation is sphere:SPHERE:SURFACE:ENTITY, and no level is specified, the identity function returns an integer whose value is sphere_TYPE (which corresponds to level=3). If level=2 is specified, SPHERE_TYPE is returned. If level=1 is specified, SURFACE_TYPE is returned. This mechanism, along with the C++ concept of polymorphism, allows a parent class to automatically call child methods where appropriate, and allows application code to determine the exact nature of the object being operated on. Refer to Figure 4-2.

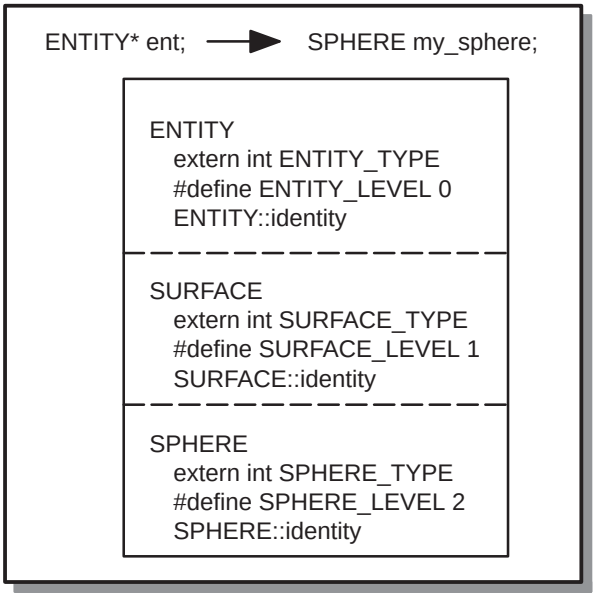

Figure 4-2. A SPHERE Object