Chapter 5. Geometry

Topic: *Construction Geometry, *Model Geometry

Geometry refers to the physical items represented by the model (such as points, curves, and surfaces), independent of their spatial—or topological—relationships. The ACIS free–form geometry routines are based on non–uniform rational B–splines (NURBS).

In addition to manifold geometry, ACIS can represent nonmanifold geometry. Geometry can be bounded, unbounded, or semi-bounded, allowing for complete and incomplete bodies. For example, a solid can have faces missing and existing faces can have missing edges. Solids can also have internal faces that divide the solid into cells.

The ACIS philosophy related to geometric definitions is that the representation be essentially coordinate system independent, numerically stable, and easily transformed. To this end every definition uses classes to represent positions in space, displacements between positions, and directions. In addition, some require scalar values to represent distances or dimensionless quantities. These are simply floating point numbers, and their specific meaning is determined within the transformation algorithm.

Chapter 2, *How ACIS Uses C++*, discusses the underlying math foundation classes needed to define geometry, which help define a model.

Types of Curves and Surfaces

Topic: *Construction Geometry, *Model Geometry

This section defines some types of curves and surfaces that are discussed in this chapter. For more information on curves and surfaces, refer to Chapter 7, *Curves and Surfaces*.

Analytic refers to a simple curve or surface that can be wholly represented by a simple algebraic formula. Analytic curves include straight lines and ellipses (including circles, which are simply special forms of ellipses). Analytic surfaces include cones (including cylinders, which are special forms of cones), planes, spheres, and tori. Some analytic surfaces, such as ellipsoids, are implemented in ACIS as splines.

Composite refers to a curve that is an ordered list of analytic and/or interpolated curves.

Interpolated refers to a curve that is the general representation of any curve that is not defined in ACIS by one of the analytic curves (i.e., by an explicit "equation"), but by reference to other geometric entities. Examples include the intersection of two surfaces, a spline curve (2D or 3D B-spline), and a silhouette curve. An interpolated curve is also called an *intcurve*.

Mesh refers to a surface composed of an array of tessellations, or facets, suitable for lightweight processing of very large amounts of data, such as are used by geophysical modeling applications.

Parameter space curve (*pcurve*) refers to a mapping of a 2D curve onto a 3D surface.

Spline refers to a curve or surface that cannot be directly represented in ACIS by one of the simple analytic surfaces (i.e., by a simple analytic formula), but that can be indirectly represented by an ordered list of analytic formulae, by segmenting the curve or surface with control points. ACIS implements some analytic surfaces, such as ellipsoids, as splines.

Construction and Model Geometry

Topic: *Construction Geometry, *Model Geometry

ACIS implements geometry in two distinct forms. *Construction geometry* refers to the C++ classes that hold the mathematical definitions of geometric objects. *Model geometry* refers to the C++ classes that add model management functionality to the construction geometry.

Construction Geometry

The construction geometry classes are lightweight, and, by themselves, are temporary in nature and are not saved as part of the user's model. They can be used on-the-fly for math calculations, or can be included as part of model geometry to form permanent model objects.

Construction geometry classes have lowercase names. These classes include compcurv, cone, ellipse, intcurve, meshsurf, plane, sphere, spline, straight, and torus. The math class SPAtransf, which represents a 3D affine transformation, is often grouped with the construction geometry classes for discussion.

The construction geometry classes do not have any private data members; everything is public and nothing is behind private methods. The reason for this is that without this management overhead, the data can be passed directly into intersectors to make the performance faster. All data and evaluators of the geometry classes can be freely accessed.

For example, the sphere class (lowercase) is a construction geometry class that mathematically defines a sphere in both *xyz* object space and in *uv* parameter space. It contains methods that construct, destroy, modify, inquire, and evaluate the sphere. It does *not* contain any model management methods, and cannot be directly saved to an ACIS save file.

Model Geometry

Model geometry is permanent and saved with the model. Model geometry classes provide model management functionality on top of the geometry definition (they include construction geometry classes as part of their data structures). This includes saving and restoring of model data, history and roll, transforms, and the ability to attach system-defined or user-defined attributes for carrying abstract data along with the model objects.

Model geometry classes are derived from the ENTITY class, and have uppercase names. These classes include COMPCURV, CONE, ELLIPSE, INTCURVE, MESHSURF, PLANE, SPHERE, SPLINE, STRAIGHT, and TORUS. The math class TRANSFORM, which represents a transformation, is often grouped with the model geometry classes for discussion; it is saved as part of the model.

For example, the SPHERE class (uppercase) is a model geometry class that defines a sphere as an object in the model. It includes as one of its data elements a sphere. It provides methods for identification, save and restore, history and roll, multiple user use counts, and other common model management operations. It also provides methods to set the sphere's center and radius, and to transform the sphere's location. This can be saved to an ACIS save file, and the saved data includes the construction geometry information from the sphere data element.

Abstract and Specific Geometry

Topic: *Construction Geometry, *Model Geometry

Both construction geometry and model geometry can be conceptually separated into *abstract geometry* and *specific geometry.* Abstract geometry is usually broken down into specific geometry. The specific geometry classes are derived from the abstract ones, and thus inherit properties common to all geometry of that type. The specific geometry classes further refine the geometry definition. For example, the abstract geometry for a curve is broken down into specific geometry for specific types of curves, such as ellipse, straight, interpolated curve, etc.

Figure 5-1 uses model geometry classes to illustrates how abstract and specific geometry relate. It shows abstract model geometry on the left and the specific model geometry into which the abstract model geometry is broken down (if any) on the right.

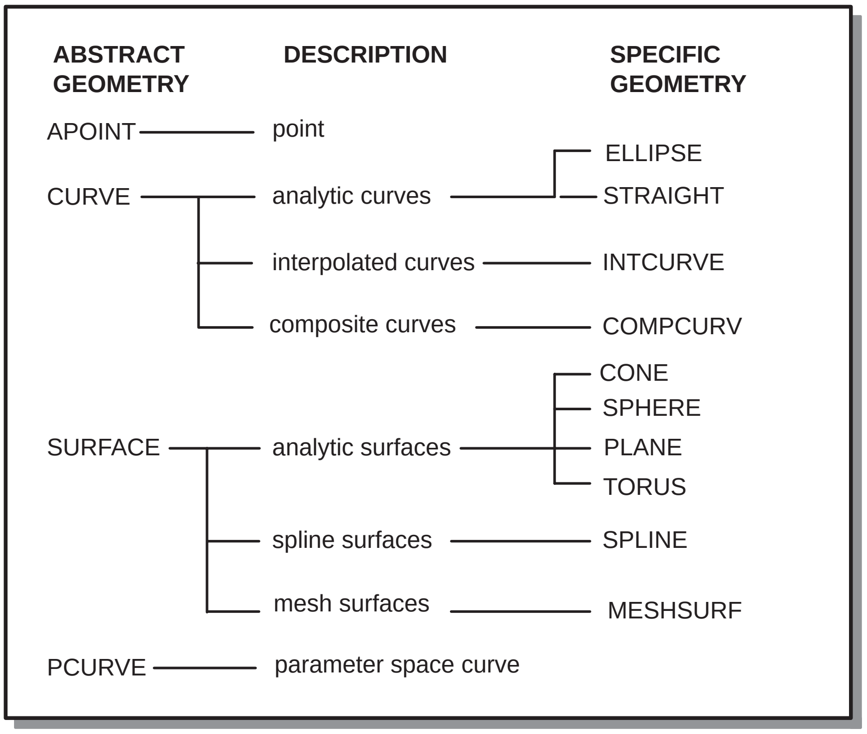

Figure 5-1. Abstract and Specific Geometry

The abstract model geometry classes CURVE and SURFACE define model management related data and methods common to their specific model geometry children classes CONE, ELLIPSE, INTCURVE, MESHSURF, PLANE, SPHERE, SPLINE, STRAIGHT, and TORUS. The abstract model geometry classes APOINT and PCURVE do not have specific geometry class children; their geometry definitions are not further refined.

This can be easily seen by looking at the reference template for a given class in online help. The *Derivation* field shows the derivation of each class, showing children on the left and ancestors on the right. Children inherit and can use the data and methods of their parents.

Note *The model geometry class describing a point is named* APOINT *to avoid compile and link conflicts with the Microsoft class* POINT.

Geometry Data

Topic: *Construction Geometry, *Model Geometry

Each type of geometry has its own unique set of data. For example:

Curve	A curve record holds a count of the number of edges that refer to the curve (a *use count*), and provides a route to the details of individual curve types (such as straight lines, ellipses, etc.).
Pcurve	A pcurve defines a 2D parametric space curve. It maps a real line interval into the domain of a parametric surface, so that evaluating the surface of the image of the mapping gives a 3D object space curve lying on the surface. A pcurve record specifies the parameterization on a surface of a coedge (edge) that represents a curve.
Point	A point holds a count of the number of vertices that refer to the point (a *use count*), and records the coordinates of the point.
Surface	A surface record holds a count of the number of faces that refer to the surface (a *use count*), and provides a route to the details of individual surface types (such as planes, cones, spheres, tori, splines, etc.).
Transform	A transform record holds object space transformations for an entire body. A transform is composed of an affine part, a translation part, a scaling part, a rotate flag, a reflect flag, and a shear flag. Transforms can be combined, inverted, compared, reflected, or rotated.

Refer to the reference templates for the classes used to implement these geometry entities for details about the type of data used by each.