Chapter 1.

Interactive Hidden Line Component

Component: *Interactive Hidden Line

The Interactive Hidden Line Component (IHL), in the ihl directory, creates views of ACIS model objects with hidden lines removed. These views can be used for such tasks as visualization and drafting.

IHL calls the ACIS faceter to facet the model, then uses the 3D mesh of facets to calculate a list of visible and hidden line segments. These line segments can then be displayed. Multiple IHL views can be calculated following a single faceting operation.

A *model edge* is a real edge of the model represented by the EDGE class. A *silhouette edge* is a visualization cue that does not necessarily correspond to a model edge, such as the circular silhouette of a sphere.

Hidden line data can be easily attached to the model using attributes, allowing the data to be saved and restored with the rest of the model.

Hidden line data can also be output directly to the application, using output manager classes, for display or other types of processing.

Support and Limitations

Topic: *Interactive Hidden Line

IHL supports the following model conditions:

- Sheet and solid bodies (may be mixed)
- Manifold and nonmanifold bodies
- Intersecting bodies
- Accepts global facet meshes corresponding to bodies or sets of bodies
- Accepts local facet meshes corresponding to portions of bodies

IHL has the following limitations:

- Wire bodies are not supported
- No smoothing is done

Visibility and Views

Topic: *Interactive Hidden Line

A *hidden line* is a line segment corresponding to part of either a model edge or a silhouette edge that is obscured from the camera (viewpoint) by part of the model. Lines on faces that point away from the camera are hidden, for example. A *visible* line is a line segment that can be seen directly from the camera with no intervening material.

Hidden line data is *view dependent*. Line segments that are hidden in one view may be visible in a different view.

The view to which the data corresponds is identified by a *view token* that is stored with hidden line data. Multiple sets of hidden line data corresponding to various views can thus be calculated, stored on the model, and passed to the application for display. API functions and Scheme extensions that manipulate hidden line data accept a view token as input.

Hidden line data may be calculated for *perspective* views, which have a visual vanishing point, (for example parallel railroad tracks that visually merge at a single point in the distance), or *orthographic* views, which have no such vanishing point. API functions and Scheme extensions that calculate hidden line data accept a perspective flag as input.

Hidden Line Data Generation

Topic: *Interactive Hidden Line

The input to IHL is a list of bodies. Some or all of the bodies in a model can be supplied, depending on what the application needs.

To generate hidden line data the application calls the api_ihl_compute function (or ihl:compute Scheme extension), passing it this list of bodies. IHL sends the bodies to the faceter and gets a corresponding mesh of facets in return. It then uses this mesh to calculate hidden line segment data.

If the application has a pre-calculated mesh of facets, such as a mesh which has been saved on the model at some earlier time, the api_ihl_compute_from_meshes function can be called to generate hidden line data directly from this mesh without calling the faceter again.

Hidden line data always includes visible line segments. It can optionally also include hidden line segments. Visibility information is attached to each segment.

Hidden line segments are output polygon by polygon so they are partially sorted in a natural way. The actual order depends strongly on the faceter.

IHL depends on the planarity of the facets for proper operation. However, the faceter often creates nonplanar facets. To avoid errors try to set the faceting refinement so that the facets are as regular and planar as possible. Refer to the *Faceter Component Manual* for more information.

Storing Hidden Line Data on the Model

Topic: *Interactive Hidden Line

The api_ihl_compute function (or the ihl:compute Scheme extension) calculates and optionally attaches hidden line data to the model bodies. If the function is called with a nonzero view token, data is attached to the model. If it is called with a view token of zero, no data is attached.

When hidden line data is no longer needed, it can be removed from the model by calling the api_ihl_clean function (or the ihl:clean Scheme extension) with the appropriate view token. Only data with a matching view token is removed, permitting the application to remove one view's hidden line data while retaining other views' data.

Note *It is important to call* api_ihl_clean *to free memory when the hidden line data is no longer required.*

Hidden line data is attached to the model using attributes. When the model is then saved or restored, the hidden line data accompanies it. The ATTRIB_IHL_VW class attaches interactive hidden line data and viewing parameters to the bodies from which the data was computed. The data is stored as a doubly linked list of IHL_EDGEs that can be queried, and a camera pointer.

Hidden Line Data Output

Topic: *Interactive Hidden Line

Hidden line data is output using *output manager* classes:

- IHL_OUTPUT_MANAGER provides a generic framework for handling hidden line data output. Other output managers are derived from it for specific tasks, such as attaching hidden line data to the model, displaying hidden line data, etc.
- IHL_STDOUT_MANAGER attaches hidden line data to the model in attributes when api_ihl_compute is called.
- *Application-specific* output mangers can be derived from IHL_OUTPUT_MANAGER to handle specific output tasks when api_ihl_compute_from_meshes is called.

IHL_OUTPUT_MANAGER

Topic: Interactive Hidden Line

The IHL_OUTPUT_MANAGER base class provides the default set of methods to specify what kind of hidden line output should be generated and what should be done when segments are generated. This class is not intended to be used directly. Child classes should be derived from this class to handle specific output tasks.

The IHL_OUTPUT_MANAGER class provides a set of *announce* methods that are called when a 2D segment has been generated, when a 3D segment has been generated, and when the next body in the list of input bodies is starting to be processed. These are generic methods which do nothing. When a specific output manager class is derived from IHL_OUTPUT_MANAGER, the announce methods are rewritten to accomplish specific output tasks.

The class also provides three methods to specify what kind of output is generated:

- 2D or 3D coordinate output can be chosen.
- Interior segments corresponding to edges within the inside of the model can be calculated or thrown away.
- Hidden segments can be calculated or thrown away.

IHL_STDOUT_MANAGER

Topic: Interactive Hidden Line

The IHL_STDOUT_MANAGER class is provided to handle output when no application-specific output manager class has been specified. It is the output manager used when api_ihl_compute is called.

The announce methods of this class store the hidden line data on the model in attributes. The attributes remain on the model until api_ihl_clean is called to remove them.

Like its parent class, IHL_STDOUT_MANAGER has three hidden line generation controls available:

- 2D or 3D coordinate output can be chosen.
- Interior segments corresponding to edges within the inside of the model can be calculated or thrown away.
- Hidden segments can be calculated or thrown away.

Application Defined Output Managers

Topic: Interactive Hidden Line

If the application needs to handle hidden line output in some specific manner, it can derive classes from IHL_OUTPUT_MANAGER and tailor the announce methods to the specific needs of the application. For example, when a 3D line segment is generated, an application-specific announce_3D_segment method could display the segment using the application's own display or plotting routines.

The application then calls api_ihl_set_output_manager with an instance of the new output manager class to make that instance the active manager. Then hidden line output is actually generated by calling api_ihl_compute_from_meshes.