Chapter 4. Classes

Topic: Ignore

The class interface is a set of C++ classes, including their public and protected data and methods (member functions), that an application can use directly to interact with ACIS. Developers may also derive their own classes from these classes to add application–specific functionality and data. Refer to the *3D ACIS Online Help User's Guide* for a description of the fields in the reference template.

PHLV5_EDGE

Class: Hidden Line Removal

Purpose: Defines an PHLV5 edge.

Derivation: PHLV5_EDGE : ENTITY_PHLV5 : ENTITY : ACIS_OBJECT : –

SAT Identifier: "phlv5_edge"

Filename: phlv5/phlv5_husk/sg_husk/phlv5/phlv5_edge.hxx

Description: Each PHLV5 edge contains pointers to modeling data, a pointer to a list of PHLV5_SEGMENTs and a pointer to a polyline representing the edge in 3D space. The polyline data can be used to display the hidden line view in an end user application.

Limitations: Not applicable

References:

PHLV5	PHLV5_SEGMENT
KERN	BODY, CURVE, ENTITY_LIST, FACE

Data:

None

Constructor:

```
public: PHLV5_EDGE::PHLV5_EDGE (
    CURVE* Curve=0,          // Curve
    BODY* Body=0,            // Body
    PHLV5_SEGMENT* Seg=0,    // Segment
    float * Point=0,         // Polyline points
    int n=0,                 // # points
    int MySense = 0          // Sense
    );
```

C++ initialize constructor requests memory for this object and populates it with the data supplied as arguments. Applications should call this constructor only with the overloaded new operator, because this reserves the memory on the heap, a requirement to support roll back and history management.

```
public: PHLV5_EDGE::PHLV5_EDGE (
    PHLV5_EDGE const& e      // PHLV5_EDGE to copy
    );
```

C++ copy constructor.

Destructor:

```
public: virtual void PHLV5_EDGE::lose ();
```

Posts a delete bulletin to the bulletin board indicating the instance is no longer used in the active model. The lose methods for attached attributes are also called.

```
protected: virtual PHLV5_EDGE::~PHLV5_EDGE ();
```

This C++ destructor should never be called directly. Instead, applications should use the overloaded lose method inherited from the ENTITY class, because this supports history management.

Methods:

```
public: void PHLV5_EDGE::clean ();
```

Cleans up pointers.

```
public: BODY* PHLV5_EDGE::GetBody () const;
```

Gets the body of the edge.

```
public: CURVE* PHLV5_EDGE::GetCurve () const;
```

Returns an ACIS curve. In the case of an edge representing a silhouette line a lazy evaluation of the curve will be performed. In order to maintain a suitable performance only call this member function when a curve is absolutely needed. In cases of general rendering, use the polyline returned from this same class..

```
public: int PHLV5_EDGE::GetNPoints () const;
```

Gets the number of points in the polyline list.

```
public: int PHLV5_EDGE::GetParam (
    double &SParam,          // Start parameter
    double &EParam           // End parameter
) const;
```

Gets the start and end parameters of the curve.

```
public: PHLV5_SEGMENT* PHLV5_EDGE::GetSegment ()
const;
```

Returns the pointer to the first segment.

```
public: void PHLV5_EDGE::GetSegmentList
     (ENTITY_LIST& seg_list )   // Entity list
    const;
```

Gets the segments in the form of an entity list.

```
public: int PHLV5_EDGE::GetSense () const;
```

Gets the sense of the edge.

```
public: float* PHLV5_EDGE::GetTabPoint () const;
```

Gets a pointer to the polyline points.

```
public: float PHLV5_EDGE::GetVal (
    int i                    // Point
    ) const;
```

Gets a specific polyline point.

```
public: void PHLV5_EDGE::SetBounds (
    double SParam,           // Start parameter
    double EParam            // End parameter
    );
```

Sets the start and end parameters of the curve.

Internal Use:	debug_ent, hook, identity, is_deepcopyable, next, previous, restore_common, save, save_common, set_body, set_curve, set_next, set_previous, set_segment, type_name, unhook
Related Fncs:	is_PHLV5_EDGE

phlv5_options

Class: Hidden Line Removal

Purpose:	Specifies options for hidden line removal.
Derivation:	phlv5_options : ACIS_OBJECT : –
SAT Identifier:	None
Filename:	phlv5/phlv5_husk/sg_husk/phlv5/phlv5_opts.hxx
Description:	Options for hidden line removal are supplied to APIs via this class. Use the “set” and “get” methods to build the object with the desired option settings, then pass this as the last argument to api_phlv5_compute.

The options that may be set are:

Resolution	The smallest meaningful quantity in PHLV5. Reduce this value if you are viewing parts modeled in very small coordinates. The default is 0.001.
Sag Resolution	The resolution used to determine if two lines are occluding each other or not. The default is 0.02.
Self–Calibration	If this is TRUE, PHLV5 computes its own resolution values based on the model size. This is highly recommended. The default is TRUE.
smooth_cosine	Sets the cosine of the angle at which two faces can meet and still be considered tangentially smooth. A value of 1.0 will produce no smooth edges and essentially turns the option off (default). A value of 0.95 will trap reasonably smooth edges.
Surface Approximation	If this is TRUE, PHLV5 uses the approximations of the surface instead of the actual procedural surface. The default is TRUE.

The hidden line style can also be set in phlv5_options. However, it is used in the Scheme interface; it is ignored by api_phlv5_compute.

Limitations: Not applicable

References: None

Data:

None

Constructor:

```
public: phlv5_options::phlv5_options ();
```

C++ initialize constructor.

Destructor:

```
public:  phlv5_options::~phlv5_options ();
```

C++ destructor.

Methods:

```
public: phlv5_hidden_line_style
    phlv5_options::get_hidden_line_style ();
```

Gets the current hidden line style. The hidden line style is used in the Scheme interface; it is ignored by api_phlv5_compute.

```
public: double phlv5_options::get_resolution ();
```

Gets the resolution.

```
public: double phlv5_options::get_sag_resolution ();
```

Gets the sag resolution.

```
public: logical phlv5_options::get_self_calibrate ();
```

Gets the self–calibration option.

```
public: double phlv5_options::get_smooth_cosine ();
```

Gets the cosine of the angle between the two edges.

```
public: void phlv5_options::set_smooth_cosine (
    double smooth_angle     // cosine of the angle
                            // between the two edges
    );
```

Sets the cosine of the angle at which two faces can meet and still be considered tangentially smooth.

```
public: logical
    phlv5_options::get_use_approx_surface ();
```

Gets the surface approximation option.

```
public: void phlv5_options::set_hidden_line_style (
    phlv5_hidden_line_style _in_style
// Style to use
    );
```

Sets the hidden line style. The hidden line style is used in the Scheme interface; it is ignored by api_phlv5_compute.

```
public: void phlv5_options::set_resolution (
    double in_resolution    // Resolution
);
```

Sets the resolution.

```
public: void phlv5_options::set_sag_resolution (
    double in_resolution    // Sag resolution
);
```

Sets the sag resolution.

```
public: void phlv5_options::set_self_approx_surface (
    logical in_use_approx_surface   // Surface approx
flag
);
```

Sets the surface approximation flag.

```
public: void phlv5_options::set_self_calibrate (
    logical in_self_calibrate   // Self-cal. flag
    );
```

Sets the self–calibration option.

Related Fncs:

None

PHLV5_SEGMENT

Class: Hidden Line Removal

Purpose: Defines an PHLV5 segment.

Derivation: PHLV5_SEGMENT : ENTITY_PHLV5 : ENTITY : ACIS_OBJECT : –

SAT Identifier: “phlv5_segment”

Filename: phlv5/phlv5_husk/sg_husk/phlv5/phlv5_seg.hxx

Description: The PHLV5_SEGMENT contains information about each section of the edge, including its visibility and parameter range data.

Limitations: Not applicable

References:

by PHLV5	PHLV5_EDGE
BASE	SPAinterval

Data:

None

Constructor:

```
public: PHLV5_SEGMENT::PHLV5_SEGMENT ();
```

C++ allocation constructor requests memory for this object but does not populate it. The allocation constructor is used primarily by restore. Applications should call this constructor only with the overloaded new operator, because this reserves the memory on the heap, a requirement to support roll back and history management.

```
public: PHLV5_SEGMENT::PHLV5_SEGMENT (
    SPAinterval const& iv,  // parameter interval
    Phlv5SegSta st,         // state
    Phlv5SegVis vi          // visibility
    );
```

C++ initialize constructor requests memory for this object and populates it with the data supplied as arguments. Applications should call this constructor only with the overloaded new operator, because this reserves the memory on the heap, a requirement to support roll back and history management.

Constructs an instance, initializing the parameter interval, state, and visibility to the given values.

```
public: PHLV5_SEGMENT::PHLV5_SEGMENT (
    phlv5_segment& _in_segment  // Segment to copy
    );
```

C++ copy constructor.

Destructor:

```
public: virtual void PHLV5_SEGMENT::lose ();
```

Posts a delete bulletin to the bulletin board indicating the instance is no longer used in the active model. The lose methods for attached attributes are also called.

```
protected: virtual PHLV5_SEGMENT::~PHLV5_SEGMENT ();
```

This C++ destructor should never be called directly. Instead, applications should use the overloaded lose method inherited from the ENTITY class, because this supports history management.

Methods:

```
public: double PHLV5_SEGMENT::end_pt () const;
```

Returns the end visibility parameter of the segment.

```
public: SPAinterval PHLV5_SEGMENT::intval () const;
```

Returns the visibility interval of the segment.

```
public: double PHLV5_SEGMENT::start_pt () const;
```

Returns the start visibility parameter of the segment.

```
public: Phlv5SegSta PHLV5_SEGMENT::state () const;
```

Returns the state of the segment.

```
public: Phlv5SegVis PHLV5_SEGMENT::visibility ()
const;
```

Returns the visibility of the segment.

Internal Use: debug_ent, hook, identity, is_deepcopyable, next, previous, restore_common, save, save_common, set_next, set_previous, type_name, unhook

Related Fncs:

is_PHLV5_SEGMENT