



www.hoops3d.com

## TSA's HOOPS/ACIS Integration

<b>1 TSA's HOOPS/ACIS Integration .....</b>	<b>2</b>
1.1 INTRODUCTION .....	2
1.2 WHY INTEGRATE ACIS AND HOOPS? .....	2
1.3 HOW DO ACIS AND HOOPS RELATE ARCHITECTURALLY? ....	3
1.4 HOOPS/ACIS INTEGRATION.....	4
1.4.1 HOOPS/ACIS Bridge .....	4
1.4.2 HOOPS/ACIS Reference Application .....	4
<b>2 HOOPS-ACIS Bridge.....</b>	<b>5</b>
2.1 HOOPS-ACIS MESH MANAGER .....	5
2.2 HOOPS-ACIS UTILITY MAPPING ROUTINES .....	5
2.3 HOOPS SEGMENT STRUCTURE GENERATED .....	6
2.3.1 Preserve Color .....	6
2.3.2 Create Body Segments .....	6
2.4 MAPPING OF ACIS ENTITIES TO HOOPS GEOMETRY .....	7
2.5 HOOPS-ACIS BRIDGE API .....	7
2.5.1 HA_Init ( ).....	8
2.5.2 HA_Close ( ) .....	8
2.5.3 HA_Read_Sat_File ( ).....	8
2.5.4 HA_Delete_Entity_Geometry ( ) .....	8
2.5.5 HA_Render_Entities ( ) .....	8
2.5.6 HA_Set_Rendering_Options ( ).....	8
2.5.7 HA_Compute_Geometry_Keys ( ).....	9
2.5.8 HA_Compute_Geometry_Key_Count ( ) .....	9
2.5.9 HA_Compute_Entity_Pointer ( ) .....	9
<b>3 HOOPS-ACIS Reference Application.....</b>	<b>9</b>
3.1 REFERENCE APPLICATION COMPONENTS .....	10
3.1.1 HOOPS/MFC Integration Classes .....	10
3.1.2 HOOPS-MVO Classes .....	11



## 1 TSA's HOOPS/ACIS Integration

### 1.1 Introduction

Using pre-packaged components in the development of applications is becoming more and more popular in the software industry today. Software components are produced by development organizations focused on a specific area of expertise and provide external development groups access to a very tangible encapsulation of this expertise. Many of the leading CAD/CAM/CAE systems today use one or more component software libraries to quickly develop highly competitive products.

Spatial Technology's ACIS 3D Toolkit® and Tech Soft America's HOOPS 3D Graphics System® are two examples of industry proven software components used in the production of CAD/CAM/CAE applications today. Organizations using these components are able to significantly reduce overall development cost and risk while delivering more robust applications to market in less time.

**ACIS** is Spatial Technology's exact boundary representation solid modeler and is used in a large number of end user applications including Autodesk's AutoCAD and Mechanical Desktop, Applicon's Bravo, and Baystate's CADKEY.

The **HOOPS 3D Graphics System** is Tech Soft America's graphics software component providing 2D and 3D, interactive, vector and raster based graphics for leading technical software companies including PTC/Computervision, MicroCADAM, Autodesk, Mechanical Dynamics, and Fluent.

### 1.2 Why integrate ACIS and HOOPS?

There are several compelling factors that led the ACIS and HOOPS teams to integrate the two components:

- The components complement each other and represent world-class technology in their respective areas of expertise: modeling and graphics.
- The HOOPS 3D Graphics System has been designed to be a complete graphics system for MCAD applications.
- Both components provide cross-platform support and are available on all the PC and UNIX platforms.
- Several independent software developers had already successfully incorporated both components into their applications.

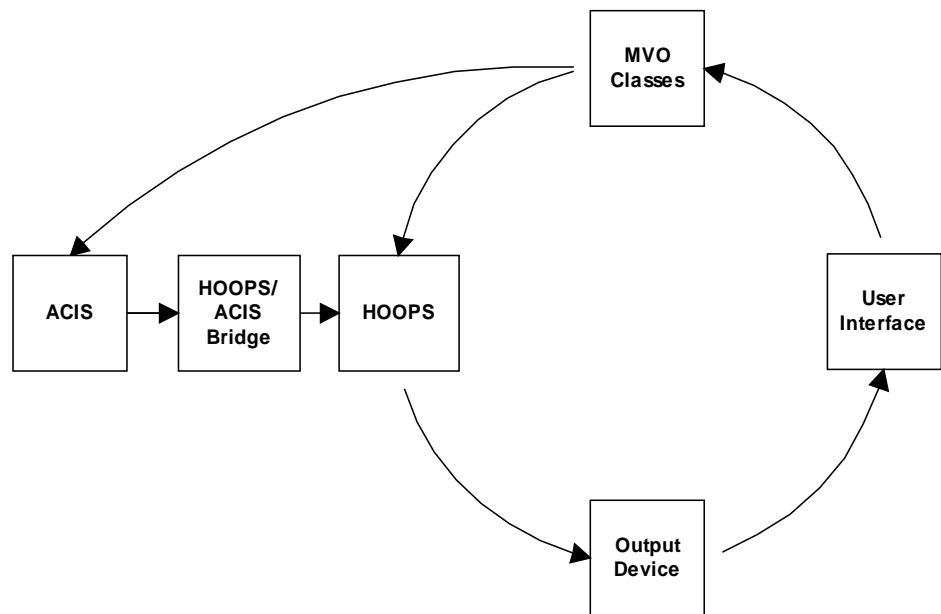


- A high-quality integration helps ensure correct application architecture while minimizing support issues. This leads to better products with shorter development cycles.

The integration of the two products provides developers with a base architecture and extensible framework for implementing their application while ensuring optimal design and rendering performance of applications built with both components. Developers may begin working immediately on higher level functionality, greatly reducing the time needed to prototype and implement applications.

### 1.3 How do ACIS and HOOPS relate architecturally?

There are six components in the architecture of an application that are relevant to the relation of ACIS and HOOPS:



- User Interface
- Model/View/Operator Objects– software objects for mapping user input to “operations” on the information in ACIS and HOOPS, the “models and views”.
- ACIS Modeling kernel
- HOOPS/ACIS Bridge – connects the two components
- HOOPS 3D graphics system
- Output Device – monitor, printer, etc...

The HOOPS/MVO framework consists of a class hierarchy that implements the Model/View/Operator paradigm – where a user is said



to operate on a model via one of its views. Thus users access Operators via the User Interface using them to create, manipulate, edit and query the information (the model and its views) in both ACIS and HOOPS.

A user of the application interacts with ACIS and HOOPS by invoking application code via the application's user interface. The application code consists of specific pieces of end user functionality implemented using the application programming interfaces (APIs) provided by ACIS and HOOPS. The HOOPS/MVO class library consists of several objects that implement common application level functionality.

## 1.4 HOOPS/ACIS Integration

Building applications with ACIS and HOOPS is certainly faster and easier than trying to build an in-house modeling kernel and graphics system from scratch. Developers choosing to use these two tools in tandem can start with integrating and then programming with the tools rather than waiting to develop them first.

To optimally integrate the two tools the developer must design a mapping of the ACIS tessellated information to the HOOPS graphics primitives, a connection to the desired GUI toolkit and a structure for the software objects which will interact with the ACIS and HOOPS components. Thus there still remains a significant amount of integration work necessary when using them together in an application.

Tech Soft America's integration of HOOPS with ACIS removes the need for developers to replicate this work, saving time and ensuring an optimally designed bridge between the two products.

### 1.4.1 HOOPS/ACIS Bridge

The integration functionality is encapsulated in the HOOPS/ACIS Bridge, a dynamic linked library (DLL) with an exposed API. The bridge module will be discussed in more detail later in this document.

### 1.4.2 HOOPS/ACIS Reference Application

The HOOPS/ACIS Reference Application is a MS Windows based application built with HOOPS and ACIS. It includes the ACIS modeling kernel, HOOPS, the HOOPS/ACIS Bridge, the MFC GUI toolkit and the parts of the HOOPS/AFC – application framework components.

The HOOPS/AFC modules used in the Reference Application are the HOOPS/MFC Integration classes and the HOOPS/MVO Classes. The application is built with the Microsoft Developer Visual C/C++ tools; the MSDev workspace is supplied including clear source for the application. Note that source code is not provided for the HA\_Bridge DLL itself.



The application implements the following functionality:

- Reads multiple ACIS part files (.sat) as well as HOOPS metafiles (.hmf).
- Direct manipulation of the model with the mouse including: modeling & viewing transformations, selection, & query
- Creation of new geometry: cylinder, sphere, & cone
- Boolean operations on user selected bodies (union, intersect, subtract)
- Printing, Print Preview, copy to clipboard, Windows metafile support

The application serves as a base architectural implementation for developers to extend by building custom HOOPS/MVO objects.

## 2 HOOPS-ACIS Bridge

ACIS developers who choose not use the ACIS graphics husk must override the standard methods in the Mesh Manager class. The ACIS geometry's graphical information is passed to the Mesh Manager methods and must be stored, optimized and appropriately formatted prior to sending it to a computer screen or printer. Most applications will need to maintain a two way mapping linking the ACIS entities to the geometric primitives in the graphics subsystem. Thus use of ACIS with an external graphics solution requires 1) a graphics subsystem and 2) a bridge between ACIS and that subsystem.

The HOOPS-ACIS Bridge (HAB) implements all the work necessary to integrate HOOPS and ACIS. The HAB maps the tessellation of ACIS models to HOOPS geometric primitives, enabling ACIS models to be displayed and interacted with via the HOOPS graphics system, and the HAB maintains a two way mapping which facilitates the development of highly interactive editing and query based applications.

The HAB ships with the HOOPS 3D Graphics System as a separate dynamic link library with its own API.

### 2.1 HOOPS-ACIS Mesh Manager

ACIS requires extensions to the existing mesh manager classes in order to provide sufficient information to render ACIS bodies and their edges to HOOPS primitives.

### 2.2 HOOPS-ACIS Utility Mapping Routines

To provide a good API for picking and other operations, it is necessary to provide functions to translate between HOOPS entities and ACIS entities. The HA\_Compute\_ functions permit the user to convert a HOOPS key to a corresponding ACIS entity and vice versa.



## 2.3 HOOPS Segment Structure generated

When ACIS Bodies or Faces are rendered via the HA\_Render\_Entity routine, the HAB assumes an open HOOPS segment. The HAB then creates a segment structure underneath the currently open segment and populates it with geometry corresponding to the ACIS entities rendered.

The HAB API provides control over how the ACIS entities are mapped to HOOPS via the routine HA\_Set\_Rendering\_Options(). The actual segment structure generated under the open segment depends on which options are chosen. The options are:

- Preserve Color
- Create Body segments

As the HAB assumes the existence of an open HOOPS segment, the user has maximum flexibility over the structure of the top part of the HOOPS segment tree. For instance, an application working with assemblies of ACIS bodies will create the appropriate assembly structure in the HOOPS segment tree and then open the appropriate segment in the tree when rendering bodies of the assembly.

### 2.3.1 Preserve Color

The ACIS color attributes can be preserved or ignored. If they are preserved, a HOOPS segment will be created for each unique color attribute in the entities being rendered. In each created segment, the HOOPS color will be set to match the color attribute of the entities which are being rendered into that segment. The HAB will automatically segregate the face and edge geometry into the appropriate segment based on the rendered entity's color. E.g., if three ACIS faces in a body are being rendered two of which are blue and one of which is red and the preserve color HAB option is on, two HOOPS segments will be created under the open segment. One will have a blue color attribute and one will have a red; the two blue ACIS faces will then be mapped to two HOOPS shells which will be inserted into the blue HOOPS segment and the one red face to one shell in the red segment. Depending on the settings passed to HA\_Set\_Rendering\_Options, there are different ways for the color segment hierarchy to be created. See the HOOPS-ACIS reference manual for further details.

### 2.3.2 Create Body Segments

This option controls whether or not a HOOPS segment will be generated for each ACIS body entity by the HAB. See the HOOPS-ACIS reference manual for further details.



## 2.4 Mapping of ACIS entities to HOOPS geometry

The HAB maps the tessellation of all ACIS entities to HOOPS geometric primitives, inserting zero or more new HOOPS geometric primitives into the HOOPS database for each ACIS entity encountered. While each ACIS entity uses a pointer for access, HOOPS entities have “keys”, long integers, which are passed to routines when accessing the entity. The HAB API provides routines for converting ACIS pointers to HOOPS keys and vice versa.

The HAB API provides control over how the ACIS entities are mapped to HOOPS via the routine `HA_Set_Rendering_Options ( )`. These options can be configured to:

- Merge Faces → Merge all Faces for an ACIS Body into the minimum number of uniquely colored shells
- Merge Bodies → Merge all Faces for *all* ACIS Bodies rendered during the duration of a *single* call to `HA_Render_Entities` or `HA_Load_Sat` file into the minimum number of uniquely colored HOOPS shells

Note that if the “preserve color” rendering option is off, then the minimum number of uniquely colored HOOPS shells mentioned above will always be exactly one.

ACIS Edges map to one or more of the following HOOPS primitives depending on the form of their tessellated geometry generated by the ACIS modeler.

- HOOPS Polyline
- HOOPS Elliptical Arc
- HOOPS Line

The HAB API provides for control over how elliptical and circular edges are mapped to HOOPS. They can be mapped to elliptical arcs or to polylines. HOOPS elliptical arcs adaptively tessellate and offer the highest degree of visual fidelity to the model for all ranges of interaction. Polylines should be used when it is more important for the tessellation of the ACIS edge to be coincident with the tessellation of the ACIS face and/or the HOOPS true hidden line algorithm is being used.

## 2.5 HOOPS-ACIS Bridge API

The HOOPS specific integration of the ACIS Mesh Manager and associated tools is supplied as a dynamic link library (DLL). The library’s API consists of the following function calls. Developers will include the HOOPS/ACIS DLL in the list of libraries needed for building their application and reference the HOOPS/ACIS header file in the files using these API calls.



## 2.5.1 HA\_Init ( )

Initializes the ACIS kernel and tessellation husk; creates the HOOPS-specific implementation of the Mesh Manager; initializes HOOPS.

## 2.5.2 HA\_Close ( )

Shuts down the ACIS kernel and HOOPS 3D Graphics System. Frees all memory allocated by the HOOPS/ACIS Bridge module.

## 2.5.3 HA\_Read\_Sat\_File ( )

Parses the input ACIS file, populates the ACIS kernel with its contents, and creates the corresponding geometry in the HOOPS database using the ACIS renderer and HOOPS/ACIS Bridge.

## 2.5.4 HA\_Delete\_Entity\_Geometry ( )

This routine will delete the geometric primitives associated with the given ACIS entities from the HOOPS database and update the mapping between the ACIS pointers and the HOOPS keys.

## 2.5.5 HA\_Render\_Entities ( )

This routine encapsulates and replaces the ACIS **api\_mesh\_entity** routine for users of the HOOPS/ACIS Bridge. It renders all ACIS solid and sheet bodies.

User should call this when they want to generate the tessellation of the ACIS model and store it in HOOPS via the HOOPS/ACIS Bridge.

The user should not call **api\_mesh\_entity** between the HA\_Init and HA\_Close.

## 2.5.6 HA\_Set\_Rendering\_Options ( )

This routine provides control over how ACIS objects are rendered to HOOPS objects via the HOOPS/ACIS Mesh Manager. In particular, it provides control over:

- Which ACIS entities to render: Face, Edges, or both.
- Whether to partition the HOOPS geometry according to their ACIS color attributes
- Whether or not to merge the geometry for all ACIS' faces for a body into the minimum number of uniquely colored HOOPS shells
- Whether or not to compress the geometry for all ACIS' bodies into the minimum number of uniquely colored HOOPS shells
- The HOOPS segment structure generated



- Whether or not to tessellate ACIS ellipses into dynamically or statically tessellated HOOPS primitives (elliptical arcs or polylines)

## 2.5.7 HA\_Compute\_Geometry\_Keys ( )

Given an ACIS pointer to any entity, this routine will return the array of HOOPS keys for the geometry in the HOOPS database generated by the HOOPS/ACIS Bridge provided that the entity has been tessellated by a HA\_Render\_Entity or via a HA\_Read\_SAT\_File( ) call.

## 2.5.8 HA\_Compute\_Geometry\_Key\_Count ( )

Given a pointer to any ACIS entity, this routine will return the number of the corresponding geometric entities in the HOOPS database

## 2.5.9 HA\_Compute\_Entity\_Pointer ( )

Given a HOOPS Key this routine will return the pointer to the ACIS entity that contains the specified HOOPS geometric entity.

## 3 HOOPS-ACIS Reference Application

The HOOPS-ACIS Reference Application is a reference implementation of an MFC application built with ACIS, HOOPS, the HOOPS-ACIS Bridge and the HOOPS-AFC. It is part of the HOOPS-ACIS integration and is supplied in executable and source form. The application was built with the Microsoft Visual C/C++ tools and includes the MSDev project workspace and source code.

Building the project produces a multi-document MFC application that

- Reads and displays ACIS parts files (SAT)
- Reads and displays HOOPS metafiles (HMF)
- Provides simple model creation (sphere, cylinder, cone)
- Provides simple boolean operations (add, subtract, intersect)
- Prints (including Print Preview, Copy to clipboard and Windows Metafile)
- Provides FACE, EDGE, BODY level selection



## 3.1 Reference Application Components

The diagram below shows the relationship of the various pieces of the HOOPS-ACIS Integration within an application built for the Microsoft Windows platform.

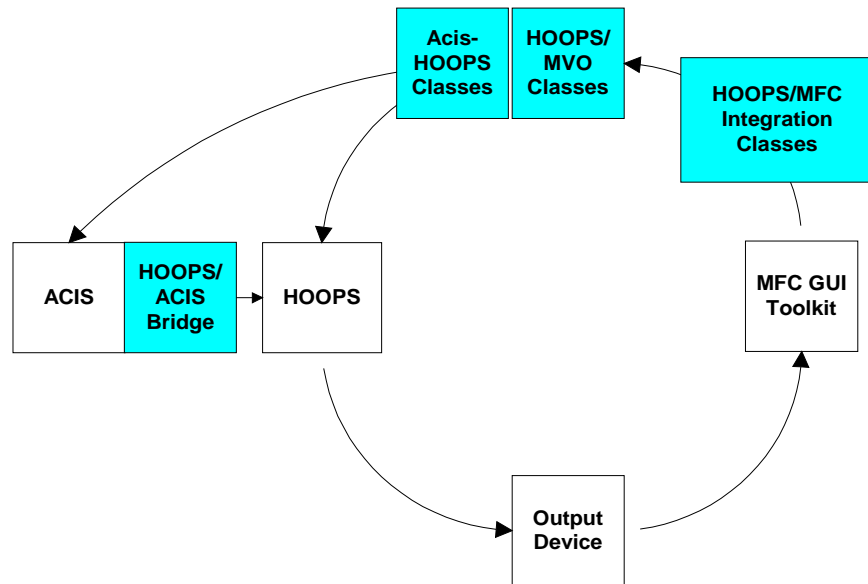


Figure 3.1 : HOOPS/ACIS Reference Application Components

### 3.1.1 HOOPS/MFC Integration Classes

Applications built for the Microsoft Windows environment typically use the Microsoft Foundation Classes (MFC) for implementing a graphical user interface (GUI). The work needed to connect HOOPS to an output window in a MFC GUI is encapsulated in the HOOPS-MFC integration classes. They configure HOOPS appropriately and manage application level issues such as:

- Integrating HOOPS within the MFC Application/Document/View paradigm
- Managing shared color palettes
- Printing and Print Preview
- Sending 3D output to the Windows clipboard and metafile



## 3.1.2 HOOPS-MVO Classes

The HOOPS-MVO Classes are a set of C++ objects that provide services found to be needed by many MCAD applications. They are platform and GUI independent and are designed to integrate seamlessly with the HOOPS-MFC integration classes and the HOOPS-MOTIF widget provided with the HOOPS 3D graphics system.

These classes provide a framework for implementing application functionality using HOOPS and ACIS. They include classes for creating models, views of the models, and interacting with models and views.

### 3.1.2.1 HOOPS-ACIS Specific HOOPS-MVO classes

The AcisHOOPS specific classes are derived from both the HOOPS-MVO classes and the HOOPS-MFC integration classes. They include:

**CAcisHoopsView** – derived from CHoopsView; handles Windows UI messages on the view, including mouse messages, toggling of the active Operator, etc...

**CAcisHoopsDoc** - derived from CHoopsDoc; handles loading of ACIS and HOOPS files.

**HAcisView** – derived from HBaseView; stores and manages the HOOPS database objects associated with the view, along with view-specific data needed by the application

**HAcisModel** - derived from HBaseModel; creates and manages the HOOPS Include Library objects associated with the model that is either being created, or was read in from a file