QUAOAR STUDIO LLC

**CUSTOMER:** INTERNAL **PROJECT:** COMMUNITY

**PROJECT REF.:** N/A **DELIVERABLE:** TECHNICAL REPORT

| TITLE | **COMPARING OPEN CASCADE KERNEL WITH ACIS MODELER** |
|---|---|
| TYPE | TECHNICAL REPORT |

**DOC. NUMBER:** -

| Circulation list | |
|---|---|
| FROM | |
| For action (TO) | |
| For information (CC) | |

**History**

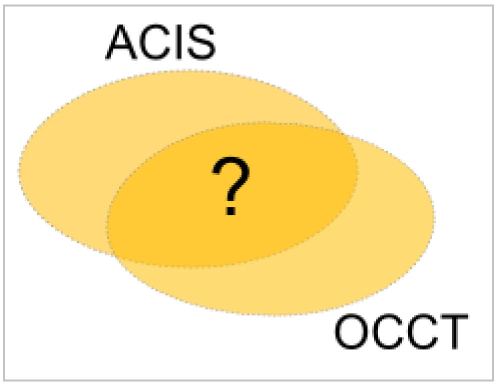
| Version | Date | Author | Description | Status |
|---|---|---|---|---|
| 1.0 | 27.08.2014 | Sergey Slyadnev | From scratch | D |
| 2.0 | 11.12.2023 | Sergey Slyadnev | Revision of references (ACIS docs became closed), review and update of the text. | A |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**D**: Draft, **P**: Pending approval, **A**: Approved

# CONTENTS

# 1. MOTIVATION

The following text provides a concise summary of the geometric modeling capabilities found in ACIS and Open CASCADE Technology (OCCT) geometric kernels. Essentially, both libraries are comparable as they tackle similar technical problems related to the CAD/CAM/CAE domain. Nevertheless, their functionality may vary to some degree. This document aims to provide a brief analysis of the current disparities between OCCT and ACIS. Conducting a thorough investigation of the ACIS kernel is challenging due to its proprietary nature. Furthermore, to fully comprehend the advantages and disadvantages of each specified geometric kernel, one would undoubtedly need practical experience in utilizing both. Thus, we confine this study to an examination of the various instruments without delving into their effectiveness and resilience.




ACIS [1] is an SDK developed by Spatial Corp, a division of Dassault Systemes since 2000. Additionally, it should be noted that Spatial Corp provides another geometric kernel called CGM (CATIA/Convergence Geometric Modeler). The article "Spatial, ACIS, CGM, and the future of geometric modelers" [2] provides a concise summary of both products. This report contains no information about CGM. It is important to mention that the data interchange capabilities are independent of the ACIS modeler and can be accessed through the InterOp product. Nevertheless, we suggest that InterOp components are classified under the same category as ACIS itself, consequently implying that data exchange functionality is not excluded from this examination.

This document is relevant for the following versions of ACIS and OCCT:

- 3D ACIS Modeler and 3D InterOp R24 (SP2), February 2014.
- Open CASCADE Technology (development version; the initial comparison was conducted in August 2014, updated in December 2023).

The following functionality is excluded from the present analysis:

- Direct editing, such as push/pull or face tweaking operations.
- Hidden Line Removal (HLR) algorithms.
- Some advanced algorithms that were not observed during ACIS evaluation period (during the time when ACIS documentation was publicly accessible).
- Commercial packages (API extensions) by OCC that are not provided in open source.
- Sheet metal processing API.

We acknowledge that the API of ACIS may have undergone significant changes in terms of its incorporated functionality since the initial assessment conducted in 2014. As of 2023, the Quaoar Studio team is no longer able to obtain the latest ACIS binaries and documentation. Nevertheless, we believe that the present column-to-column comparison remains important for acquiring a more profound comprehension of the scope of OpenCascade functionality. This comparison offers a different perspective for assessing OpenCascade, facilitating a more thorough evaluation of its range of capabilities.

## 2. Criteria

Both ACIS and OCCT offer implementations of the widely recognized B-rep (Boundary Representation) paradigm for solid modeling, resulting in highly similar underlying principles. These principles encompass a distinction between *geometry* and *topology*, as well as the utilization of conventional techniques for modeling and other fundamental mathematical methods. Hence, we can evaluate the libraries by examining their fundamental characteristics and progressing to more sophisticated modeling approaches. The following categories of functionality are being considered:

| | |
|---|---|
| Ecosystem | Essential tools and classes that serve as the fundamental basis for a modeler. |
| Basic Math | Procedures for fundamental mathematical operations, such as linear algebra, optimization, locating function roots, etc. |
| Basic Geometry | The available geometric primitives that include conics, quadrics, parametric curves, and surfaces. This module may also include low-level techniques of evaluation, e.g., De-Boor algorithm for spline evaluation, etc. |
| Basic Geometric Tools | Tools for the construction, modification, and analysis of geometric primitives. E.g., curvature analysis, surface extension, interpolation, approximation, intersections, etc. |
| Basic Topology | The contents of topological structures, the usage of tolerances, the application of isometric transformations, assemblies.<br><br>Topology is a mathematical discipline that focuses on the study of spatial characteristics that remain unchanged when objects are deformed in a continuous manner, without tearing or gluing. These characteristics are known as topological invariants. Topology, loosely speaking, describes a "rubber" model that does not have a fixed position in space. For instance, a circular edge and an oval edge possess the same topology, but differ geometrically. Similarly, a square face and a trapezoidal face are topologically identical, meaning they have the same number of edges and vertices, but they are not geometrically equivalent in terms of their shape and proportions. When a topological entity is linked to a geometric entity, its shape, location, and orientation become permanently defined in space. In ACIS, OCCT, and other widely-used modelers, topology refers to the relationships of adjacency and connectedness between different objects inside the model. This is the foundation of a Boundary Representation (B-rep). |
| Basic Modeling | The core modeling functionality that has become an established standard for any B-rep kernel. |
| Advanced Tools | The advanced modeling functionality that can be useful in specific engineering fields. |
| Data Exchange | Interoperability functions with other CAD kernels and platforms. |
| Application Infrastructure | End-user’s application development API. |

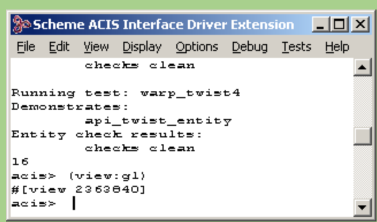
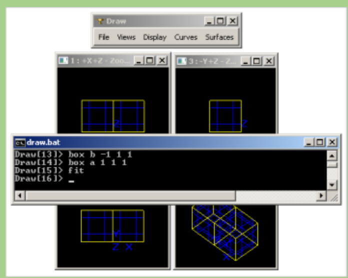
The analysis findings are visually highlighted using three different coloring methods:

| | | |
|---|---|---|
| The functionality is considered sufficient. | The functionality is limited or cumbersome compared to the “opponent” library. | The functionality is not available or malfunctioning. |

# 3. ANALYSIS

## 3.1. Ecosystem

| | ACIS | OCCT |
|---|---|---|
| Programming Language | C++. | C++.<br><br>SWIG-based wrapper can be utilized in order to run OCCT functionality with different languages (Java, C#, Python etc). |
| Standard C++ Types<br><br>Whether type aliases are used or not. | Type aliases are rarely used. However, ACIS often favors the utilization of small objects rather than incorporating built-in types (for example, a parameter on a curve or surface is defined with a distinct C++ class).<br><br>ACIS primarily attempts to utilize standard C++ types and functions to the highest degree possible. If a programmer has knowledge of conventional C++ capabilities, including STL, he will not need to invest excessive time in studying the ACIS environment. | Type aliases are used: int = Standard_Integer, double = Standard_Real etc. |
| Units | Unitless. Conversion of units is supported in CAD translators. | By default, millimeters are used as the unit of measurement. However, the modeling capabilities are not affected by this choice. CAD translators have the capability to do unit conversions. |
| Memory Management<br><br>Memory allocation/deallocation is a usual hotspot in modeling operations. Normally, heavy algorithms prefer using "memory arenas" instead of "new/delete" functions. Different types of "arenas" are implemented by means of memory allocator classes. | The ACIS memory management system enables the monitoring of memory leaks, gathering data on memory usage, and allowing for customization of memory allocation and deallocation. The configuration parameter allows for the activation of a memory leak audit. | The legacy memory management of OCCT effectively handles the allocation and deallocation of multiple C++ objects. Subclasses of the kernel's Standard_Transient class have automatic memory management enabled. In addition to the legacy memory manager, it is possible to utilize the standard OS allocation/deallocation functions and scalable allocator provided by Intel TBB (c). The selection of these management strategies is determined via a |

| | | |
|---|---|---|
| | | configuration variable. There is no audit of memory leaks. |
| Implicit Initialization and Terminating<br><br>Some libraries require explicit initialization and termination of their internal resources. | Explicit initialization with functions "api_start_modeller" and "api_stop_modeller" is necessary. | No explicit initialization is required. There is no any notion for a “modeling scope” or a “session” in OpenCascade. |
| Exceptions<br><br>How modeler treats exceptions. | ACIS attempts to be exception-safe. It has embedded error-tracking capabilities: each API function returns an error code (wrapped with a value-object called "outcome"). | OpenCascade is not exception-safe. It throws an exception whenever its functionality is used incorrectly. All exceptions are derived from base Standard_Failure class. Error codes are also used in some contexts. |
| Scripting<br><br>Entry point to the modeler. | Scheme ACIS Interface Driver Extension (AIDE): command-line application giving interface to the library by Scheme-based commands (Lisp dialect).<br><br>Subjectively, we found Scheme AIDE less ergonomic than OpenCascade’s DRAW console: Lisp-like syntax is more sophisticated compared to Tcl; camera behaves less friendly (rotations are not calibrated around center of the model); viewer IDs are not easy to track; FPS seems to be worse even on simple scenes.<br><br> | Test Harness Draw: Tcl-based command-line application giving interface to the library.<br><br> |
| High-level "Powerful" API<br><br>Geometric modeling libraries often necessitate writing of extensive C++ (or scripting) code, even for simple operations. In order to enhance usability, well-known modelers frequently | ACIS possesses an API that comprises global methods with names prefixed by "api_". Besides APIs, there are also "direct interface" functions accessible, which can be either global functions or class-level functions. | OCCT is completely class-oriented. Global functions are rarely used. API is available via dedicated classes (e.g., in BRepPrimAPI package). |

| | | |
|---|---|---|
| offer a "superlayer" API that is both straightforward and robust, encapsulating a significant portion of the underlying low-level functionality. In addition, such API typically manages input validation and error reporting. | | |
| Possibility to use low-level structures<br><br>Occasionally, the existing API may include excessive intelligence, making it a better choice to utilize lower-level functions. For example, an API for lofting may employ the reconstruction of topological structures (contours) that are redundant in purely geometric use cases. Lower-level geometrical procedures can be employed more efficiently in such situations. | The available API has limitations in terms of the capability it offers for typical geometric modeling applications. The usual "style" of ACIS-based code follows a structure similar to this:<br><br>`API_BEGIN`<br>`  api_*(...)`<br><br>`  api_*(...)`<br>`  ...`<br>`API_END`<br><br>E.g., to create a solid box, one would write the following code:<br><br>`API_BEGIN`<br>`  BODY* body;`<br>`  api_make_cuboid(..., body);`<br>`API_END`<br><br>OCCT offers an equivalent to "api_make_cuboid", yet it additionally permits constructing the "box" shape from elementary objects, giving you complete control over their contents and relationships. ACIS appears to have more restrictive rules when it comes to unusual usage scenarios. | All low-level functions and data structures are available even if they are only used by OCCT internally. Still, composing a shape bottom-up from such entities is a laborious and error-prone process. |
| Global (Static) Variables<br><br>Global variables retain their values during the whole work session. These variables can be utilized to modally influence the behavior of the library until their values are subsequently altered. | Available.<br><br>Even topology correctness rules may vary slightly depending on the global options. For instance, depending on "periodic_no_seam" option, ACIS will require or not the presence of a seam edge on faces with underlying periodic surfaces. Thus, a cylindrical face may have two disconnected bounding loops without a seam in-between. | Available.<br><br>It is usual to configure interoperability options using the global variables (e.g., units of measurement). |

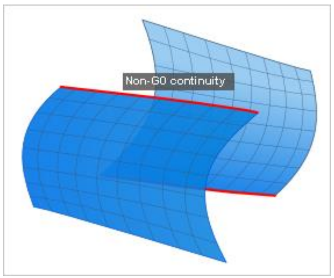
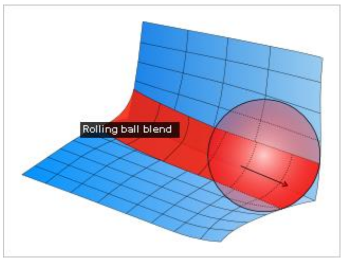
### 3.2. Basic Math

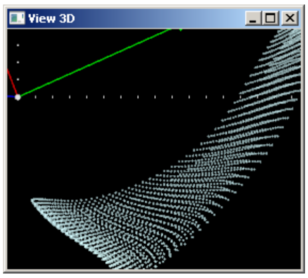
| | ACIS | OCCT |
|---|---|---|

| | | |
|---|---|---|
| Linear Algebra<br><br>Geometric modeling algorithms heavily rely on linear algebra techniques, such as solving linear equations, matrix decomposition, and calculating eigenvalues. Some modelers choose to independently implement linear algebra, whereas others opt to utilize third-party tools. | ACIS uses Eigen library.<br><br>As ACIS is a proprietary product we cannot analyze how deep this dependency is.<br><br> | All essential Basic Linear Algebra Subprograms (BLAS) are internally implemented. OpenCascade's use of its own linear algebra ensures complete independence from third-party libraries. At the same time, it is acknowledged that OpenCascade's inherent implementation of BLAS may be less accurate and efficient compared to the one found in well-known and supported libraries like Eigen. |
| Numerical Methods<br><br>Many modeling operations need to solve variational and optimization problems. In such circumstances, one can employ the widely recognized numerical techniques for locating function roots or extremums. | ACIS Laws component provides a uniform interface for a set of numerical tools. In addition to evaluation methods associated with Laws, several numerical functions are available:<br><br>– Finding global maximums and minimums;<br>– Finding multidimensional local minimums;<br>– Performing numerical integration;<br>– Finding function roots;<br>– Numerical differentiation. | Numerical methods covering the same problems are available in math package of OCCT (TKernel + TKMath pair of libraries). |

### 3.3. Basic Geometry

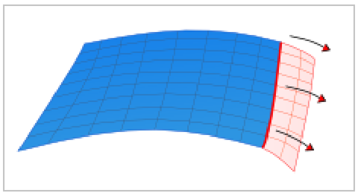
| | ACIS | OCCT |
|---|---|---|
| 3D Precision<br><br>The minimum perceptible distance at which two points are regarded as coincident. | SPAresabs (1.0E-6). | Confusion (1.0E-7). |
| Continuity Requirements<br><br>The minimum required order of smoothness for the geometry. The codes G0-GN are employed to indicate *geometric continuity*, which is not influenced by parameterization. On the other hand, the codes C0-CN are utilized to indicate *parametric continuity*, which is determined by the chosen parameterization. | G0 is acceptable but strongly discouraged as point perpendiculars cannot be evaluated in non-G1 discontinuity zones. | G0 is accepted in general but some algorithms (e.g. offsets) require G1 or C1 continuity. |

| | | |
|---|---|---|
|  | | |
| Analytic Curves<br>Special cases for curves. | Straight lines, ellipses and helices.<br>Circle is not presented in ACIS: special case of ellipse is used instead. | Straight lines, all conic curves (circles, ellipses, parabolas, hyperbolas). |
| Other Curve Types<br>Free-form parametric curves and special-type curves. | Composite curves, B-spline curves. | B-spline curves, Bezier curves, offset curves. |
| Canonical Surfaces<br>Special cases for surfaces. | Planes, cones, spheres, tori. | Planes, cones, spheres, tori, cylinders. |
| Other Surface Types<br>Free-form parametric surfaces and special-type surfaces. | B-spline surfaces, blend surfaces, ruled surfaces, sum surfaces, offset surfaces, surfaces of revolution, skinned surfaces, net surfaces, swept surfaces, surfaces defined by Laws. | B-spline surfaces, Bezier surfaces, ruled surfaces, surfaces of revolution, swept surfaces, offset surfaces. |
| Procedural Curves and Surfaces<br>Parametric curves and surfaces that are defined declaratively as procedures. There are no explicit parametric equations available for evaluation. For instance, a curve can be precisely described as the intersection between two surfaces. A surface can be defined as path swept by a rolling ball contacting two support surfaces.<br> | Such definitions are typically more compact and precise than spline approximations. Also, the procedural definitions are more immutable compared to the associated approximations, as they establish an ideal approach to create geometry (which may be restored with a chosen level of accuracy). Nevertheless, procedural geometry places significant demands on third-party modelers, as the modeling capabilities they offer must be sufficiently robust to handle such complex geometry. It is common practice to derive standard (spline) | API level does not support procedural geometry. By employing adapters, it becomes feasible to define a wide range of parametric curves and surfaces. Some algorithms utilize this method internally, however the outcome is often provided as a NURBS approximation.<br>We do not consider the entities like offset curves, offset surfaces, surfaces of revolution, etc. as procedural as they have deterministic parametric equations. |

| | | |
|---|---|---|
| | approximations from these declarations and to omit the original declarations. | |
| Point Clouds<br><br>Point clouds are intensively used in metrology, reverse engineering, visualization and other engineering fields.<br><br> | ACIS provides dedicated tools for working with point clouds. These tools allow not only storing the 3D positions of points, but also performing Boolean operations and compaction. | OCCT does not provide any tools tailored to point cloud processing (except for some AIS classes). However, the most of desired functionality can be implemented with use of existing common tools. |

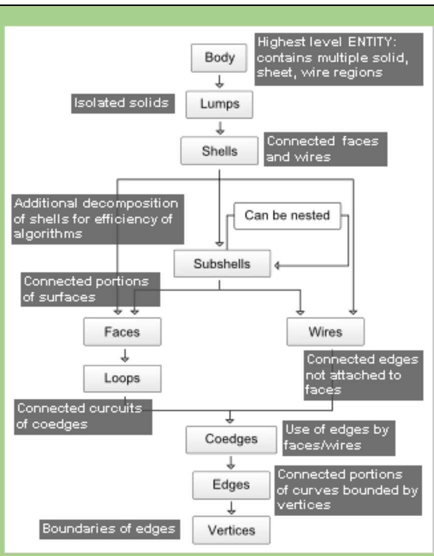
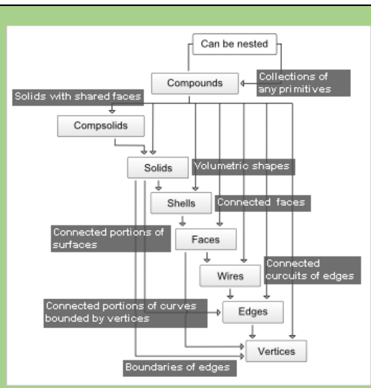
## 3.4. Basic Geometric Tools

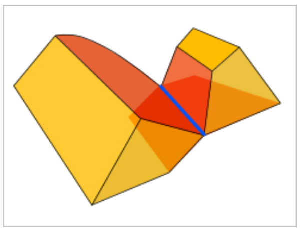
| | ACIS | OCCT |
|---|---|---|
| Geometry Validation<br><br>Possibility to check geometric primitives for errors in definition. | Self-checks are available in each geometric primitive via a dedicated method. List of invalidities is returned as a result. | Validity inspections are typically conducted throughout the primitive construction phase. An exception is thrown if the defined properties are inconsistent, e.g., if the number of poles in a B-spline curve does not correspond to the B-spline degree and knot vector. Other than that, Boolean operations may come with their own ad-hoc checks. Also, the high-level BRepCheck_Analyzer class incorporates fundamental validity checks (the check list implemented by this tool is the central one in OpenCascade library). |
| Differential Properties<br><br>Possibility to query local properties of curves and surfaces, such as derivatives, Gaussian/Mean/Principal curvatures, etc. | + | + |
| Intersections<br><br>Curve-curve, surface-surface, surface-curve intersections. | + | + |
| Extending Surface | + | + |

| | | |
|---|---|---|
| Surface extrapolation out of its parametric domain.  | | |
| Projections<br>Projection of a point to a curve, a point to a surface, a curve to a surface. | + | + |
| Calculation of Distances<br>Distances between geometric primitives. | + | + |
| Conversion of Analytic Geometry into B-spline Geometry<br>Can be useful in some data interoperability circumstances. | + | + |

## 3.5. Topological Model

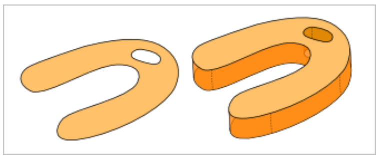
| | ACIS | OCCT |
|---|---|---|
| Modeling Type<br>Solid modeling, surface modeling, wireframe modeling, CSG (Constructive Solid Geometry) etc. | Solid, surface, and wireframe modeling.<br>Cellular modeling based on representation of geometry with volumetric and sheet blocks (see below) is also allowed. | Solid and surface modeling. |
| Completeness of Topological Model<br>Types of the supported topological entities and their relationships (including iterators and circulators). | Represented by the following diagram: | Represented by the following diagram: |

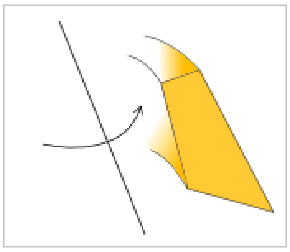
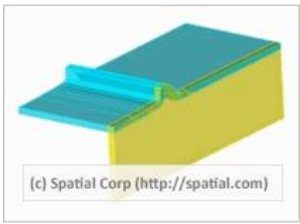
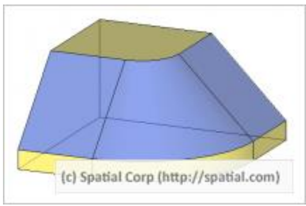
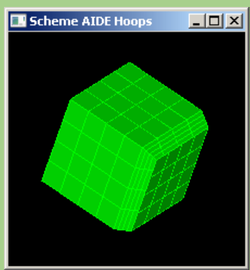
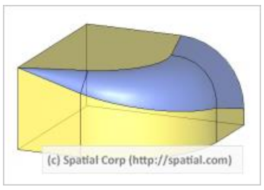
| | | |
|---|---|---|
| | <br><br>The child entities store references to their parent entities. References to the next and previous entities are also kept (e.g., the next COEDGE for the given one in a loop).<br><br>Geometry can be shared among multiple topological entities. For example, two faces may refer to the same surface. This is beneficial for minimizing memory usage. | <br><br>The child entities do not store references to the parent entities. Utilization of specialized exploration tools is necessary to locate parent entities for the provided child. The entity does not store references to its next and prior entities at its own level.<br><br>Geometry can be shared among topological entities, although most algorithms generate shapes where each object has its own separate copy of the underlying geometric primitive. This might be beneficial for preventing data races in parallel calculations, although, such logic and the corresponding checks are not provided "out of the box."<br><br>Topological model of OCCT is basically inspired by ISO 10303-42. |
| Transformations<br><br>Orthogonal transformations (do not assume modification of geometric shape). | Applied at the level of BODIES. | It is possible to associate transformations with any topological primitive, although it is strongly advised not to use them on primitives other than "bodies". |
| Non-manifold Topology<br><br>Intuitively, a manifold surface has the property that, around every one of its points, there exists a neighborhood that is homeomorphic to the plane. | + | + |

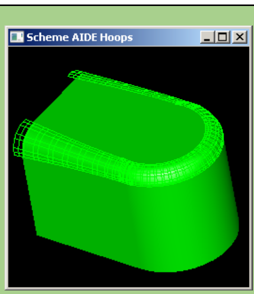
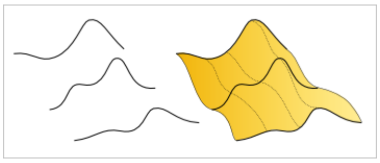
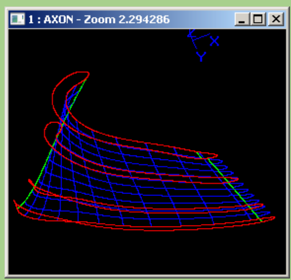
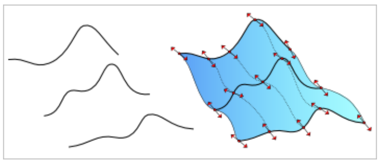
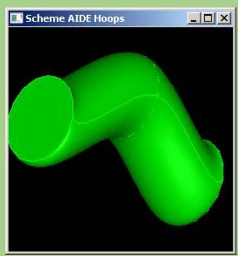
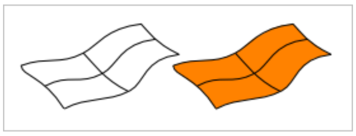
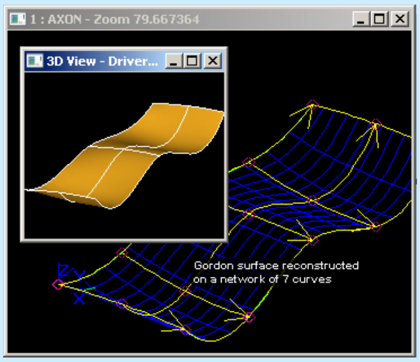
| | | |
|---|---|---|
|  | | |
| Tolerant Modeling<br><br>An essential aspect of a reliable geometric modeling system is the efficient handling of geometric imprecisions. | Only one tolerance, known as SPAresabs, is utilized in 99% of geometric comparisons. Additionally, there exists a specialized precision value called the SPAresnor, which is specifically designed for angle comparisons. This tolerance is determined by the value of SPAresabs and the maximum distance between two points in the model. According to the ACIS documentation, it is not recommended to modify this value.<br><br>Convergence criteria for numerical methods use precisions different from tolerance: e.g., SPAresfit value is used for B-spline approximations.<br><br>A single global tolerance may seem inadequate in certain situations, such as in data exchange workflows. When dealing with such circumstances, ACIS employs what are known as "tolerant entities": edges and vertices that have corresponding local tolerance values. The entities that exhibit local tolerances are represented with specialized data structures, such as TCOEDGE instead of COEDGE, TEDGE instead of EDGE, and TVERTEX instead of VERTEX. ACIS has the ability to transform from a non-tolerant entity to a tolerant entity if it deems this conversion to be essential. There are no specific tolerances for B-rep faces. | In the ACIS terminology, all topological entities in OCCT are considered tolerant entities, meaning that specific tolerance values are assigned to vertices, edges, and faces. The tolerance of a lower-level entity must exceed the tolerance of the corresponding upper-level entity. For example, the tolerance of a vertex should not be smaller than the tolerance of its owning edge.<br><br>Convergence criteria for numerical methods use dedicated precisions. E.g. Precision::Approximation() is used for B-spline approximations. |
| Cellular Topology<br><br>An addendum to B-pep has been developed to enable the representation of a model using a set of volumetric (3D) or sheet (2D) cells. In a | ACIS offers alternate data structures specifically designed for cellular topology. Both volumetric and sheet cells are allowed. 2D cells may surround cavities in their interior. These structures are connected to a B-rep model by attributes. The latter binding enables the preservation of | OpenCascade's scope is limited to B-rep representation schemes. However, there may have been some legacy voxelization API found in past releases of the library. |

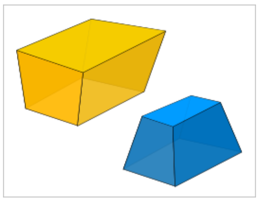
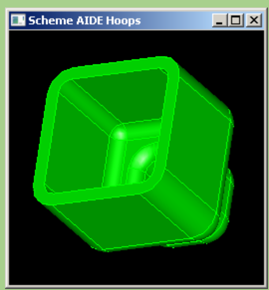
| | | |
|---|---|---|
| cellular representation scheme, the information related to wires is not used. The application-specific data can be linked to individual cells, providing information about materials, boundary conditions, etc. | cellular data structures throughout the modeling process. | |

### 3.6. Basic Modeling

| | ACIS | OCCT |
|---|---|---|
| Validation Tools<br>The ability to verify the topological and geometrical validity of the entire model is a fundamental feature that is especially critical for data exchange. Modelers are unlikely to work on shapes that are invalid, even if end-users (human or a caller code) may not identify these invalidities. | + | + |
| Extrusion of Planar Sketches<br>Generation of 3D models from 2D sketches by sweeping along straight line onto the given distance.<br> | Available. In addition, ACIS provides a dedicated API for the extrusion of basic planar faces that consist solely of straight-line segments and arcs. The latter allows for achieving better performance in the domains of Electronic Design Automation (EDA), Computer-Aided Engineering (CAE), and Architecture, Engineering, and Construction (AEC). | + |
| Bodies of Revolution<br>Generation of 3D models by rotation of a given profile around an axis. | + | + |

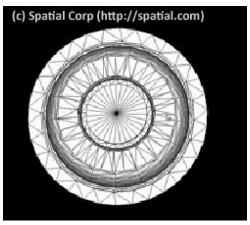
| | | |
|---|---|---|
|  | | |
| Boolean Operations<br>Cut, Fuse, Common, Intersect.<br> | Available. Non-manifold topology is permitted as an outcome. Surfaces can be in close proximity. ACIS incorporates the notion of "Fuzzy Booleans," an enhancement to the conventional Boolean Operations activated upon the input of a "fuzz" parameter. The fuzz parameter specifies a proximity range in which entities from separate bodies are considered to be overlapping. The value of this parameter must exceed the global resolution (SPAresabs). Otherwise, it is disregarded. | Available. Non-manifold topology is permitted as an outcome. Surfaces can be in close proximity. OpenCascade has introduced the concept of "Fuzzy Booleans" to provide the caller code with improved control over the tolerances used. Booleans rely on the local tolerances of the subshapes when the "fuzzy value" is not specified. The implementation of the "fuzzy" mode for Booleans in OpenCascade was influenced by the same capability in ACIS. |
| Chamfers<br>Cross-section is flat or sometimes conic. Normally chamfers can be represented by ruled surfaces.<br> | +<br> | + |
| Fillets<br>Cross-section is circular. Single- and variable-radius fillets are considered.<br> | Available. Generally, spline surface is used to represent a blend. Whenever possible, a simple analytic surface is used. | + |

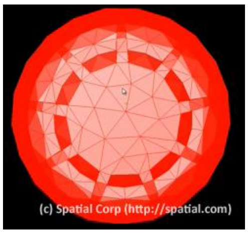
| | | |
|---|---|---|
| |   | |
| Skinning<br><br>Reconstruction of a surface from a sequence of curves.<br><br> | + | +<br><br>  |
| Lofting<br><br>The same as skinning but with possibility of direct control over the surface tangencies across the defining curves.<br><br> | Available. Supported are the following types of lofting: lofting between surfaces, lofting with Laws, lofting with guide curves, lofting to a point.<br><br>  | Available. Most types of lofting can be implemented either with GeomFill package or by means of lower-level interpolation and approximation tools (e.g., with AdvApprox package). |
| Net Surfaces<br><br>Construction of surfaces on a grid (mesh) of curves. Unlike skinning and lofting, where only one parameter (u or v) is defined by the guide curves, here both parameters are defined by a network of curves. Net surfaces allow better control on the corners of the constructed surface (especially important for accurate gap filling).<br><br> | + | Possible but not directly exposed with the API of the library.<br><br>  |

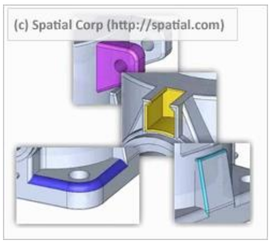
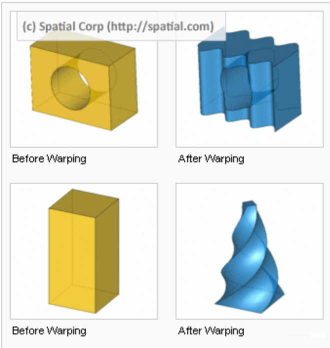
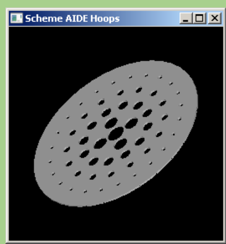
| | | |
|---|---|---|
| Gap Filling<br>Construction of a face (a surface patch) filling a hole represented by a circuit of edges (curves). | Available in Covering component.<br> | Available via Constrained Filling tool (PLATE). |
| Drafting<br>A practical use of drafting is for molded items.<br> | + | + |
| Shelling<br>Used for sheet thickening. Normally consists in removal of one or several faces with subsequent offsetting of the remaining faces. | +<br> | + |
| Mass-Inertia Properties<br>Mass, center of gravity, gyration radii, principal axes and moments of inertia etc. | + | + |
| History of Modifications<br>Tracking what is happening with topological primitives during the modeling session. | Sophisticated treatment by means of so called "History Streams" is implemented. Tracking of modification for each ENTITY is automated by ACIS. | Supported at the level of modeling API. The history of modifications can be mapped to OCAF (Open CASCADE Application Framework). However, tracking the chain of modifications for each topological entity should be done manually by a programmer (using BRepTools_History class). |
| Local Operations | + | + |

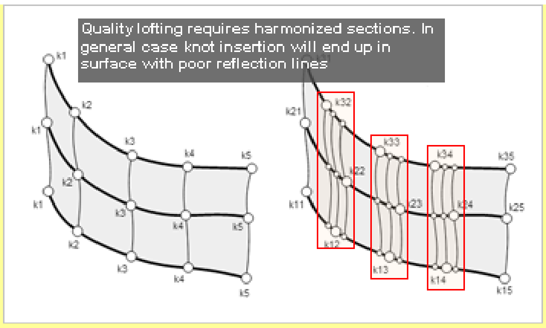
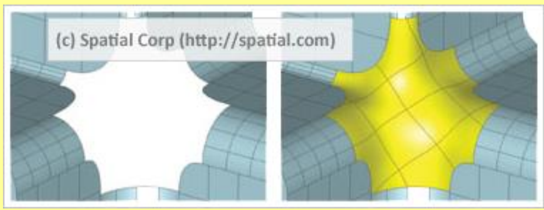
| | | |
|---|---|---|
| Local modification of faces in a prescribed way. Such functionality normally affects only a limited part of the entire topology. Examples: change a surface for a face, build local depression or protrusion etc. | | |

## 3.7. Advanced Tools

| | ACIS | OCCT |
|---|---|---|
| Model Simplification<br>Speed up collision detection by removal of inaccessible areas; prepare model for FEA solvers (defeaturing); compactify data for Web visualization, etc. Such removal capabilities are part of "direct modeling" philosophy. | Removal of small features: faces with the given area, protrusions and depressions of the given height etc. All topological constraints are kept satisfied (model does not fall apart). | Limited capabilities: removal of small edges (deletion or merge with neighbors), removal of isolated faces, face removal with reintersecting neighbor faces. |
| Geometry Simplification<br>Conversion from B-spline geometry to analytical form. | + | Removal of redundant spline knots/poles and similar primitive-level functionality. Conversion of splines to analytical geometry is available as a feature of Analysis Situs. |
| Facets<br>Coarse facets are normally used for visualization and in some other scopes for simplifying calculations (e.g., clearance analysis).<br> | The Faceter Component (FCT) is used to generate coarse view-independent meshes. The facet information is not stored in topological structures and SAT files. However, facets can be output through Mesh Managers to application-specific data structures. | B-rep Mesh component is used to produce facets. The resulting discrete model is bound to the corresponding topological entities (as a specific kind of geometry) and stored in OpenCascade native files. |
| Surface Meshes | The outcomes of coarse | No open-source |

| | | |
|---|---|---|
| High-quality surface meshes are commonly used for engineering analyses.   | faceting (FCT) can be transferred to VKI (Visual Kinematics) libraries for various types of high-quality mesh generation. The FCT results serve as inputs for the subsequent meshing process. Therefore, the B-rep model is not really utilized, and instead, ACIS communicates with VKI through an intermediate tessellation mesh. VKI enables the production of high-quality surface, tetrahedral, and hexahedral meshes.<br><br>Additional meshing techniques may have become accessible with the acquisition of Distene firm by DS. However, this functionality is not intrinsically integrated into ACIS. | faceters with controlled element size are available. |
| Parallel faceting<br><br>Parallel computations may significantly improve the efficiency of the faceting procedure. There are two sorts of parallelism that we are considering: single-body faceting and multi-body faceting. Single-body faceting involves the simultaneous processing of all faces. Multi-body faceting does not exhibit parallelism among faces, but rather among bodies. The latter form of parallelism can be more efficient for models that involve a large number of bodies. Typically, this type of parallel processing doesn't require extensive synchronizations between threads. In the case of single-body faceting, it is uncommon for edges to be exclusive to individual faces, as they are typically shared by multiple faces. | Two forms of parallelism that can be employed are face-based and body-based. | Parallelism is implemented at the level of faces and edges. |
| Collision Detection<br><br>This standard functionality allows checking clashes between several workpieces. | + | - |
| Feature Recognition<br><br>Recognize mechanical features of the given type. | Given a seed face and a feature type, returns all the faces that define that feature. ACIS can detect protrusions, depressions and fillet chains. | -<br><br>Feature recognition is available in Analysis Situs. |

| | | |
|---|---|---|
|  | | |
| Generalized Space Warping<br>Blending, stretching, twisting, non-uniform scaling.<br> | Available. General mathematical functions (ACIS textual Laws) are used to perform almost any kind of procedural morphing. | - |
| Patterns<br>Automated creation of repetitive mechanical features, such as arrangement of holes and grooves. Without a specific treatment, implementation of this feature normally requires a series of Boolean operations to be done. | Implemented with one Boolean operation taking advantage of dedicated pattern description. SAT files are compressed correspondingly.<br> | More efforts (C++ code) are necessary to reproduce these kinds of operations. One Boolean operation can be used as well. |
| Sculptured Surface Modeling<br>Manipulating B-spline control points allows changing shape of a surface, but does not enforce geometric constraints. Another popular technique of surface modeling is lofting: here constraints are satisfied, but no sculpting is possible. Moreover, lofting techniques very often end up with surfaces of bad quality. | Supported by ACIS Deformable Modeling (ADM) component. | PLATE-based interpolation only. Other "constrained filling" strategies provided in OpenCascade are not enough functional. |

| | | |
|---|---|---|
| <br><br>A different surface modeling technique consists in bending of a simple analytic surface (e.g., plane) until geometric constraints are satisfied. Energy-based optimization problem can be formulated for this procedure. The target surface will be the one yielding a minimum value on the energy criterion.<br><br> | | |

## 3.8. Data Exchange

This section specifically includes only the CAD formats that are used most often.

| | ACIS | OCCT |
|---|---|---|
| STEP (AP203/214) | + | + |
| IGES | + | + |
| ACIS SAT | Native | - |
| OCCT BRep | - | Native |
| Parasolid XT | + | - |
| DXF | Reader only | - |
| Healing Component<br>When a CAD model is converted from one geometric modeling system to another, there can be problems with its validity. One may encounter anomalies in the topological structure, tolerance values, and geometry definition, particularly when dealing with procedural geometries. Hence, to guarantee the accuracy of the translated model in terms of its topology and geometry, extra efforts might be required. Post-processors designed to address known interoperability problems are commonly referred to as "Healers". | + | + |

### 3.9. Application Infrastructure

| | ACIS | OCCT |
|---|---|---|
| Undo/Redo, Save/Restore, Copy/Paste<br><br>Standard functionality in geometric modeling applications. | Implemented within the ENTITY base class, which serves as the foundation for most geometrical and topological data structures. The client code must inherit its classes from ENTITY in order to utilize the Undo/Redo, Save/Restore, and Copying infrastructure. This technique is quite invasive. Additionally, the Save/Restore feature is constrained with the native SAT and SAB file formats, making it hard to build a custom format.<br><br>Basically, the mentioned functionality is essential part of the modeler. | Separated from the topological model. Implemented within the OCAF module (Open CASCADE Application Framework).<br><br>This functionality is not a part of the modeler. Hence, there is no issue with utilizing traditional Undo/Redo, Save/Restore, and Copy/Paste functionalities in applications that do not require much of the OpenCascade's modeling capabilities. |
| Binding Application-specific Data to geometry<br><br>In a CAD model, our interest often extends beyond the geometric shape. It is a common practice in engineering applications to associate certain attributes with distinct components of a model. These can include colors, materials, names, boundary conditions, etc. | Attributes can be used to associate application-specific data with the elements of a CAD model. Attributes are linked to various types of topological and geometric primitives, which are derived from the ENTITY base class. An alternative is to utilize Cellular Topology in order to associate attributes with certain spatial cells. Directly associating attributes with the topological entities facilitates the persistent storage of the corresponding data during the modeling process. | The topological model is lightweight and exclusively consists of boundary representation. It is recommended to connect all properties specific to an application using auxiliary data structures, such as OCAF (Open CASCADE Application Framework). |
| Application Framework<br><br>Components for jump-starting development of modeler-based applications. Can include a ready-to-use viewer, facilities for organization of data into hierarchical structures, MVC architecture etc. | RADF (Rapid Application Development Framework) is completely separated from ACIS. RADF is fundamentally a C# application. | OCAF (Open CASCADE Application Framework) is shipped with OCCT. |

## 4. CONCLUSIONS

It is important to acknowledge that the modeling capabilities of ACIS are more extensive than those of OpenCascade. For example, OCCT does not support generalized textual Laws and procedural geometry. In addition, there is a deficiency of widely-used modeling capabilities related to the MCAD (Mechanical CAD) domain, such as feature recognition and direct modeling. The reason for this is that [after Euclid Quantum] OpenCascade is not used as the geometric kernel for any industrially proven MCAD-system. Consequently, the specific need for advanced modeling capabilities is typically less for OCCT compared to ACIS, CGM, or Parasolid. However, OCCT may be enhanced with various modeling capabilities using its existing mathematical foundation. The latter kind of extension is our focus in Analysis Situs open-sourced project.

ACIS appears to have the following disadvantages in comparison to OCCT:

1. The absence of general infrastructure components, such as model-independent Undo/Redo or SAT-independent Save/Restore services, can make the creation of engineering applications that do not prioritize geometric modeling more challenging.

2. The library is fragmented to a greater extent, with data interchange, visualization components, and application development tools being distributed as separate [paid] products.

3. It is neither free nor open source.

When comparing OpenCascade to ACIS, it is important to note the following main drawbacks of OpenCascade:

1. Limited availability of direct modeling, feature recognition and advanced surface modeling techniques.

2. The robustness of OpenCascade leaves much better to be desired.

3. OpenCascade does not provide an alternative to the cellular topology of ACIS and remains limited with traditional B-rep (there is no API related to voxelization, mesh-based modeling, cellular structures, etc.).

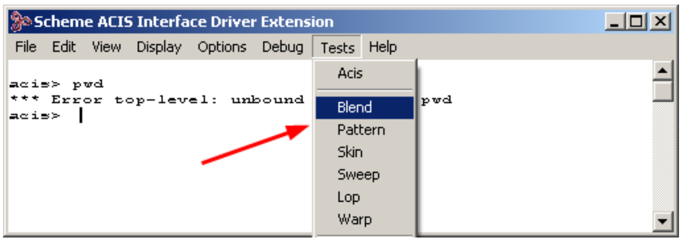
Despite our lack of practical experience with ACIS, we can nevertheless determine that both libraries are somewhat similar. It is important to emphasize that OCCT is an open-source library, which means its potential for expansion is practically unbounded.

# 5. Appendices

## 5.1. Appendix 1: interesting features of ACIS

Along with the functional differences outlined in this paper, ACIS additionally includes certain convenient features that could be beneficial in OCCT.

- User interface for launching functional tests. Using Scheme AIDE, running modeling tests and discovering their results in the corresponding 3D viewer is a straightforward process. The tests are categorized into meaningful classes, such as "Blend" and "Pattern". Consequently, newcomers are not overwhelmed by the enormous amount of documentation required to begin using the library.




- History Stream seems to implement a well-designed principle for tracking of modifications throughout the modeling session.

- Journaling is a maintenance feature that automatically generates Scheme AIDE files when an API of ACIS is invoked. Subsequently, these files might be utilized for the purpose of issue reporting and troubleshooting. Below is an example of code that has been enhanced with journaling:

```
AcisOptions* acis_options = ACIS_NEW AcisOptions();
Aapi_set_journal_name(acis_options, "JOURNAL_3D");

Aapi_start_journal(acis_options);

...
// Run modeling API
api_unite(e1,
          path,
          acis_options);

// Disable journaling
api_pause_journal(acis_options);
api_unite(e2,
          path,
          acis_options);

// Enable journaling back
api_resume_journal(acis_options);
api_unite(e3,
          path,
          acis_options);
...

api_end_journal(acis_options);
ACIS_DELETE acis_options;
```

ACIS journaling generates both Scheme AIDE and SAT files. The Scheme script imports the SAT files and performs the target function calls. The journaling technique facilitates the isolation of detected issues in C++ programs.

It is important to mention that ACIS offers a restricted range of journaling capabilities. To determine whether journaling is supported for a specific operation, it is necessary to consult the API documentation.

## 5.2. Appendix 2: evaluation of ACIS Modeler on Windows

Spatial Corp provides evaluation period for their products. In order to try out ACIS Modeler one may follow the small instruction below.

1. Fill in evaluation form on the official site.

2. Using the provided customer ID and password (it may require additional negotiations with Spatial Corp), sign-in to the Download Center and choose the product for evaluation (3D ACIS Modeler).

3. Download the product binaries for your target platform.

4. Follow the online wizard to obtain evaluation product key. This key will be offered as a C++ function:

```
void unlock_spatial_products_6401()
{
  const char* unlock_str =
    "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

  spa_unlock_result out = spa_unlock_products( unlock_str );

  if ( out.get_state() != SPA_UNLOCK_PASS )
  { /* insert your error handling code here */; }
}
```

5. Unzip the downloaded archive with ACIS Modeler and run setup.exe. Choose installation directory, e.g.:

```
%A3DT% = D:\DevTools\Spatial\ACIS_R24
```

6. Check the directory containing Scheme AIDE sources:

```
%A3DT%\scm\acis3dt
```

   In order to get started with ACIS we will use Scheme AIDE interpreter (similar to Draw in OCCT). However, it is first necessary to build Scheme AIDE from sources. In 'acis3dt' directory create a batch file (msvc.bat) with the following contents:

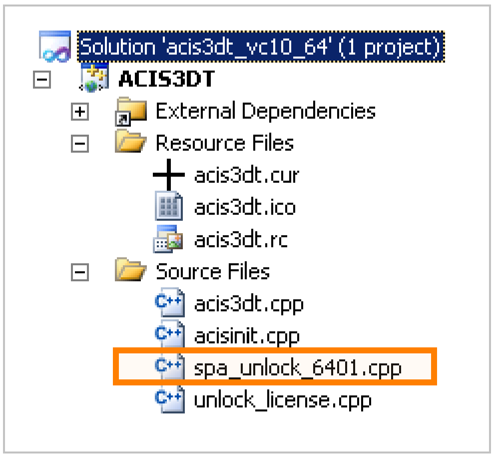
```
@echo off

set "A3DT=D:\DevTools\Spatial\ACIS_R24"
set "PATH=%A3DT%\NT_VC10_64_DLL\code\bin;%PATH%"

start "%VS100COMNTOOLS%\..\IDE\devenv.exe" acis3dt_vc10_64.sln
```

   Notice that 'NT_VC10_64_DLL' is a directory with ACIS libraries built in Release mode. There should be normally another directory with Debug build.

7. Run 'msvc.bat' and rebuild project in Visual Studio. You will get a license error. In order to fix it, include the downloaded C++ file with license function into the project.



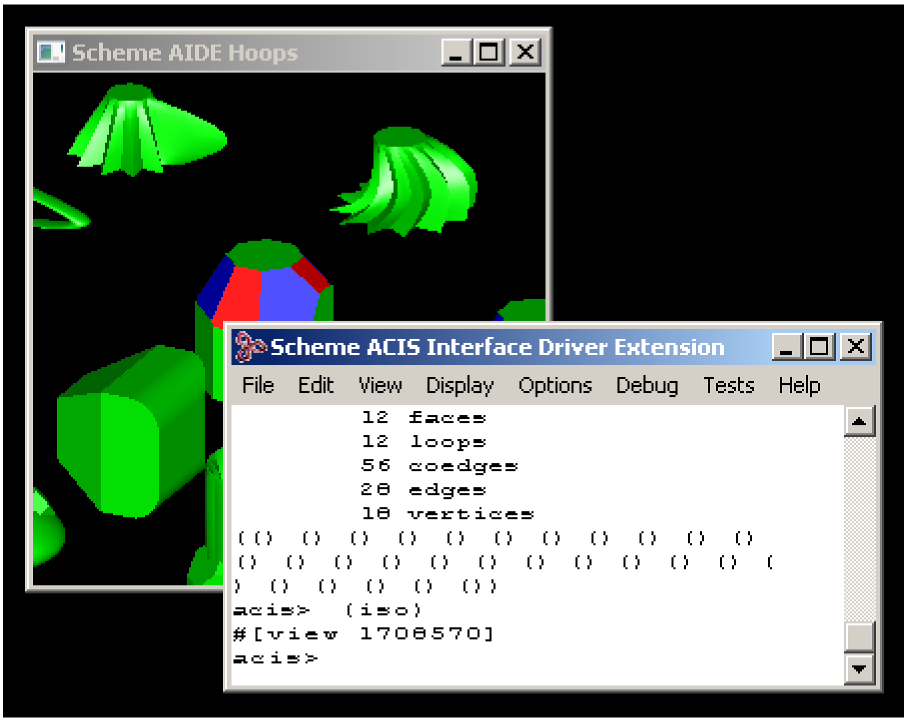

   Then in 'unlock_license.cpp' invoke the provided license function:

```
#ifdef _MSC_VER

extern void unlock_spatial_products_6401();

GLOBAL_PROC void unlock_license()
{
  unlock_spatial_products_6401();
}
...
```

8. Build and run.




## 6. REFERENCES

1. http://www.spatial.com/products/3d-acis-modeling
2. https://www.3dcadworld.com/spatial-acis-cgm-and-the-future-of-geometric-modeling-kernels/