STL File Format

Rapid Prototyping Systems

- **Conversion to STL Format:**
  - To establish consistency, the STL (stereolithography, the first RP technique) format has been adopted as the standard of the rapid prototyping industry.
  - The second step, therefore, is to convert the CAD file into STL format. This format represents a three-dimensional surface as an assembly of planar triangles
  - STL files use planar elements, they cannot represent curved surfaces exactly. Increasing the number of triangles improves the approximation

Rapid Prototyping Systems

- **Slice the STL File:**
  - In the third step, a pre-processing program prepares the STL file to be built.
  - The pre-processing software slices the STL model into a number of layers from 0.01 mm to 0.7 mm thick, depending on the build technique.
  - The program may also generate an auxiliary structure to support the model during the build. Supports are useful for delicate features such as overhangs, internal cavities, and thin-walled sections.

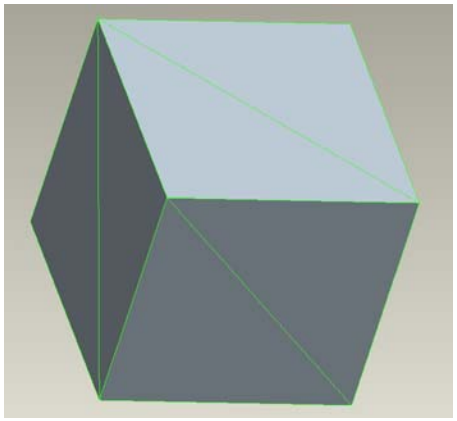
STL Format: B-rep, solid object

- An STL file is saved with the extension “.stl," case-insensitive.
- STL is a triangular facet based representation that approximates surface and solid entities only. Entities such as points, lines, curves, and attributes such as layer and color will be ignored during the output process
- An STL file consists of a list of facet data.
- Each facet is uniquely identified by a unit normal (a line perpendicular to the triangle and with a length of 1.0) and by three vertices (corners).
- The normal and each vertex are specified by three coordinates each, so there is a total of 12 numbers stored for each facet.

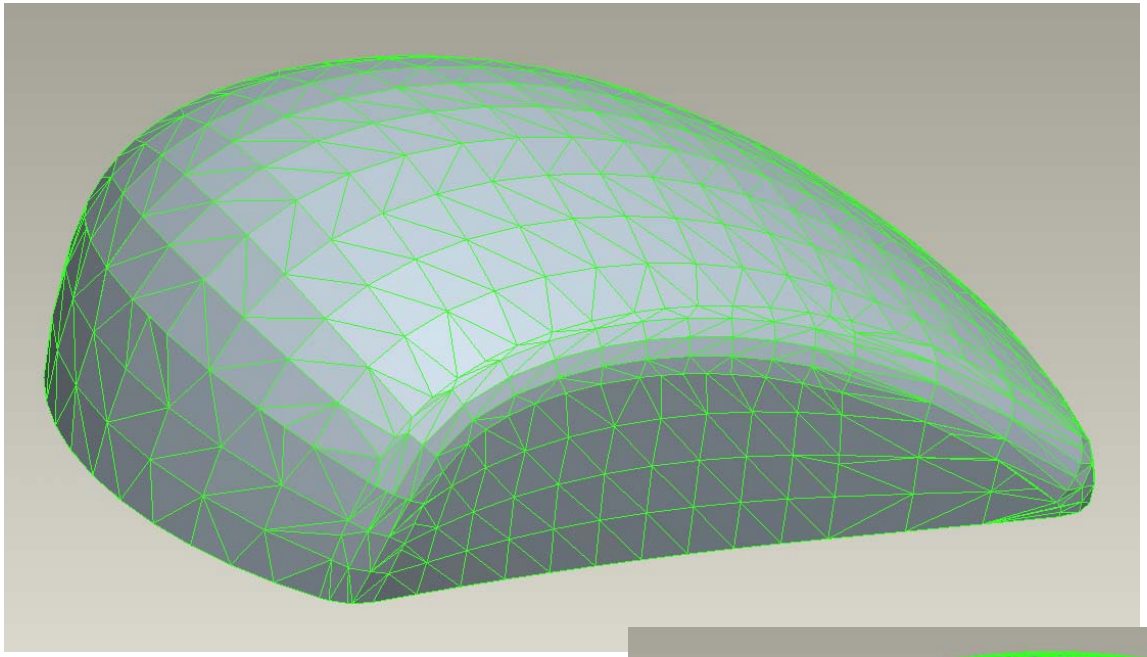
ASCII STL File Format

```
**********************************************************
solid PRT0002
  facet normal 0.000000e+000 0.000000e+000 -1.000000e+000
    outer loop
      vertex 1.000000e+001 7.500000e+000 0.000000e+000
      vertex -1.000000e+001 -7.500000e+000 0.000000e+000
      vertex -1.000000e+001 7.500000e+000 0.000000e+000
    endloop
  endfacet
  ……
  ……
  facet normal -1.000000e+000 0.000000e+000 0.000000e+000
    outer loop
      vertex -1.000000e+001 7.500000e+000 3.000000e+000
      vertex -1.000000e+001 7.500000e+000 0.000000e+000
      vertex -1.000000e+001 -7.500000e+000 0.000000e+000
    endloop
  endfacet
endsolid PRT0002
**********************************************************
```

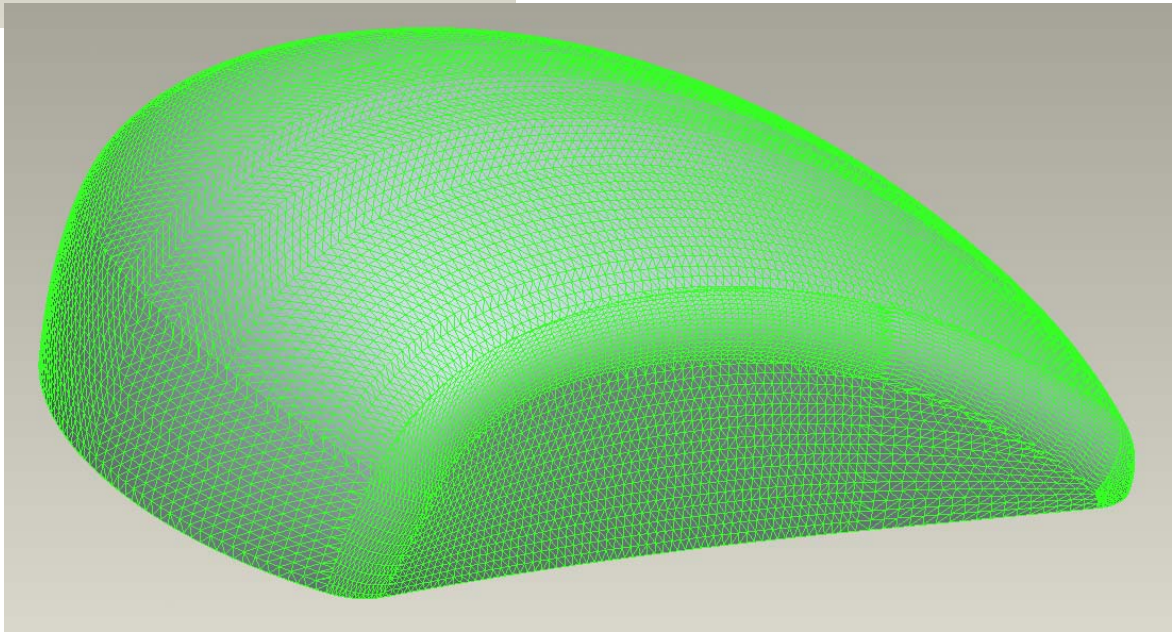
```
BLOCK
 facet normal 0.000000e+00 -1.000000e+00 0.000000e+00
   outer loop
     vertex 0.000000e+00 0.000000e+00 -1.000000e+00
     vertex 1.000000e+00 0.000000e+00 0.000000e+00
     vertex 0.000000e+00 0.000000e+00 0.000000e+00
   endloop
 endfacet
 facet normal 0.000000e+00 0.000000e+00 1.000000e+00
   outer loop
     vertex 1.000000e+00 1.000000e+00 0.000000e+00
     vertex 0.000000e+00 0.000000e+00 0.000000e+00
     vertex 1.000000e+00 0.000000e+00 0.000000e+00
   endloop
 endfacet
 facet normal -1.000000e+00 0.000000e+00 0.000000e+00
   outer loop
     vertex 0.000000e+00 1.000000e+00 0.000000e+00
     vertex 0.000000e+00 0.000000e+00 -1.000000e+00
     vertex 0.000000e+00 0.000000e+00 0.000000e+00
   endloop
 endfacet
 facet normal 0.000000e+00 0.000000e+00 1.000000e+00
   outer loop
     vertex 1.000000e+00 1.000000e+00 0.000000e+00
     vertex 0.000000e+00 1.000000e+00 0.000000e+00
     vertex 0.000000e+00 0.000000e+00 0.000000e+00
   endloop
 endfacet
 facet normal 0.000000e+00 -1.000000e+00 0.000000e+00
   outer loop
     vertex 0.000000e+00 0.000000e+00 -1.000000e+00
     vertex 1.000000e+00 0.000000e+00 -1.000000e+00
     vertex 1.000000e+00 0.000000e+00 0.000000e+00
   endloop
 endfacet
 facet normal 1.000000e+00 0.000000e+00 0.000000e+00
   outer loop
     vertex 1.000000e+00 1.000000e+00 -1.000000e+00
     vertex 1.000000e+00 0.000000e+00 0.000000e+00
     vertex 1.000000e+00 0.000000e+00 -1.000000e+00
   endloop
 endfacet
 facet normal 1.000000e+00 0.000000e+00 0.000000e+00
   outer loop
     vertex 1.000000e+00 1.000000e+00 0.000000e+00
     vertex 1.000000e+00 0.000000e+00 0.000000e+00
     vertex 1.000000e+00 1.000000e+00 -1.000000e+00
   endloop
 endfacet
 facet normal 0.000000e+00 0.000000e+00 -1.000000e+00
   outer loop
     vertex 0.000000e+00 1.000000e+00 -1.000000e+00
     vertex 1.000000e+00 0.000000e+00 -1.000000e+00
     vertex 0.000000e+00 0.000000e+00 -1.000000e+00
   endloop
 endfacet
 facet normal 0.000000e+00 0.000000e+00 -1.000000e+00
   outer loop
     vertex 1.000000e+00 1.000000e+00 -1.000000e+00
     vertex 1.000000e+00 0.000000e+00 -1.000000e+00
     vertex 0.000000e+00 1.000000e+00 -1.000000e+00
   endloop
 endfacet
 facet normal -1.000000e+00 0.000000e+00 0.000000e+00
   outer loop
     vertex 0.000000e+00 1.000000e+00 -1.000000e+00
     vertex 0.000000e+00 0.000000e+00 -1.000000e+00
     vertex 0.000000e+00 1.000000e+00 0.000000e+00
   endloop
 endfacet
 facet normal 0.000000e+00 1.000000e+00 0.000000e+00
   outer loop
     vertex 1.000000e+00 1.000000e+00 -1.000000e+00
     vertex 0.000000e+00 1.000000e+00 -1.000000e+00
     vertex 0.000000e+00 1.000000e+00 0.000000e+00
   endloop
 endfacet
 facet normal 0.000000e+00 1.000000e+00 0.000000e+00
   outer loop
     vertex 1.000000e+00 1.000000e+00 0.000000e+00
     vertex 1.000000e+00 1.000000e+00 -1.000000e+00
     vertex 0.000000e+00 1.000000e+00 0.000000e+00
   endloop
 endfacet
endsolid BLOCK
```



An Example STL File – Block.stl



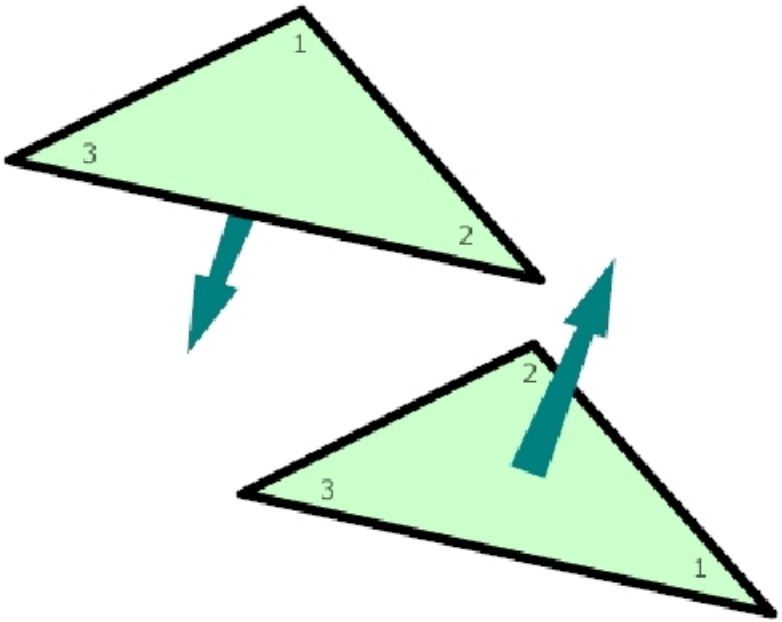
The density of triangle facets change according to the geometry



And it Changes with Chord Height affecting final surface resolution

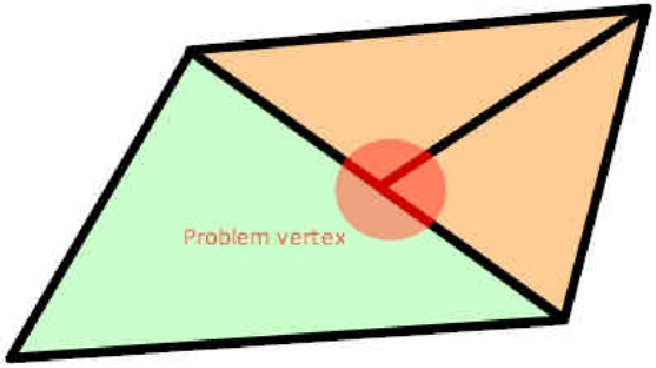
STL Format: B-rep, solid object

- There are two storage formats available for STL files, which are Ascii and Binary. ASCII file is human-readable but larger than binary
- Each facet is part of the boundary between the interior and the exterior of the object. The orientation of the facets (which way is "out" and which way is "in") is specified redundantly
- First, the direction of the normal is outward. Second, which is most commonly used now-a-day, list the facet vertexes in counter-clockwise order when looking at the object from the outside (right-hand rule).



Vertex to Vertex Rule

Each triangle must share two vertices with each of its adjacent triangles. In other words, a vertex of one triangle cannot lie on the side of another.

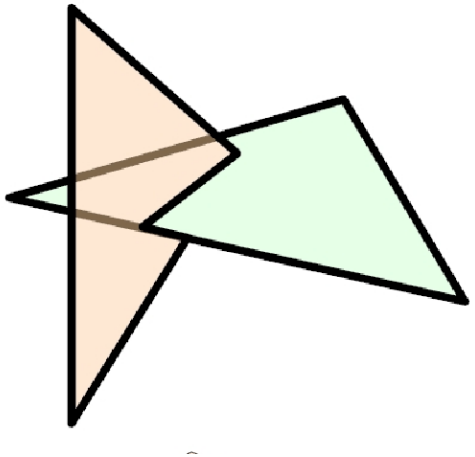


A violation of the vertex-to-vertex rule.

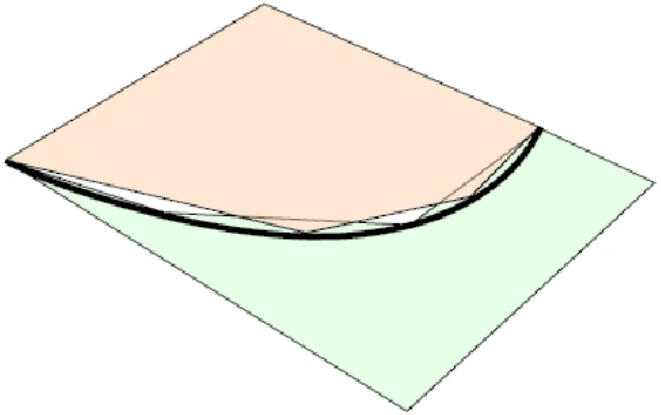
- to check:
  1. F must be even
  2. E must be a multiple of three
  3. 2*E must equal 3*F

F, E, V, and B are the number of faces, edges, vertices, and separate solid bodies.

“Leaky” Models



Two facets crossed in the 3D space.



The triangulated edge of two surface patch do not match, thus produce gaps between faces.

All facets in a STL data file should construct one or more non-manifold entities according to Euler's Rule for legal solids:

$$F-E+V=2B$$

F, E, V, and B are the number of faces, edges, vertices, and separate solid bodies.
If the relation does not hold, we say that the STL model is 'leaky'. When a 'leaky' STL file is processed by slicing algorithms, the result produces slice boundaries that are not fully closed.

Errors in STL file format:

1. Gap or Missing of Facets
2. Degenerate Facets
3. Overlapping Facets
4. Non Manifold Topology conditions (Points, Edges and Faces)

STL File- ERRORS – Missing Facets

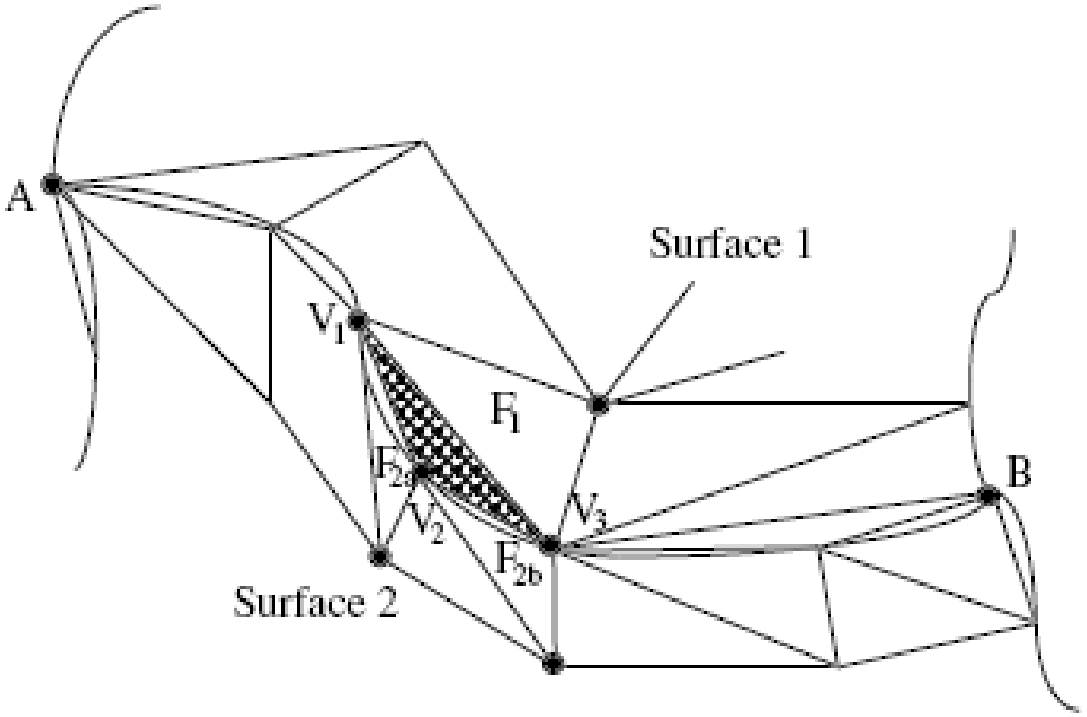


Figure 6.3: Gaps due to missing facets [4]

Shell Puncture

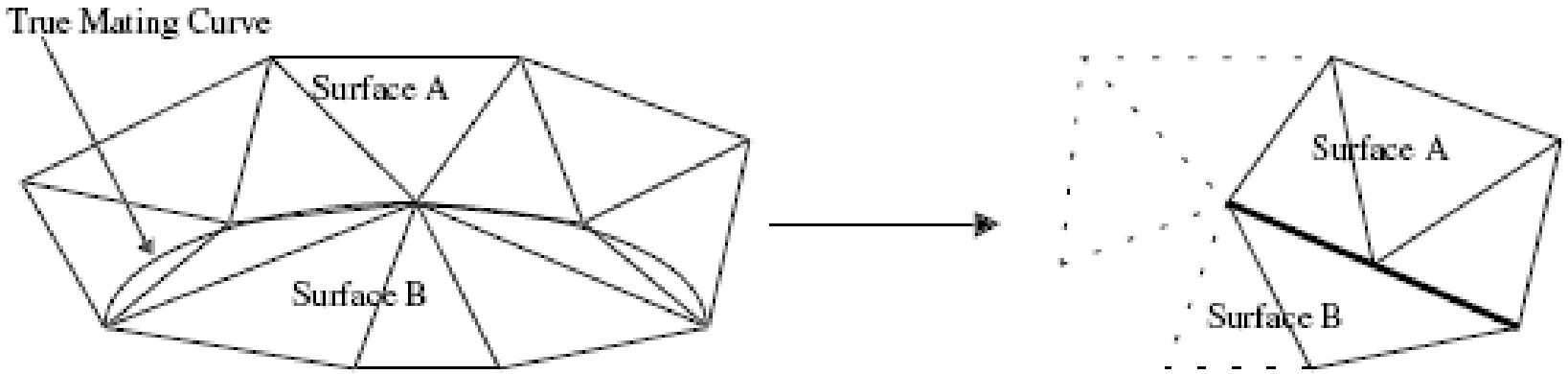


Figure 6.4(a): Shell punctures created by unequal tessellation of two adjacent surface patches along their common mating curve

Figure 6.4(b): Shell punctures eliminated at the expense of adding degenerate facet

Overlapping , Non Manifold

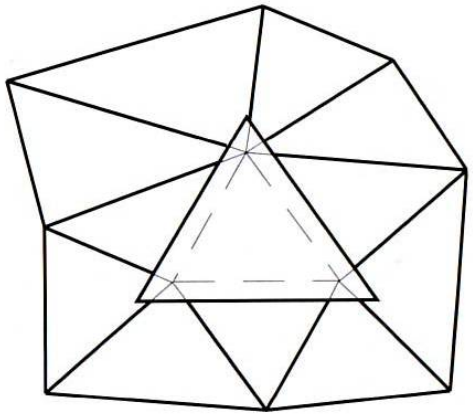


Figure 6.5: Overlapping facets

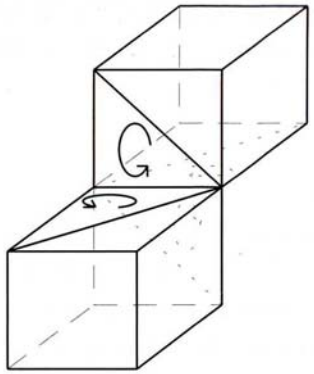


Figure 6.6(a): A non-manifold edge whereby two imaginary minute cubes share a common edge

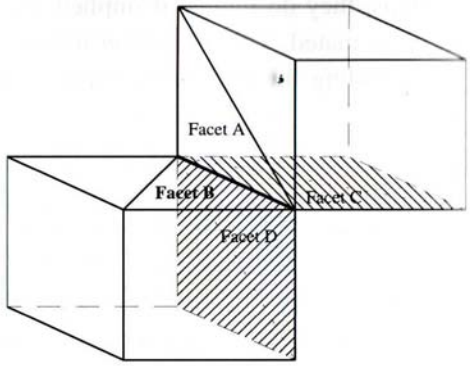


Figure 6.6(b): A non-manifold edge whereby four facets share a common edge after tessellation

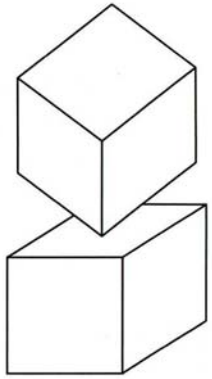


Figure 6.6(c): Non-manifold point

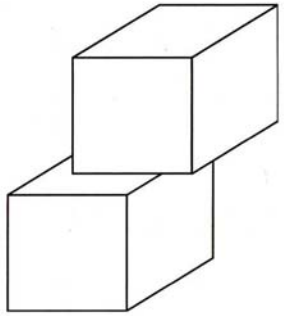


Figure 6.6(d): Non-manifold face

Valid Model

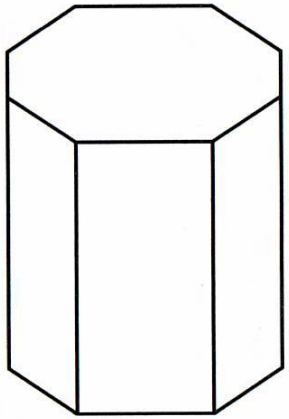


Figure 6.7(a): A valid 3D model

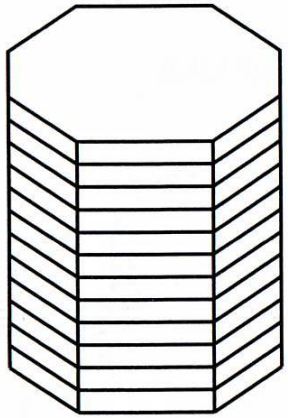


Figure 6.7(b): A 3D model sliced into 2D planar layers

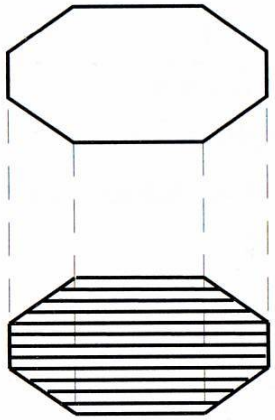


Figure 6.7(c): Conversion of 2D layers into 1D scan lines

Invalid model

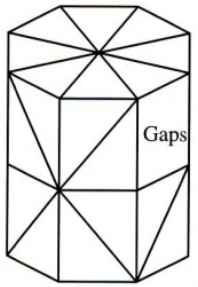


Figure 6.8(a): An invalid tessellated model

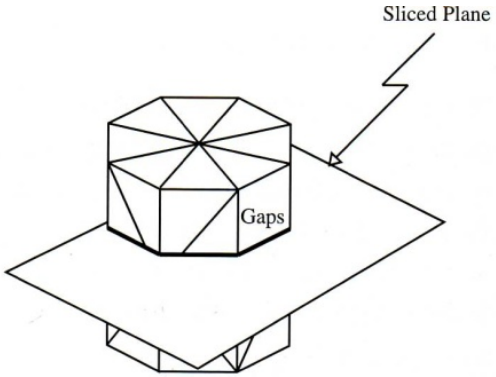


Figure 6.8(b): An invalid model being sliced

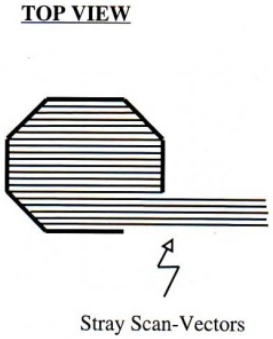


Figure 6.8(c): A layer of an invalid model being scanned

STL File Repair - Methods

- Generic Solution – Manual repairer of shell puncture / missing facets and “patch up”
- Special Algorithms- used for
  - » Two or more gaps formed a coincidental vertex
  - » Degenerate Facets
  - » Overlapping facets

  Generic Solution

Solving Missing Facets

- ➢ Checking approve edges & Facets
- ➢ Gap Detection
- ➢ Sorting loops

Step 1. Detection of Missing Facets

(1) Step 1: Checking for Approved Edges with Adjacent Facets

The checking routine executes as follows for Facet A as seen in Figure 6.9:

(a) (i) Read in first edge {vertex 1-2} from the STL file.
(ii) Search file for a similar edge in the opposite direction {vertex 2-1}.
(iii) If edge exists, store this under a temporary file (e.g., file B) for approved edges.
(iv) Do the same for 2 and 3 below.

(b) (i) Read in second edge {vertex 2-3} from the STL file.
(ii) Search file for a similar edge in the opposite direction {vertex 3-2}.
(iii) Perform as in (a)(iii) above.

(c) (i) Read in third {vertex 3-1} from the STL file.
(ii) Search file for a similar edge in the opposite direction {vertex 1-3}.
(iii) Perform as in (a)(iii) above.

This process is repeated for the next facet until all the facets have been searched.

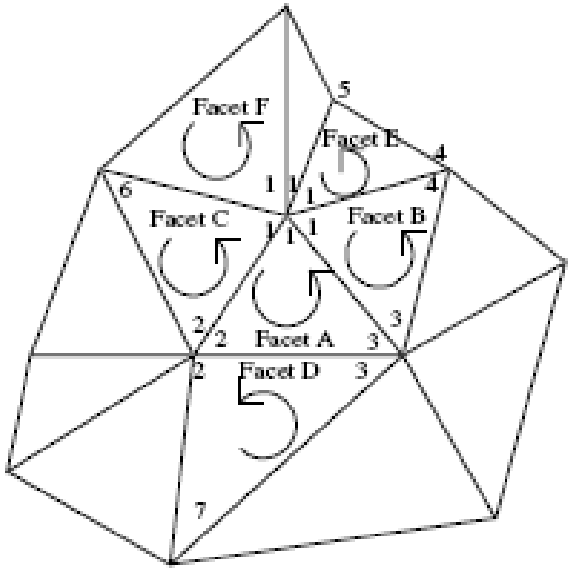


Figure 6.9: A representation of a portion of a tessellated surface without any gaps

Step -2. Detection of Gap

For Facet A (please refer to Figure 6.10):

(a) (i) Read in edge {vertex 2-3} from the STL file.
(ii) Search file for a similar edge in the opposite direction {vertex 3-2}.
(iii) If edge does not exist, store edge {vertex 3-2} in another temporary file (e.g., file C) for suspected gap's bounding edges and store vertex 2-3 in file B1 for existing edges without adjacent facets (this would be used later for checking the generated facet orientation).

For Facet B,

(b) (i) Read in edge {vertex 5-2} from the STL file.
(ii) Search file for a similar edge in the opposite direction {vertex 2-5}.
(iii) If it does not exist, perform as in (a)(iii) above.

(c) (i) Repeat for edges: 5-2 ; 7-5 ; 9-7 ; 11-9 ; 3-11.
(ii) Search for edges: 2-5 ; 5-7 ; 7-9 ; 9-11 ; 11-3.
(iii) Store all the edges in that temporary file B1 for edges without any adjacent facet and store all the suspected bounding edges of the gap in temporary file C. File B1 can appear as in Table 6.1.

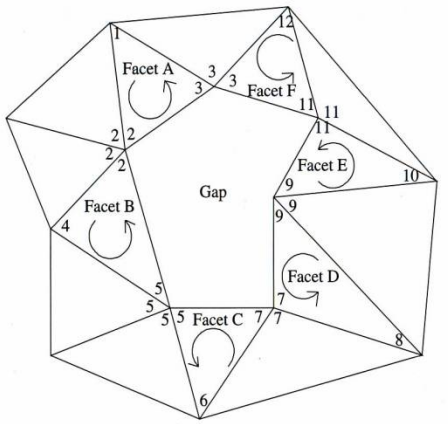


Figure 6.10: A representation of a portion of a tessellated surface with a gap present

Table 6.1: File B1 contains existing edges without adjacent facets

Vertex	Edge					
	First	Second	Third	Fourth	Fifth	Sixth
First	2	7	3	5	9	11
Second	3	5	11	2	7	9

Step-3. Detection of Erroneous (Incorrect) Edges

Table 6.2: File C containing all the "Erroneous" edges that would form the boundary of each gaps

Vertex	Edge								
	First	Second	Third	Fourth	Fifth	Sixth	Seventh	Eighth	Ninth
First	3	5	*	11	2	7	*	9	*
Second	2	7	*	3	5	9	*	11	*

*Represent all the other edges that would form the boundaries of other gaps

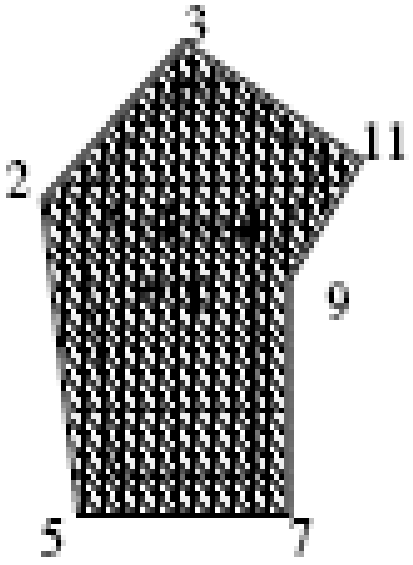


Table 6.3: File D containing sorted edges

	First edge	Second edge	Third edge	Fourth edge	Fifth edge	Sixth edge
First vertex	3	2	5	7	9	11
Second vertex	2	5	7	9	11	3

Step -4 Generation of Facets for the Repair of Gaps (1/2)

Table 6.4: Process of facet generation

		V3	V2	V5	V7	V9	V11
Generation of facets	F1	1	2	—	—	—	3
	F2	E	1	—	—	2	3
	F3	E	1	2	—	3	E
	F4	E	E	1	2	3	E

V = vertex, F = facet, E = eliminated from the process of facet generation

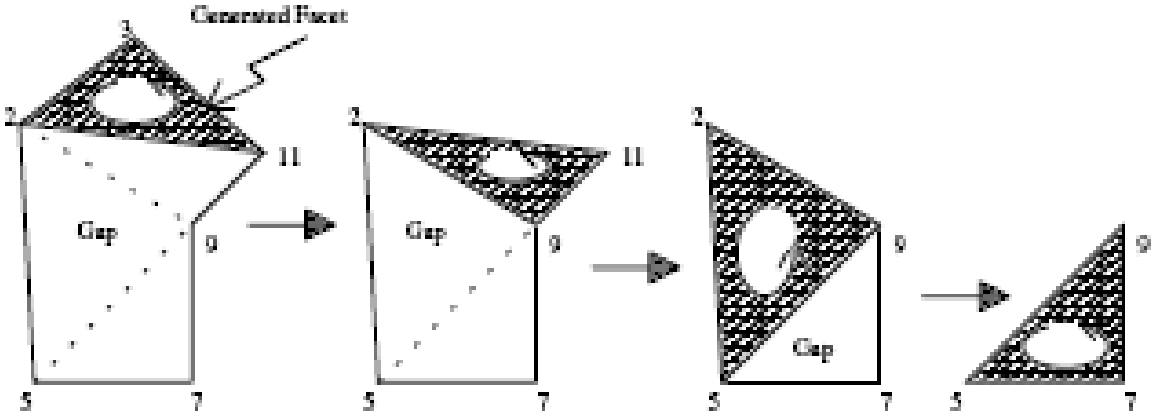


Figure 6.12(a): First facet generated

Figure 6.12(b): Second facet generated

Figure 6.12(c): Third facet generated

Figure 6.12(d): Fourth facet generated

Generation of Facets for the Repair of Gaps

(a) Generating the first facet: First two vertices (V3 and V2) in the first two edges of file D will be connected to the first vertex in the last edge (V11) in file D and the facet is stored in a temporary file E (see Table 6.5 on how the first generated facet would be stored in file E). The facet is then checked for its orientation using the information stored in file B1. Once its orientation is determined to be correct, the first vertex (V3) from file D will be temporarily removed.

(b) Generating the second facet: Of the remaining vertices in file D, the previous second vertex (V2) will become the first edge of file D. The second facet is formed by connecting the first vertex (V2) of the first edge with that of the last two vertices in file D (V9, V11), and the facet is stored in temporary file E. It is then checked to confirm if its orientation is correct. Once it is determined to be correct, the vertex (V11) of the last edge in file D is then removed temporarily.

(c) Generating the third facet: The whole process is repeated as it was done in the generation of facets 1 and 2. The first vertex of the first two edges (V2, V5) is connected to the first vertex of the last edge (V9) and the facet is stored in temporary file E. Once its orientation is confirmed, the first vertex of the first edge (V2) will be removed from file D temporarily.

(d) Generating the fourth facet: The first vertex in the first edge will then be connected to the first vertices of the last two edges to form the fourth facet and it will again be stored in the temporary file E. Once the number of edges in file D is less than three, the process of facet generation will be terminated. After the last facet is generated, the data in file E will be written to file A and its content (file E's) will be subsequently deleted. Table 6.5 shows how file E may appear.

Table 6.5: Illustration of how data could be stored in File E

Generated facet	First edge		Second edge		Third edge	
	First vertex	Second vertex	First vertex	Second vertex	First vertex	Second vertex
First	V3	V2	V2	V5	V5	V3
Second	V2	V9	V9	V11	V11	V2
Third	V2	V5	V5	V9	V9	V2
Fourth	V5	V7	V7	V9	V9	V5

Example – Checking of Algorithms for ODD/Even number Edges

Table 6.6: Process of facet generation for gaps with even number of edges

Facets	Vertices									
	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
F1	1	2								3
F2	E	1							2	3
F3	E	1	2						3	E
F4	E	E	1					2	3	E
F5	E	E	1	2				3	E	E
F6	E	E	E	1			2	3	E	E
F7	E	E	E	1	2		3	E	E	E
F8	E	E	E	E	1	2	3	E	E	E

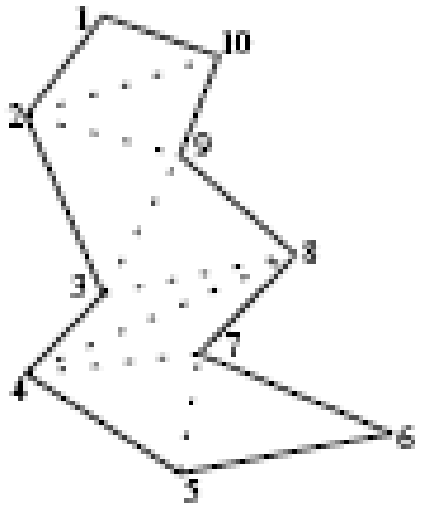


Figure 6.13: Gaps with even number of

With reference to Table 6.6,

First facet generated:
Edge 1 → V1, V2
Edge 2 → V2, V10
Edge 3 → V10, V1

Second facet generated:
Edge 1 → V2, V9
Edge 2 → V9, V10
Edge 3 → V10, V1

Odd Number Edges

Table 6.7: Process of facet generation for gaps with odd number of edges

Facets	Vertices										
	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11
F1	1	2									3
F2	E	1								2	3
F3	E	1	2							3	E
F4	E	E	1						2	3	E
F5	E	E	1	2					3	E	E
F6	E	E	E	1				2	3	E	E
F7	E	E	E	1	2			3	E	E	E
F8	E	E	E	E	1		2	3	E	E	E
F9	E	E	E	E	1	2	3	E	E	E	E

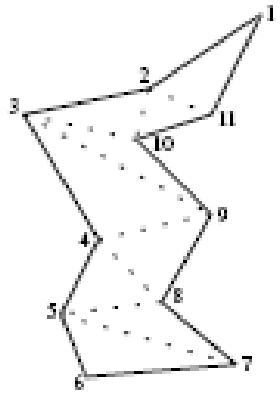


Figure 6.14: Gaps with odd number of edges

F1 → First and second vertices are combined with the last vertex. Once completed, eliminate first vertex. The remainder is ten vertices.

F2 → First vertex is combined with the last two vertices. Once completed, eliminate the last vertex. The remainder is nine vertices.

F3 → First and second vertices are combined with the last vertex. Once completed, eliminate first vertex. The remainder is eight vertices.

F4 → First vertex is combined with last two vertices. Once completed, eliminate the last vertex. The remainder is seven vertices.

This process is continued until all the gaps are patched.

Solving of wrong orientation of Facets

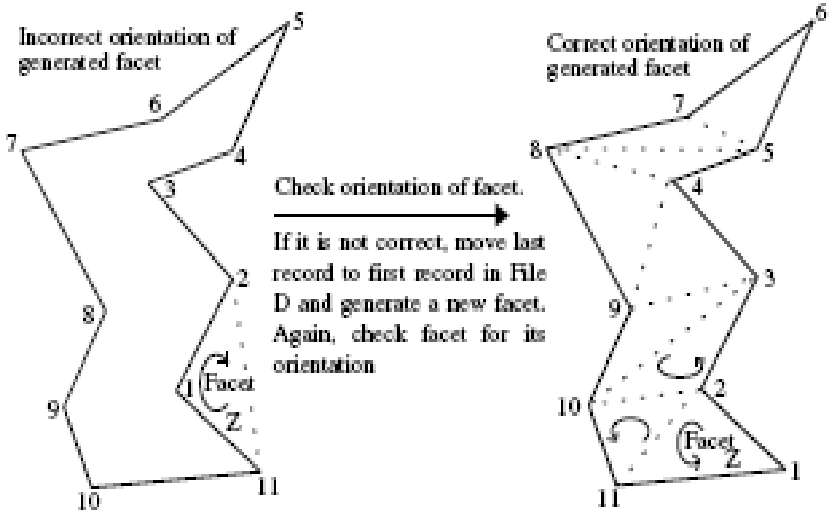


Figure 6.15: Incorrectly generated facet's orientation and its repair

Table 6.8: Illustration showing how file D is manipulated to solve orientation problems

Vertex	Edge										
	First	Second	Third	Fourth	Fifth	Sixth	Seventh	Eighth	Ninth	Tenth	Eleventh
First	1	2	3	4	5	6	7	8	9	10	11
Second	2	3	4	5	6	7	8	9	10	11	1

Table 6.9: Illustration showing the result of the shift to correct the facet orientation

Vertex	Edge										
	First	Second	Third	Fourth	Fifth	Sixth	Seventh	Eighth	Ninth	Tenth	Eleventh
First	11	1	2	3	4	5	6	7	8	9	10
Second	1	2	3	4	5	6	7	8	9	10	11

Other translators

1.HP/GL File

- It is standard data format for graphic plotters.
- Data types are two dimensional, including lines, circles, splines , texts etc.
- From a designer point of view, to automate a slicing routine which generates a section slice, invoke the plotter routine to produce a plotter output file and the loop back to repeat the process.

Advantages:

- Lot of commercial CAD systems have the interface to output the HPGL format.

- Slicing not required since it is a 2D geometry data format

Disadvantages:

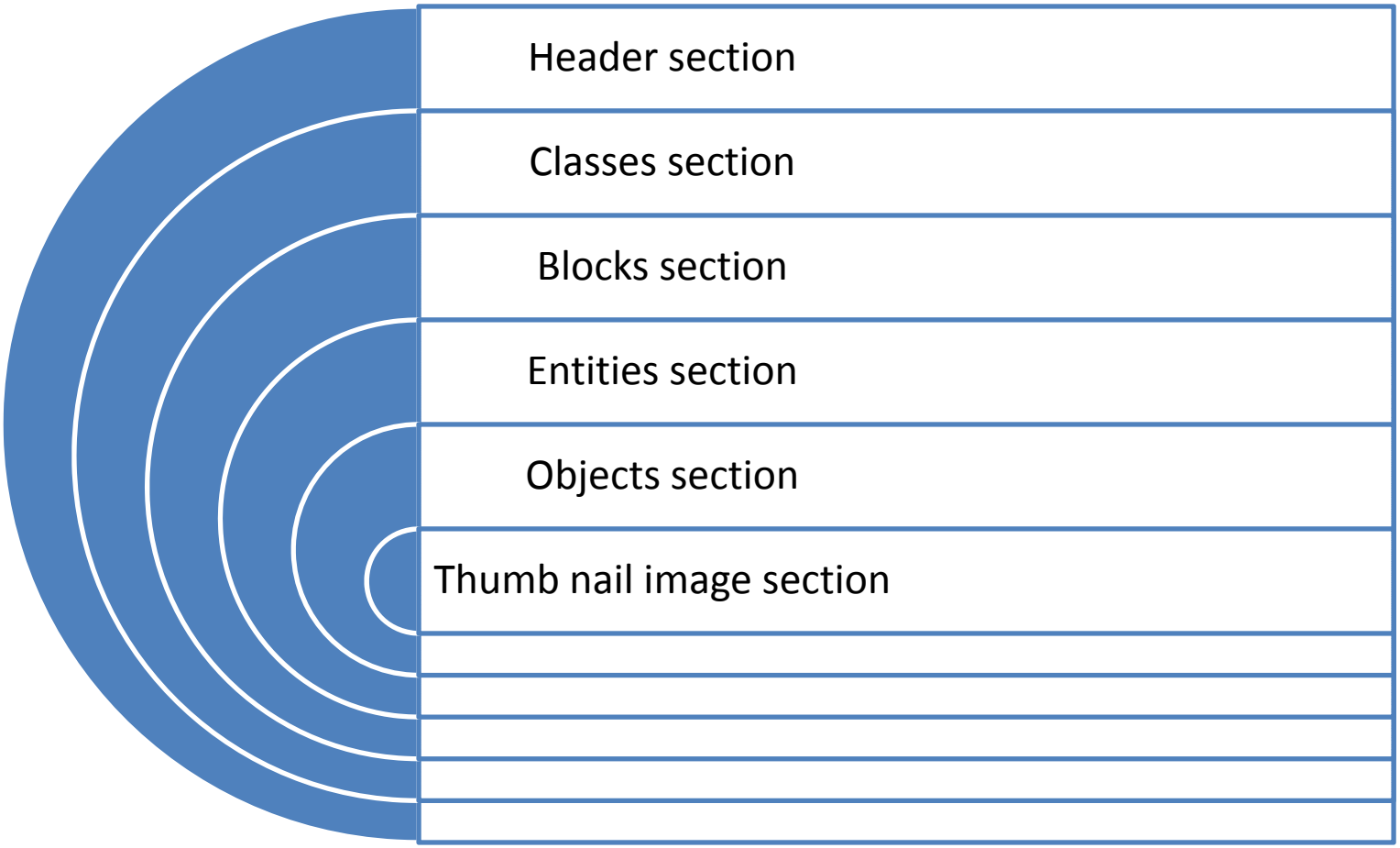
- All the support structures required must be generated in the CAD system and to be sliced.

- Files would not be attached, leaving hundreds of small files needed to be given logical names and then transferred.

2.DRAWING EXCHANGE FORMAT(DXF)

- **AutoCAD DXF** (Drawing Interchange Format, or Drawing Exchange Format) is a CAD data file format developed by Autodesk for enabling data interoperability between AutoCAD and other programs.

File format


Header section
Classes section
Blocks section
Entities section
Objects section
Thumb nail image section

File structure:

- **HEADER** section – General information about the drawing. Each parameter has a variable name and an associated value.

- **CLASSES** section – Holds the information for application-defined classes whose instances appear in the BLOCKS, ENTITIES, and OBJECTS sections of the database. Generally does not provide sufficient information to allow interoperability with other programs.

BLOCKS section – This section contains Block Definition entities describing the entities comprising each Block in the drawing.

ENTITIES section – This section contains the drawing entities, including any Block References.

OBJECTS section – Contains the data that apply to non- graphical objects, used by Auto LISP and Object ARX applications.

THUMB NAIL IMAGE section – Contains the preview image for the DXF file.
END OF FILE

3.CT DATA

- This is particular for medical imaging and is not standardized data.
- Formats are proprietary and unique from one CT scan machine to another.
- Scan generates data as a grid of three dimensional points, where each point has a varying shade of gray indicating the density of the body tissue found at particular point.

Applications:

- Data from CT scans used to build Knee, skull , femur and other bone models on stereolithography systems.
- It is used to generate implants.
- It is used to produce models of human temporal bones.

Approaches for creating model

- CAD systems
- STL-interfacing
- Direct Interfacing

Advantages:

- It is possible to produce structures of the human body by RPT systems.

Disadvantage

- ✓ Increased difficulty in dealing with image data as compared with STL data.
- ✓ Special interpreter to process CT data.

4.IGES (Initial Graphics Exchange Specification)

- It defines a neutral data format that allows the digital exchange of information among Computer-aided design (CAD) systems.

- Using IGES, a CAD user can exchange product data models in the form of circuit diagrams, wireframe, freeform surface or solid modeling representations.

IGES

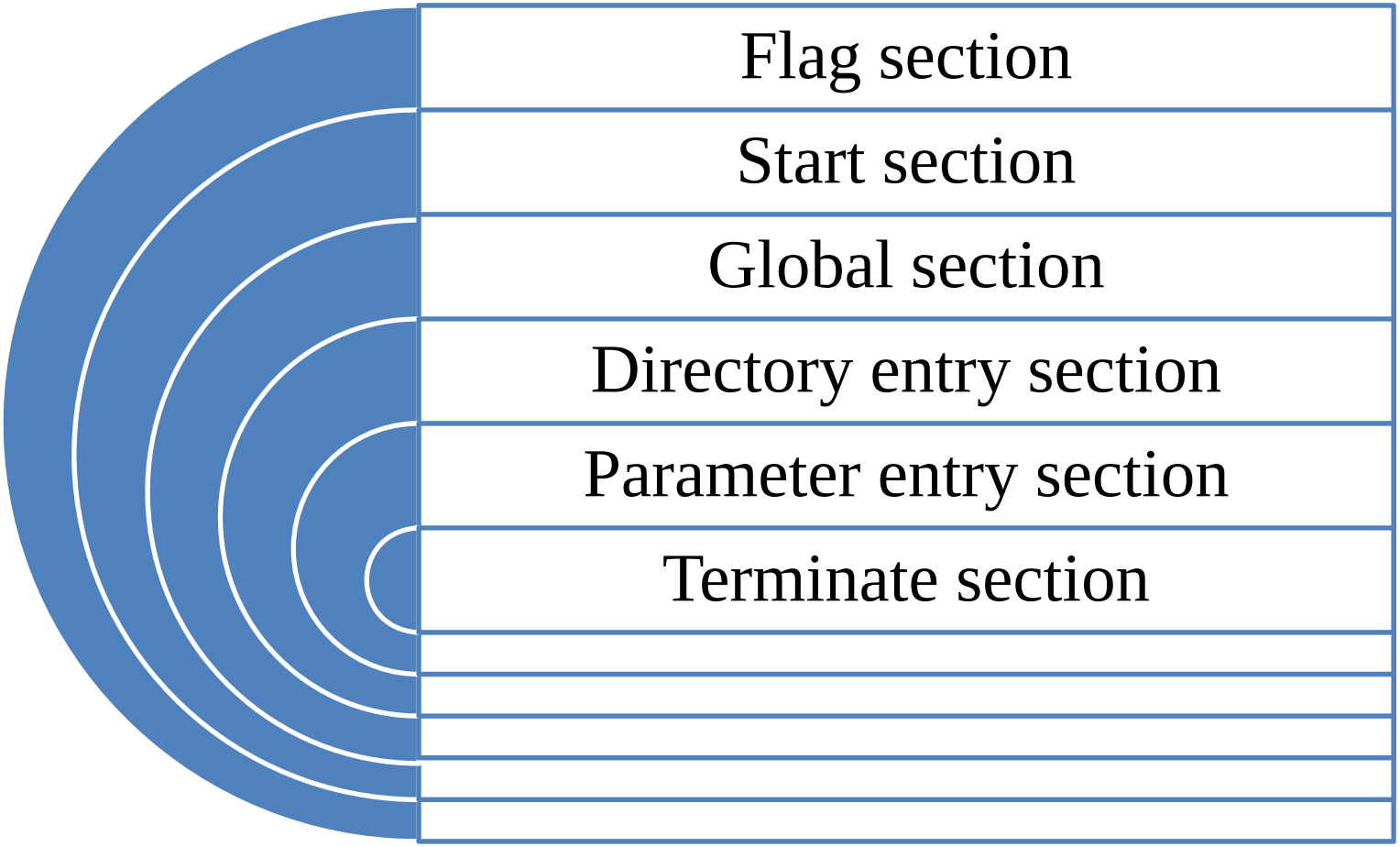
- Applications supported by IGES are

  Traditional engineering drawings.

  Models for analysis.

  Manufacturing functions.
- It includes not only the geometry information (Parameter Data Section) but also topological information (Directory Entry Section).

File structure



Advantages

- Its wide adoption and comprehensive coverage.
- It was adopted in every commercial CAD/CAM system.
- It provides the entity of points, lines, arcs, splines, surfaces and solid elements.
- It can precisely represent CAD model.
- It includes few data conversions, smaller data files, simpler control strategies.

Disadvantages

- Since it is the standard format to exchange data between CAD systems, it also includes much redundant information that is not needed for rapid prototyping systems.
- The algorithms for slicing an IGES file are more complex than the algorithms slicing a STL file.
- The support structures needed in RP systems such as the SLA cannot be created according to the IGES format.

5. SLC (SterioLithography Contour File

- StereoLithography Contour file format is developed at 3D systems.
- It takes 2D slices directly from a CAD model instead of using an intermediate STL model.
- SLC data can be generated from various sources either by conversion from CAD solid from systems which produced data arranged in layers, such as CT scanners.

Key words

- Segment.
- Polyline.
- Contour boundary or profile boundary.
- Contour layer.

Overview of the SLC file structure

- Header section.
- 3D reserved section.
- Sampling table section.
- Contour data section.

Header section

The Header section is an ASCII character string containing global information about the part and how it was prepared.

Header key words:

1. SLCVER.
2. UNIT.
3. EXTENTS like *X,Y,Z* boundaries of CAD model.

   and many more…….

3D reserved section

This section holds a space of 256 byte memory.

This section is reserved for future use.

Sampling table section

- The sample table describes the sampling thicknesses (layer thickness or slice thickness) of the part.
- There can be up to 256 entries in the table at a time.
- Each entry describes the Z start, the slice thickness.

Sample table size: 0.4 0.006 0.005 0.0

0.4 - first Z level of the part.

0.006 - layer thickness.

0.005 - line width compensation value.

Contour data section

The contour data section consists of

- Z layer
- Number of boundaries.
- No.of vertices for the first boundary.
- No.of gaps for the first boundary.
- Vertex list for first boundary.

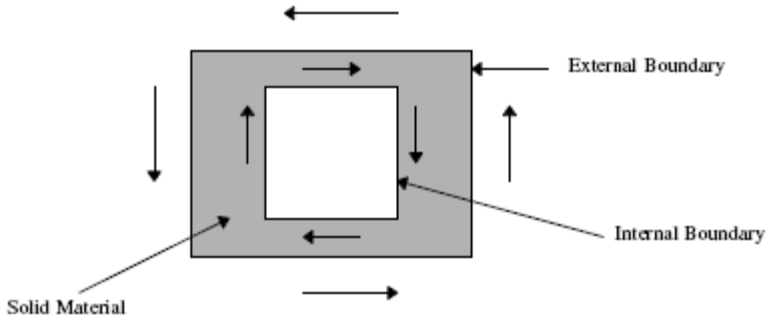


Figure 6.28: Contour boundary description

6. CLI (Common Layer Interface)

- Developed in a Brite Euram Project with support of major European car manufacturers.
- Meant for layer by layer manufacturing technologies.
- Parts are build by succession of layer descriptions.
- CLI file may be in,
  1) ASCII format,
  2) BINARY format.

.

- The geometry part of the file is organized in ascending order.
- Every layer is started by a layer command, giving the height of the layer.
- Ultimately layers consist of series of geometric commands.
- The CLI format has two kinds of entities.

    1. polyline.
    2. Hatching.

Polyline :

- These are closed (i.e.) they have unique sense.
- Sense is directional means , clockwise or counter clockwise.
- Directional sense states that polyline is on the outside of the part or surrounding the hole in the part.
- Clockwise polylines – surrounds holes.

  counter-clockwise polylines – surrounds the part.

Hatching :

- Helps to distinguish between the inside and outside of the part.
- It takes up considerable file space.
- The information and directions are already specified by polylines so that hatching files excluded from output files.

Advantages :-

- Only supports polyline entities, it is simpler format compared to the HP/GL format (Hewlett-Packard Graphics Language).
- The slicing step can be avoided in some applications.
- Error detection in layer information is easier , automated recovery procedure can be used and editing is also not difficult.

Disadvantages :-

- The CLI format only has capability of producing polylines of outlines of the slice.
- Outline curves of parts are obtained by reducing the curve to segment of straight lines, some advantages over STL format are lost.
- CLI format include layer information like the HP/GL format but the CLI format only includes polylines entities, while HP/GL support arcs and lines.
- The CLI format is simpler than HP/GL format and has been used by several rapid prototyping systems so, we hope that the CLI format will become an industrial standard such as STL.

7. RPI (Rapid prototyping Interface) FILE

- **Designed-the Rensselaer Design Research Center, Rensselear Polytechnique Institute.**
- **Developed from currently available STL format data, capable of representing facet solids along with extra information about facet topology.**
- **Topological information maintained by each solid entity with indexed lists of vertices, edges and faces.**
- **Index numbering instead of vertex co-ordinates to reduce redundant information**
- **Format developed in ASCII to facilitate cross-platform data exchange and specification.**
- **RPI format file is collection of entities,each of which defines data in it.**
- **Each entity composed of entity name ,a record count, a schema definition, schema termination symbol and corresponding data.**

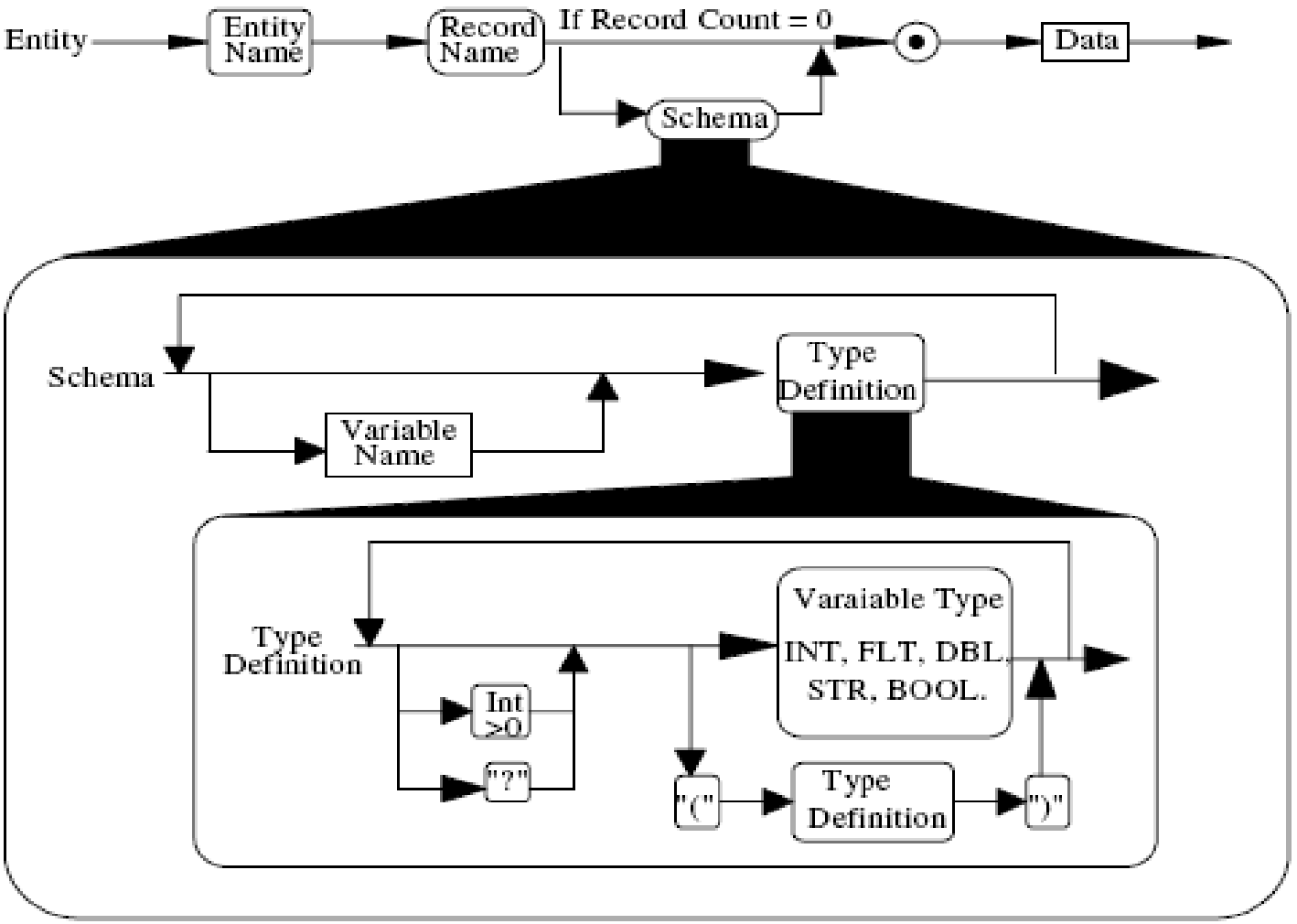


Figure 6.29: RPI format entity syntax diagram

ADVANTAGES & DISADVANTAGES

Advantages :-

- **Topological information added to RPI format due that flexibility is achieved and allows users to balance storage and processing data.**
- **Redundancy removed and compact size.**
- **Representation of CSG primitives provided and represent multiple instances of both facet and CSG solids.**
- **Format extensibility is made possible by interleaving format schema.**

Disadvantages :-

- **An interpreter which processes a format as flexible and extensible as the RPI format is more complex than STL format.**
- **Surface Patches suitable for solid approximation can not be identified in the RPI format.**

8. LEAF-LAYER EXCHANGE ASCII FORMAT

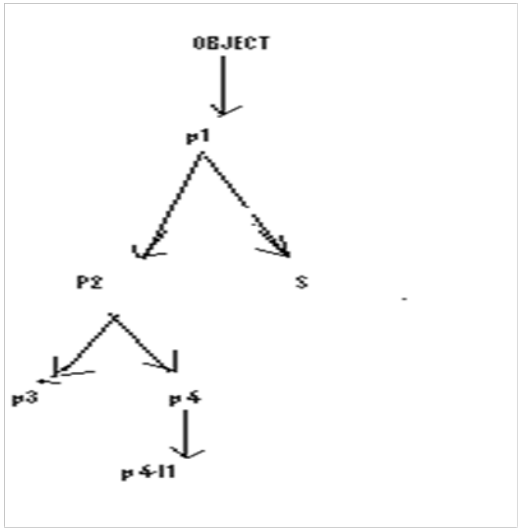
- **Developed by:-Helsinki University of Technology.**

- **To Describe this model concepts from object oriented paradigm are borrowed.**

- **The LEAF format is described at several levels ,mainly logical levels using data model based on object oriented Concepts .**
- **Object Tree**

LMT-file→ PART→ LAYER→ 2D PRIMITIVES.

- **EXAMPLE:**



ADVANTAGES

- **Easy to implement and use.**
- **Not ambiguous.**
- **Allows data compression.**
- **Machine as well as LMT process independent .**
- **Slices of CSG model can be represented directly in LEAF.**
- **Support structures are easily separated from the original part.**

DISADVANTAGE

- **New interpreter is needed to connect RPT systems.**
- **Structure format is more complicated than that of STL format.**
- **STL format cannot be changed in this format.**