Introduction of Parasolid

Basic Concepts & Functionalities

Human Centered CAD Laboratory

What are the lectures about?

Contents

- Introduction of Parasolid
- Feature of Parasolid
- Functionality of Parasolid
- Application Demo

Requirements

- Background knowledge of CAD,OpenGL,C++
- For details please refer to Parasolid Online Documentati on

Online Document

Web Address

- http://cad.snu.ac.kr/parasolid

Document about

- Description about Parasolid
- Explanation of PK Function
- Code Example based on Parasolid

What's Parasolid (1)

Geometric Modelers

- **PARASOLID**， Unigraphics Solutions Inc: Unigraphics, IDEAS, SolidEdge, Solidworks
- **ACIS**，Spatial Technology: AutoCAD，CADKEY，Mechanical Desktop，Bravo
- **CAS.CADE**，MDTV(Matra Datavision): Euclid-IS, QUANTUM

What's Parasolid (2)

- Parasolid: an exact boundary representation (B-rep) geometric modeler
  - Build and manipulate solid objects
  - Calculate mass property (mass center, inertia)
  - Output objects in various ways
  - Store objects in files on disk, and retrieve them later
  - Clash detection
- Kernel Features
  - Hardware independent-Windows NT, UNIX, Linux
  - OS, GUI independent
  - Written in C

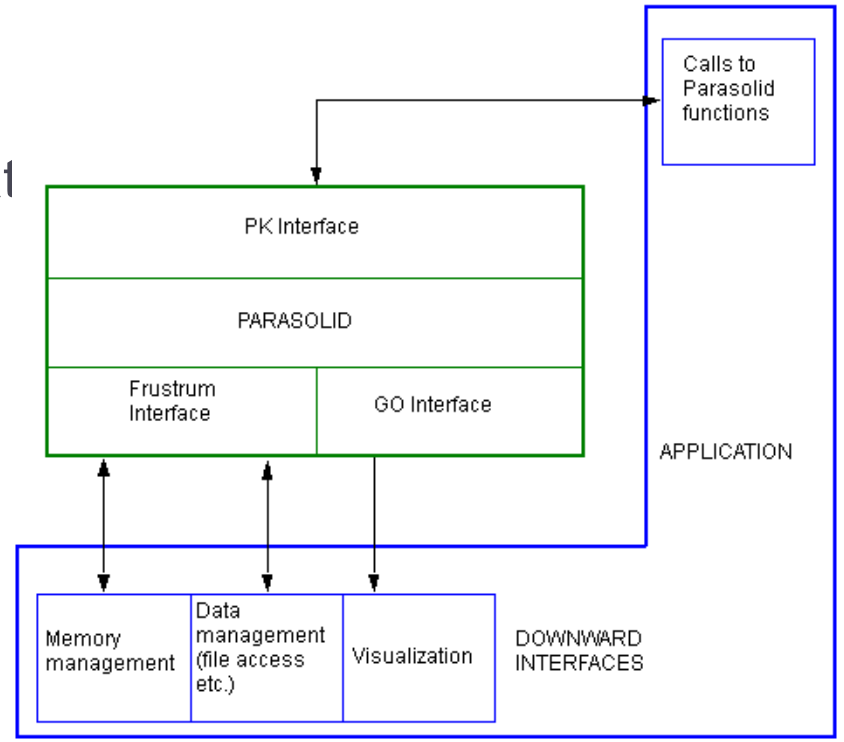
Parasolid Interfaces

PK Interface

- The main interface of Parasolid
- They are the means by which applicat models and manipulates objects and controls the functioning of the modeler.

Downward interface

- They are called by the modeler to perform data-intensive or system type operations.
- **Frustrum**, **Graphical Output (GO)**



Interfaces between Parasolid and application

PK Interface(1)

- PK interface

  The PK Interface (usually referred to as the PK) is a library of functions that provide access to Parasolid.

- Four categories

  Part construction

  Functions for building, modifying, and combining parts

  Enquiry

  Functions for returning information about the properties of parts, computing mass properties, geometric information, or rendering information.

  Kernel management

  Functions for managing the characteristics of the kernel, such as rollback information, frustrum functions, session parameters, and so on.

  Part management

  Functions for loading and saving part files, and for managing attributes.

PK Interface(2)

- Getting started with the PK interface
  - Call PK_SESSION_start to start a kernel **session**
  - Call PK_SESSION_stop to stop the kernel **session**.
- Classes of PK interface functions

  PK interface functions fall into the following sub-classes:

  CLASS---Navigating the class hierarchy

  SESSION---Session management

  ERROR---Error handling

  MEMORY---Memory management

  PARTITION, PMARK, MARK, DELTA---Rollback mechanism

PK Interface(3)

- PK function form: PK_<OBJECT>_<text>
  - <OBJECT> the entity class that the function operates on. BODY,FACE and so on.
  - <text> a verb/noun combination that describes the operation.
    create, make, delete for example.
  - Example: **PK_BODY_create_solid_block**
- Arguments to PK functions:
  - Simple values, arrays, and structures.
  - Using an options structure for options.
- Error handling
  - Return value PK_ERROR_no_errors means being successful
  - Error handle methods: registering an error handler or do yourself by checking the return value of each function

PK Interface(4)

- PK function Example

```
PK_BODY_create_solid_block
( --- received arguments ---
double x, --- block extent in local x direction (>0)
double y, --- block extent in local y direction (>0)
double z, --- block extent in local z direction (>0)
const PK_AXIS2_sf_t *basis_set, --- position and orientation
--- returned arguments ---
PK_BODY_t *const body --- solid body returned
)
```

Downward Interfaces (frustrum)

The frustrum

- The Frustrum is a set of functions which must be written by the application programmer. They are called by Parasolid to perform the following tasks

  - Frustrum control
  - File (part data) handling: Saving and retrieving Parasolid part files and other data.
  - Memory management : Allocating memory for internal calculations and data structure storage.
  - Graphical output

- Before start a session for using Parasolid Kernel, registering the frustrum

Downward Interfaces (frustrum)

Frustrum Functions you may need to provide:

FSTART: initialize the Frustrum

FMALLO: allocate a contiguous region of virtual memory

FMFREE: free a region of virtual memory (from FMALLO)

FFOPRD/FFOPWR: open all guises of Frustrum file

FFREAD/FFWRIT:read from/write to a file where permitted

FFCLOS:close a Frustrum file

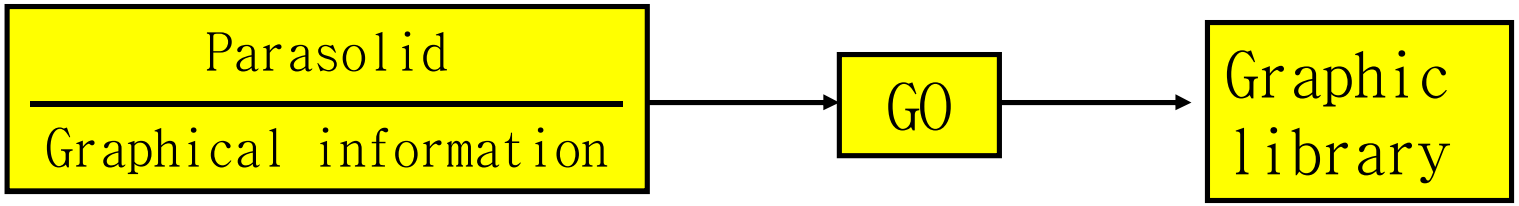
FABORT:tidy up/longjump following aborted operation

FTMKEY:key name server required by TESTFR

FSTOP:close down the Frustrum

Downward Interfaces (GO)

Graphical Output (GO): required for displaying models



Rendering Function to produce line data

PK_GEOM_render_line

PK_TOPOL_render_line

PK_TOPOL_render_facet

GO Functions

GOOPSG: Open a hierarchical segment.
GOSGMT: Output a single-level segment.
GOCLSG: Close a hierarchical segment.

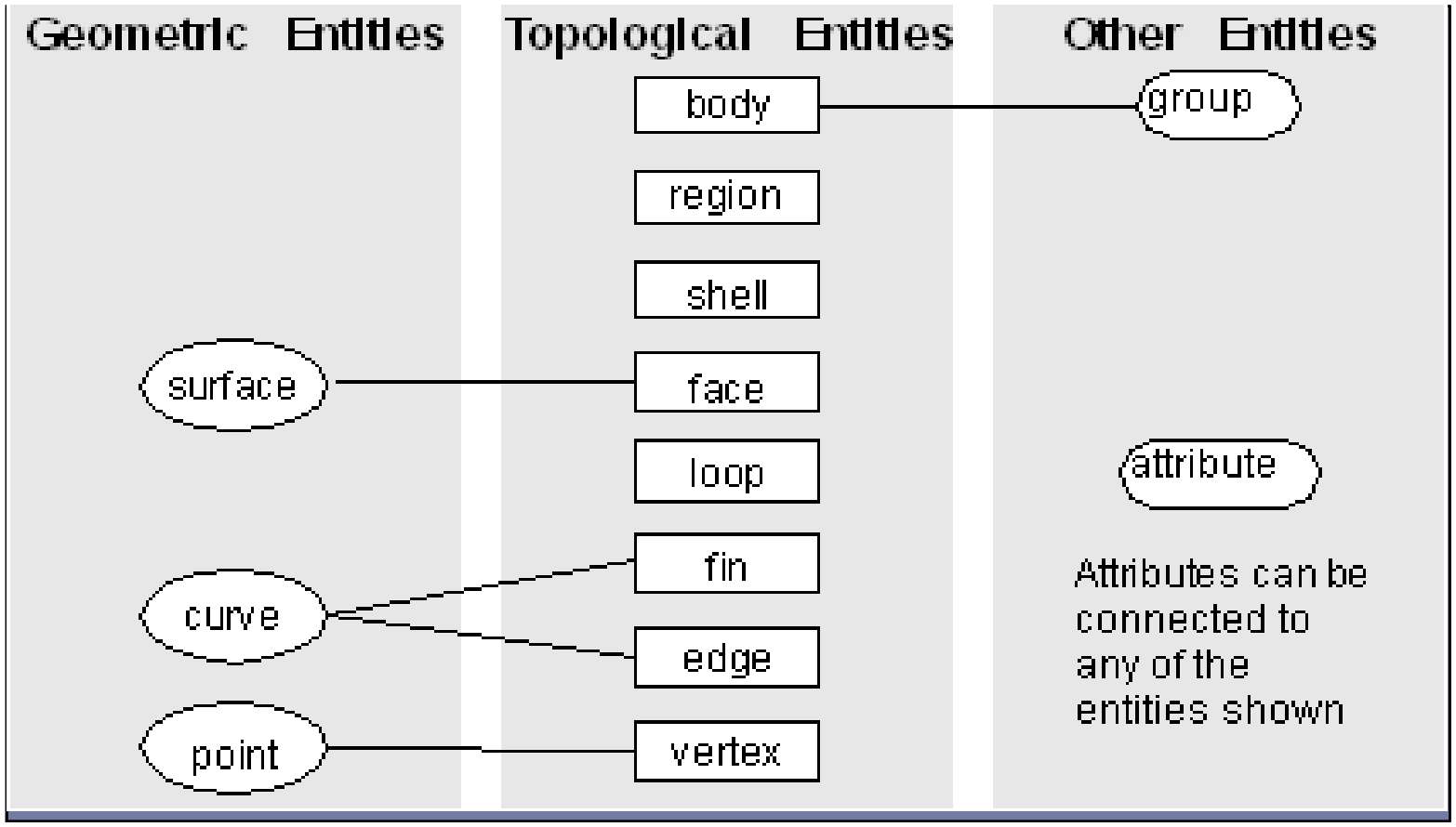
File type & File extension

- **Filename extensions**
  text-based: .*_t
  binary-based: .*_b (recommended)

- **File type (FAT)**
  Transmit (Part): .x_t , .x_b
  Schema: .s_t,.s_b
  Journal: .j_t, .j_b
  Snapshot: .n_t .n_b
  Partition: .p_t , .p_b

Entities in Parasolid

Entities are grouped into three main types.



Topological Entities (1)

Topological Entities---comprise all the entities that con stitute the structure or skeleton of a model

1. Body

Acorn: An isolated vertex.

Wire: Connected edges. (manifold)

Sheet: Connected faces. (manifold)

Solid: A solid region.

General

2. Region

- Subset of 3-d space
- A body always has an infinite void region, all regions in a body comprises the whole of 3D space.

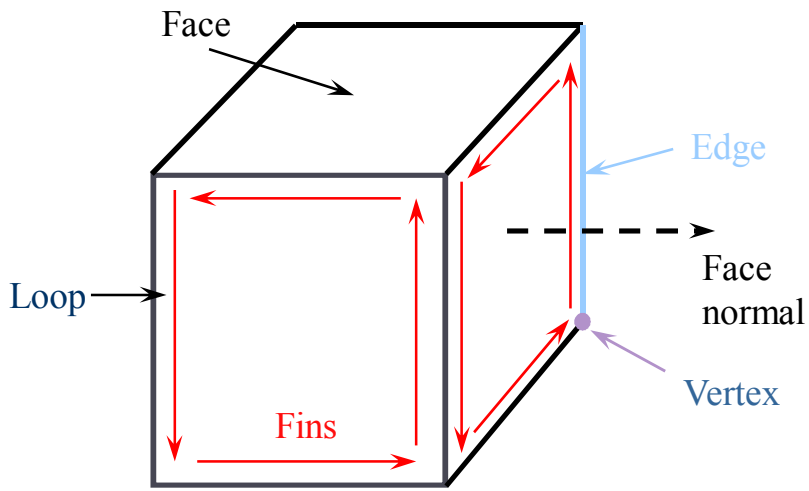
Topological Entities (2)

3. Shell

Can be regarded as a bo
undary of a region

4. Face

A subset of a surface



5. Loop

- A connected component of a face boundary.
- The direction of the loop is such that the face is locally on the left of the loop, when seen from above the face and looking in the directi on of the loop.

Topological Entities (3)

6. Fin---represents the oriented use of an edge by a loop

7. Edge---An edge is a bounded piece of a single curve A **wir**
eframe --- no fins
A **laminar** --- one fin
A **manifold** --- two fins
A **general** edge --- more than 2 fins

8.Vertex---A vertex represents a point in space

Manifold Body (1)

Minimal body

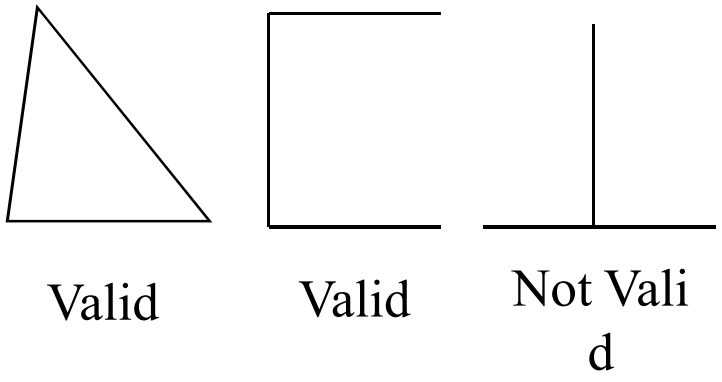
- ‘Zero dimensional’
- A point in space
- One void region, shell, loop, vertex
- Created from a point using PK_POINT_make_minimum_body

Wire

- ‘One dimensional’
- A set of simply-connected edges
- No more than two edges at any vertex
- One void region, one shell

Acorn

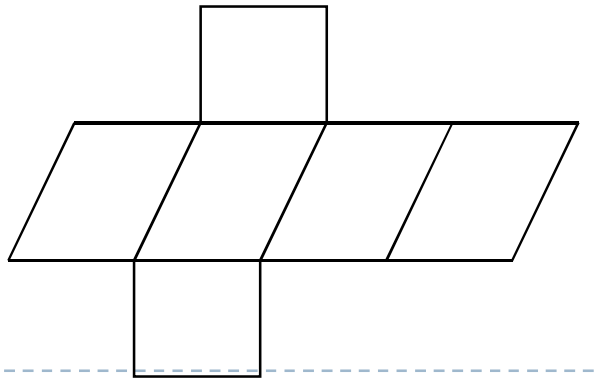
- The same as Minimal bodies except they can have multiple vertices



Manifold Body (2)

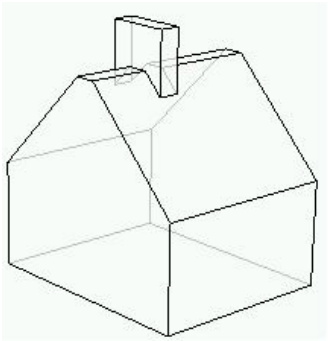
Sheet

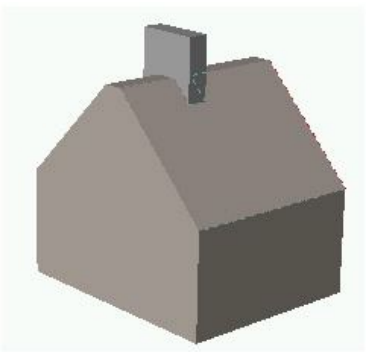
- 'Two dimensional'
- Set of (simply) connected faces (at least one face)
- Zero thickness
- No more than 2 faces at any edge
- Can be 'open' or 'closed'
- One or two void regions and one or two shells



Solid

- 'Three dimensional'
- Occupies a continuous, finite volume
- At least two regions
  - one solid region
  - any number of void regions
  - one void region is infinite



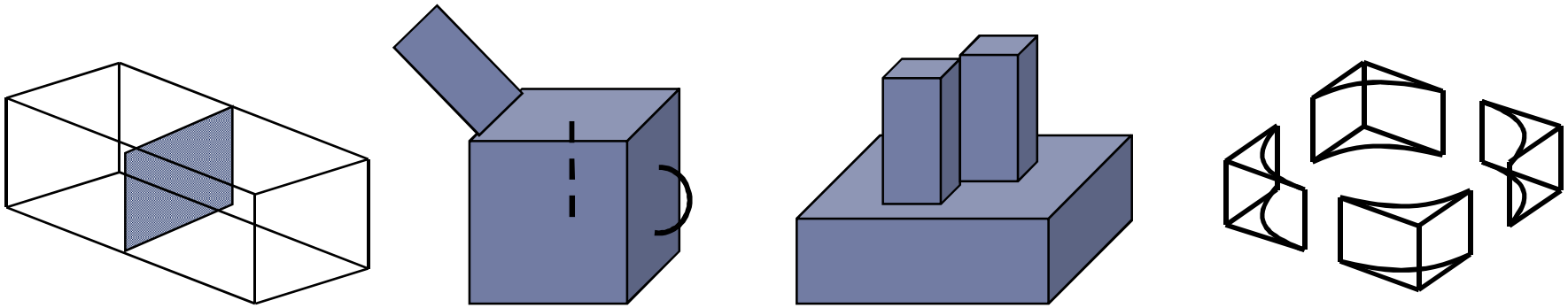


General Body

manifold bodies

default type, Acorn body, wire body, sheet body, solid body

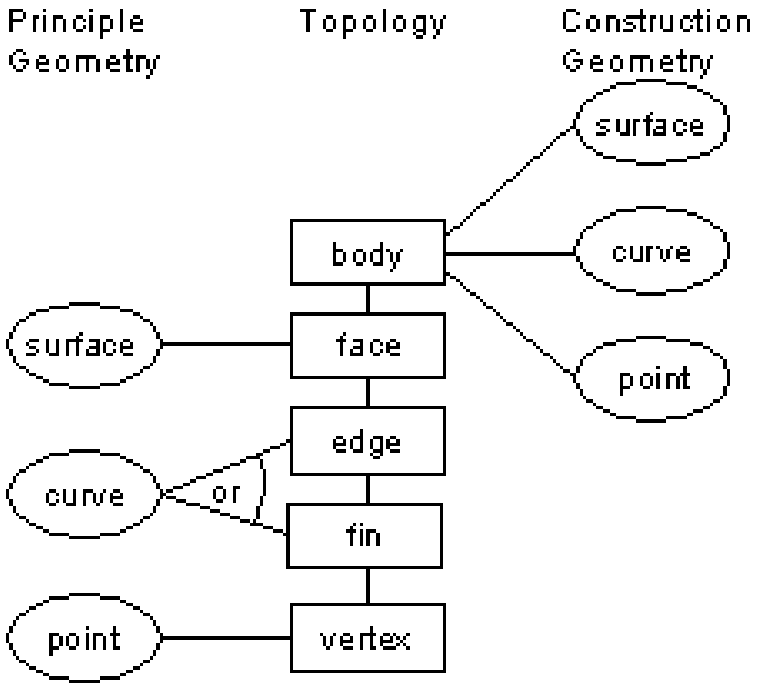
general bodies



1. Cellular body - general body partitioned by an internal face
2. Mixed Dimension body
3. Non-manifold body - the edge between two bosses is non-manifold
4. Disjoint body - consisting of four disconnected pieces

Geometrical Entities

Orphan geometry

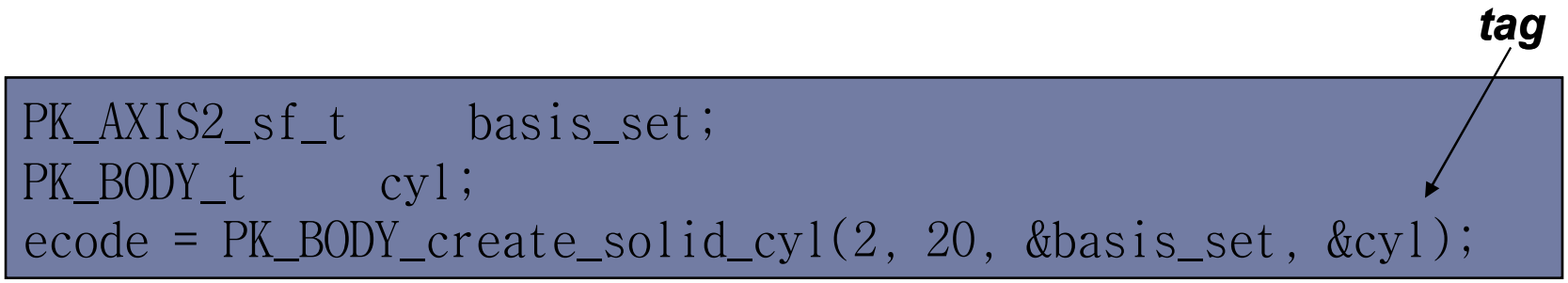


Orphan geometry is geometry not attached to any topological entity

Entity related

Tag

- Identifier of a particular entity within a session.
- Being unique in one session
- Integer PK functions use as pointer to an entity
- Lifetime: while the entity it refers to still exists
- Is created only within the kernel and can only be got from the kernel



Parasolid Functionality

Model representation

- Geometry & Topology
- General Topology
- Tolerant modeling

Creation & editing

- Primitives
- Lofting & sweeping
- Blending
- Hole Filling
- Booleans & patterning
- Offset, hollow, thicken
- Face Change
- Model Simplification
- Tapering

Enquiries

- Data structure enquiries
- Mass properties
- Closest Approach
- Clashing & Containment

Rendering & Selection

- Wire frame, hidden line
- Faceting

Application support

- Attributes
- Session & partitioned rollback

Data import/export

- XT format
- Trimmed surface & B-rep support

Creating Manifold Bodies (1)

Four ways to create a manifold body

- make a primitive body from raw data
- make a body from geometry
- make a body from topology
- make a body from an existing body

Creating Manifold Bodies (2)

- **Creating a primitive body from raw data**

  - A series of PK functions can be used to generate all the topology and geo metry required for block, cone, cylinder, prism, sphere, torus

  - examples: PK_BODY_create_solid_block, PK_BODY_create_sheet_circl e

- **Making a body from geometry**

  - The PK functions also are available to create a body from existing geomet ry

  - Example: PK_SURF_make_sheet_body

- **Making a body from topology**

  - Some local operations can create a body from existing topology

  - Example: PK_FACE_make_sheet_body

Creating Manifold Bodies (3)

- **Making a Body from an Existing Body**
  - copy body

    PK_ENTITY_copy

  - Extracting manifold bodies from a general body

    For example, this could be used when a boolean operation has return ed a general body containing manifold components

    PK_BODY_make_manifold_bodies.

Simple to Complex body (1)

Several operations can be used to generate a more co mplex type of body

- Sweep and spin
- Imprint
- Thicken
- Change region type
- Add rubber face
- Pierce

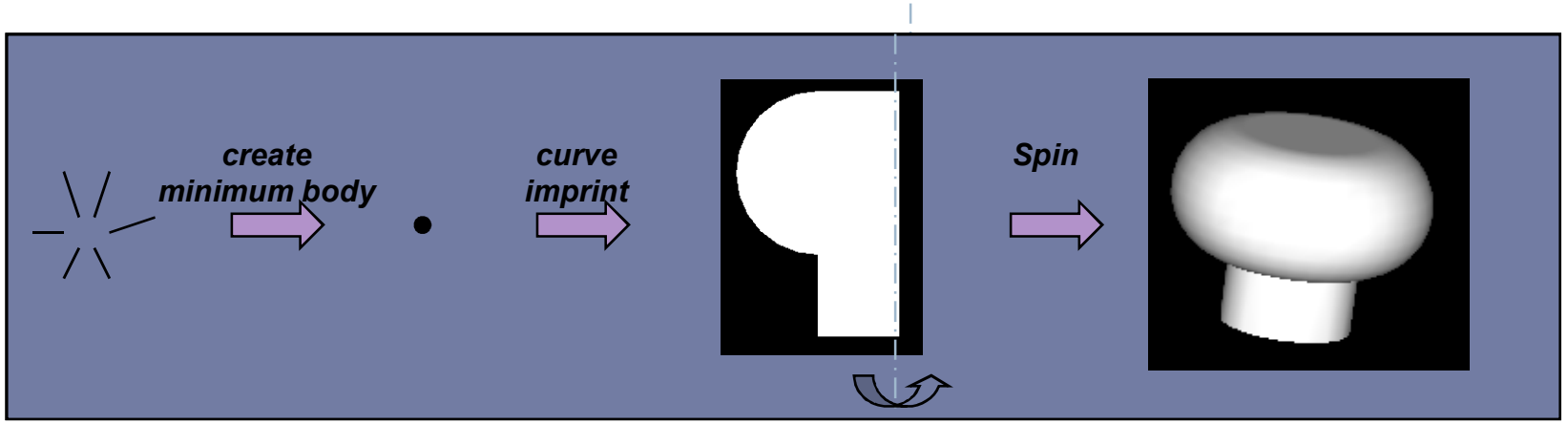
Simple to Complex body (2)

Sweep and spin

- change the type of the body
- PK_<ENTITY>_sweep, PK_<ENTITY>_spin

Original & resultant entities:

- minimum bodies - wire bodies
- wire bodies - sheet body
- sheet bodies - solid body
- faces of solid bodies - an extension to the solid



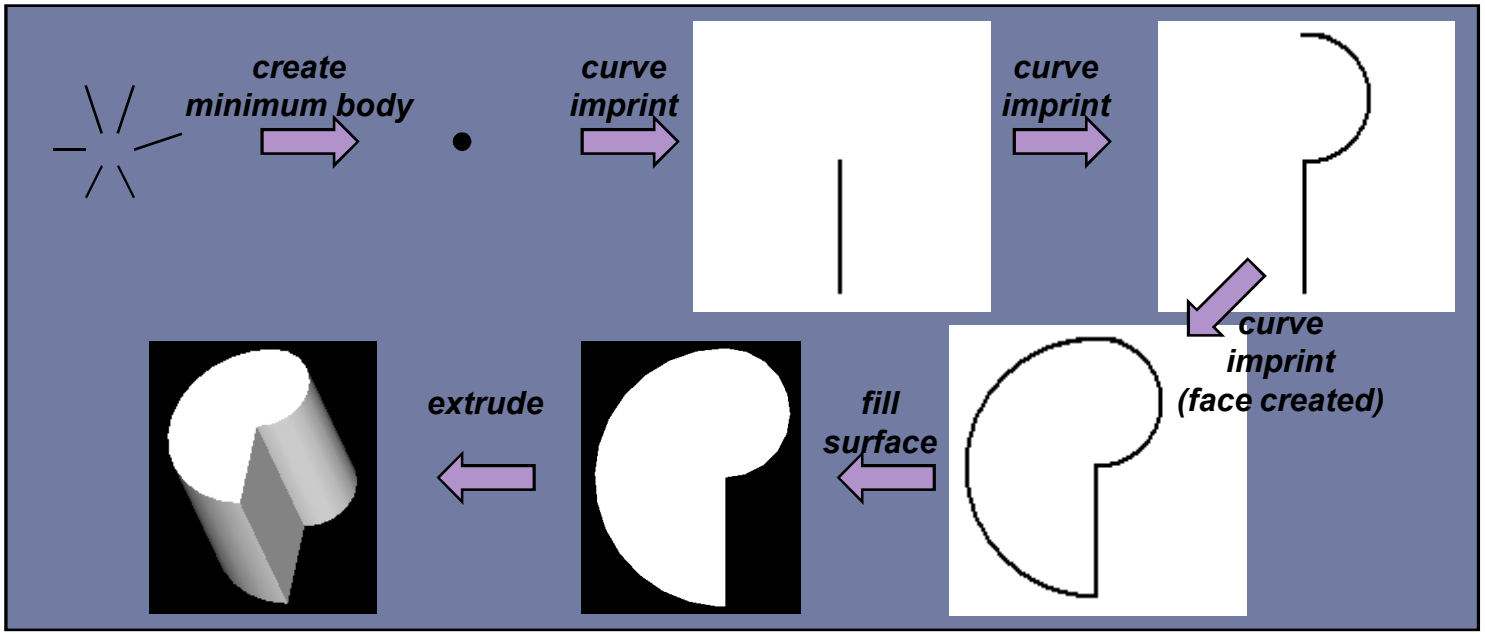
Simple to Complex body (3)

Imprint

PK_BODY_imprint_curve, PK_FACE_imprint_curve

PK_REGION_imprint_curve

- to create new faces
- to create a profile

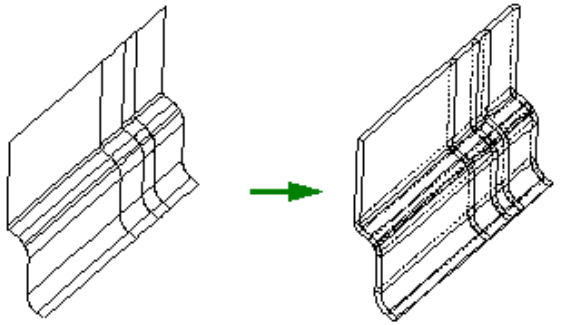


Simple to Complex body (4)

Thicken

thickens a sheet body into a solid body

PK_BODY_thicken



Change region type

PK_REGION_make_solid : 3D space defined by sheet body

PK_REGION_make_void : solid convert to sheet body

Simple to Complex body (5)

Add rubber face:

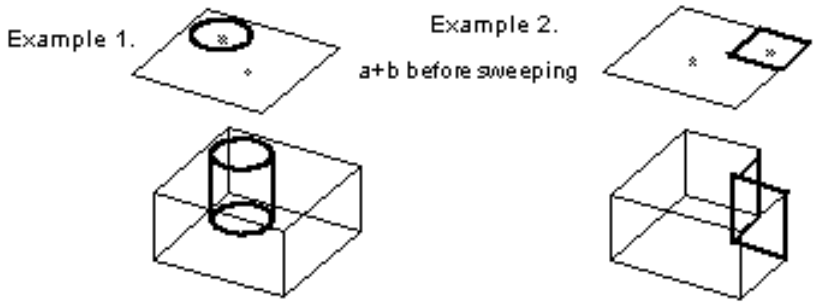
PK_EDGE_make_faces_from_wire

be used to attach rubber faces to loops of wireframe edges in a wire b ody, thereby creating a sheet body.

Pierce

PK_FACE_delete_from_sheet_body

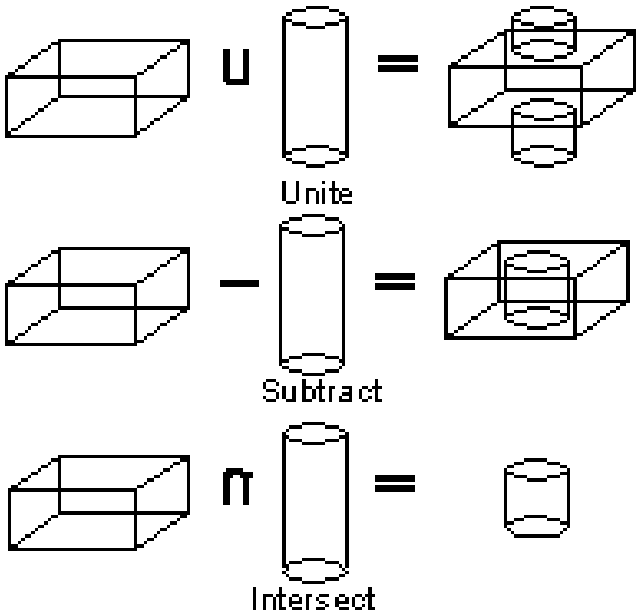
be used to create a solid with holes through it from a profile.



Simple to Complex body (6)

Boolean Operation

- Union
- Intersection
- Subtraction



Inquiries (1)

Class Structure

- PK_ENTITY_is, PK_ENTITY_is_<object>
- PK_ENTITY_ask_class

Data Connections

- Topological Relationships
- Topological/Geometric Relationships
- Session management relationships

Data Relationships

- PK_FACE_ask_oriented_surf

Inquiries (2)

Topological

- PK_REGION_is_solid
- PK_BODY_ask_type
- PK_SHELL_find_sign

Geometric

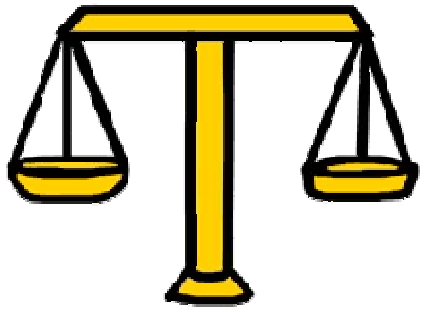
- PK_CURVE_eval ; returns points and derivatives
- PK_SURF_eval_with_normal ; returns surface normal
- PK_SURF_ask_param ; returns parameters of surface

Inquiries (3)

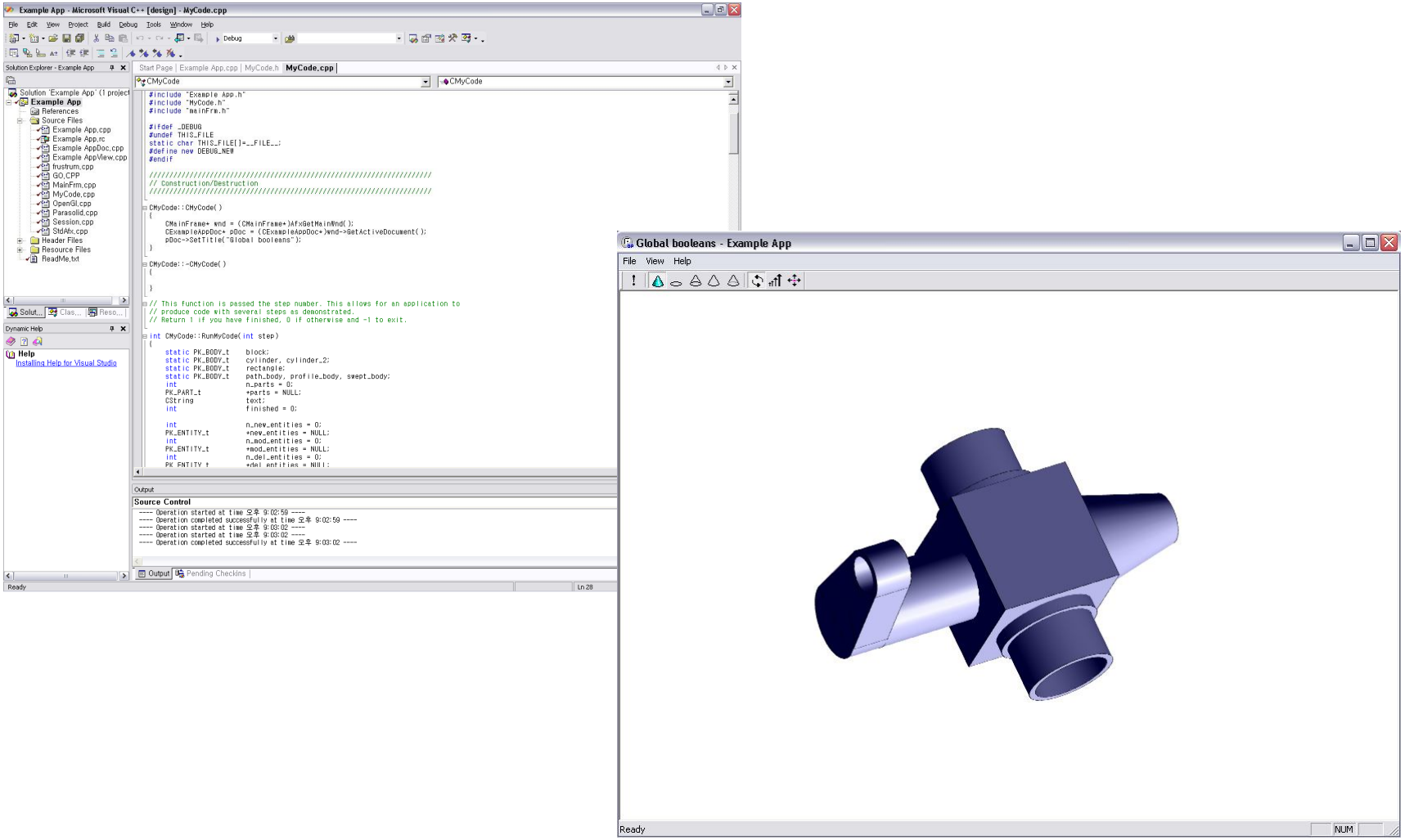
Mass Properties

- PK_TOPOL_eval_mass_props

Edge	Length
Face	Perimeter, Area, CofG
Wire Body	Length, Mass
Sheet Body	Circumference, Surface area, Mass, CofG
Solid Body	Surface Area, Mass, CofG, Moments of Inertia
Assembly	



Application Demo



THANK YOU