Parasolid

Overview of Parasolid

July 2022

Important Note

Francis House
112 Hills Road
Cambridge CB2 1PH
UK
Tel: +44 (0)1223 371555
email: parasolid.support.plm@siemens.com
Web: www.parasolid.com

Trademarks

Table of Contents

Introduction 1

1.1 Welcome

Welcome to the *Overview of Parasolid*. This manual provides a detailed overview of the powerful and versatile 3D modelling functionality that the Parasolid component software library brings to your application. This manual gives a broad, high-level overview of Parasolid functionality. Not all areas are covered, and more detailed information is provided in the full Parasolid documentation set.

The *Overview of Parasolid* is intended for anyone evaluating, or thinking of evaluating, the Parasolid kernel modelling library. The information it contains is useful to senior managers, product planners, system architects and developers alike.

This manual does not contain any specific details about the Parasolid API – the functional interface between your application code and Parasolid. Material on integrating Parasolid with your application and learning the Parasolid API is available on the Parasolid Jumpstart Kit: see Section 2.3 for details.

1.2 What is in this manual?

The *Overview of Parasolid* is divided into the following chapters.

Chapter 2, “About Parasolid”, gives a brief summary of the various functional areas covered by the Parasolid library, and tells you about the various components that comprise the Parasolid product. This chapter includes a comprehensive overview of the teaching and reference material available for learning how to incorporate Parasolid into your application and start using it.

Chapter 3, “Model Structure”, explains the structure used for models created using Parasolid, as well as the different types of body that are supported.

Chapter 4, “Booleans and Related Functionality”, describes the functionality available for uniting and subtracting bodies, for finding the intersections between bodies, and for using features on a body to create instances and patterns.

Chapter 5, “Local Operations”, describes Parasolid’s local operations functionality. Local operations are modelling operations that affect just a part of a model, and include operations that change the topology of a model, such as tapering, transforming or replacing geometry, as well as hollowing and offsetting operations, and functionality to simplify parts of a model.

Chapter 6, “Working with Sheets and Wires”, explains how to model with wires and sheets, including functionality for trimming, sewing together and thickening sheets.

Chapter 7, “B-Spline Curves and Surfaces”, describes Parasolid’s functionality for curves and surfaces.

Chapter 8, “Convergent Modeling”, provides an overview on using mesh information for operations instead of classic geometry.

Chapter 9, “Building Bodies from Profiles”, explains how you can extrude, spin, sweep or loft between profiles (wire or sheet sketches) to create entirely new bodies.

Chapter 10, “Blending”, describes Parasolid’s comprehensive and unrivalled blending functionality. Parasolid can replace edges in a body with blend faces, or create new blend faces between other sets of faces.

Chapter 11, “Importing Foreign Data”, describes how you can import data from other modelling systems into Parasolid. Dedicated functionality is provided to help you successfully import such data, create Parasolid entities from it, and then model reliably in Parasolid using this new data.

Chapter 12, “Storing Data”, describes Parasolid’s functionality for storing data to and retrieving parts from external storage such as disk or databases. It explains Parasolid’s use of the XT format, and the concept of the Parasolid pipeline for straightforward data exchange between Parasolid-powered applications.

Chapter 13, “Enquiring Model Data and Identifying Details”, describes the wide range of functionality for returning information on the status of any Parasolid entity. Also discussed is Parasolid’s support for model analysis functionality, via such operations as detection of clashes between bodies, the calculation of mass properties information, and closest approach information.

Chapter 14, “Displaying Data”, describes the support provided for displaying your model on screen, and for storing graphical information for later display.

Chapter 15, “Application Support”, describes the support Parasolid provides for recording information that is not directly related to modelling. This includes the ability to attach arbitrary information to entities, support for recording the history of, and moving forwards and backwards through, a modelling session.

Chapter 16, “Writing Parasolid Applications”, gives a brief overview of the steps you need to take to integrate Parasolid into your application code.

Chapter 17, “Multi-Processing Support”, describes Parasolid’s support for running on machines that includes several processors.

1.3 What platforms are supported?

The Parasolid library is available on the following platforms:

- Intel-based PCs running Windows 8.1, Windows 10, or Linux
- AMD-based PCs running Windows 8.1, Windows 10, or Linux
- AMD and Intel-based 64-bit PCs running Windows 8.1, Windows 10, or Linux
- Intel-based Apple Macintosh systems running Mac OS X 10.12 or later and iOS 11 or later.
- Android 7.0 or later running on 64-bit ARM-based devices

1.3.1 Security processes for Parasolid development

Parasolid is compliant with the Microsoft Security Development Lifecycle (Microsoft SDL) standard. This standard is a set of rules and coding practices produced by Microsoft for use in producing software free from common security flaws, such as easily exploitable buffer overflow. For more information see Section 6.6.2, “Microsoft Security Development Lifecycle (MS-SDL)”, in the Parasolid *Installation Notes*.

1.3.2 Symmetric Multi-Processing

Parasolid is **thread-safe** on all platforms, in that it can be called from multiple threads within an application safely. In addition, many Parasolid API functions can be executed simultaneously by different application threads. The Parasolid API, described in Chapter 16, “Writing Parasolid Applications”, manages application threads to ensure safety.

A number of functional areas within Parasolid have been adapted for symmetric multi-processing (or parallel processing) via the multi-threading environments supported under Microsoft Windows, UNIX, and Linux. This can improve the performance of Parasolid when executing on multi-processor platforms. See Chapter 17, “Multi-Processing Support”, for more information.

1.4 The Parasolid product portfolio

The Parasolid product portfolio includes several toolkits that provide functionality for integrating digital design information into your application, together with the various technologies associated with those toolkits. The Parasolid library is the main topic of the current manual, but a complete list is given below.

1.4.1 Parasolid

Recognised as the world's leading, production-proven core solid modeler, Parasolid is an exact boundary-representation geometric modeler supporting solid modelling, generalized cellular modelling and integrated free-form surface/sheet modelling.

1.4.2 Parasolid Bodyshop

Parasolid Bodyshop is a toolkit targeted towards solving problems that typically arise during the exchange of data between different CAx applications, and improving the quality of the translated data.

Whether you already have a translator solution, or you are developing your own Parasolid translator, Parasolid Bodyshop incorporates suites of tools to assist you and helps to improve the overall success rate of your data conversion.

For more information about Parasolid Bodyshop, send an email to `ps-support.plm@siemens.com`.

1.4.3 Parasolid Translators

Siemens Industry Software also offers a number of translator toolkits for translating between alternative solid modelling formats and the Parasolid format. These toolkits give you the capability to include native translator functionality in your applications, allowing you to read and write solid modelling data in a variety of different formats.

Parasolid Translator for STEP is a bi-directional translator between Parasolid and STEP-AP203. It can create STEP-AP203 files that represent Parasolid models and it can read STEP-AP203 files and produce Parasolid models from them. Parasolid STEP supports STEP AP203 and AP214 Class II, class III, class IV, class V entities and class VI entities and assemblies.

Parasolid Translator for IGES is a Parasolid-powered component library that offers a robust and reliable solution for bi-directional translation of geometric data between Parasolid-powered applications and other heterogeneous CAD systems through IGES files. Parasolid IGES also supports JAMA-IS version 1.04: an IGES data exchange protocol established for the Japanese automobile industry to realize smooth data interchange among dissimilar CAD/CAM systems.

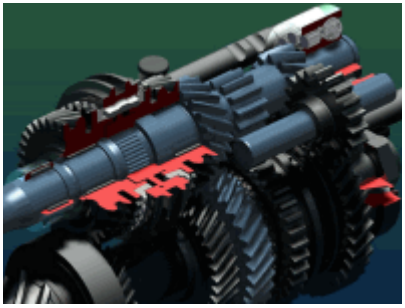
Parasolid Translator for ACIS is a software toolkit that supports bi-directional translation of data between Parasolid and Spatial Technology's ACIS SAT format. The licensee must acquire the required ACIS licenses to use Parasolid ACIS.

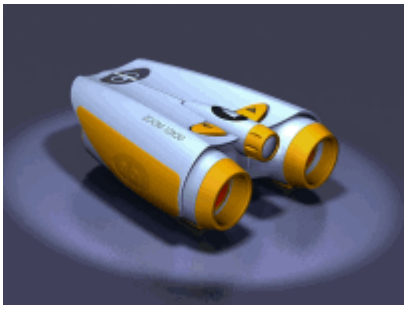
For more information about Parasolid Translators, send an email to `ps-support.plm@siemens.com`.

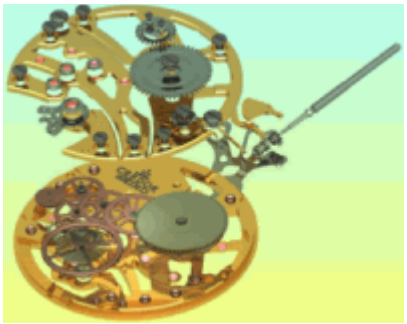
About Parasolid 2

2.1 What is Parasolid

Parasolid offers high performance modelling in a broad range of areas, giving you the ability to create new models or edit existing models using an unparalleled range of tools. The major areas of functionality supported include:

Functionality	Description	
Complex blending	Parasolid provides a wide range of blending on complex geometry with unmatched reliability. Capabilities include rolling-ball, variable-radius, face-face, cliff-edge, overlapping, curvature-continuous, disc, conic-section and conic hold line.	
Hollowing, shelling, offsetting and thickening of surfaces	These invaluable techniques create thin-walled parts. They are conceptually simple to the CAD user but the topological and geometric changes required are a rigorous test of a modeler's reliability.	
Tapering and parting line calculations	These operations primarily serve mold and die designers. Parasolid can apply a taper that follows complex parting lines. These capabilities and non-uniform scaling make Parasolid a powerful mold-design tool.	

Functionality	Description	
Local operations	Many Parasolid operations may be applied to specific areas of a body, such as faces or groups of faces. Parasolid ensures the integrity of models to which these operations are applied remains intact, by making appropriate changes to surrounding topology and geometry. Supported local operations include replacing and transforming specific topologies, hollowing and offsetting, and model simplification via the identification and removal of features such as holes and blends.	
Complex modelling using B-surfaces	Parasolid contains fully integrated B-curves and B-surfaces using industry standard NURBs. The kernel also simplifies geometry to analytic surfaces (planes, cylinders, cones, spheres and tori) whenever possible to optimise reliability and performance. If more free-form construction techniques are required, NURBS of any shape can be arbitrarily trimmed to meet a designer's concept. Such surfaces can be sewn to construct a solid or a sheet model as required.	
Convergent Modeling™ technology	Parasolid supports functionality for performing operations on facet bodies which have Parasolid topology but reference mesh and polyline data rather than classical geometry. Topological operations include, but are not limited to, booleans, hole-filling, local operations, offsetting, thickening, hollowing, rendering, local operations, sectioning, replacement of surfaces, as well as topological and geometric enquiries.	

Functionality	Description	
Rendering support for large models	In today's digital design environment, parts are no longer designed in isolation, and models with tens of thousands of parts are becoming more common-place. Parasolid supports rendering such large models, including the ability to control memory usage by only rendering areas of interest, or by ignoring small features that may not be relevant in a particular context.	
Application support	The functionality offered by Parasolid encompasses far more than just modelling operations. A major area of functionality is dedicated support for application-specific functionality, such as the ability to store attribute information on modelling entities, comprehensive roll-back and partitioning capabilities, debugging facilities for the developer, and enhanced performance for multi-processor machines.	

Parasolid also offers the developer unprecedented accuracy. Parasolid's default session precision is $1.0e^{-8}$ in a world size of $1.0e^{3}$. This gives an accuracy ratio of $1.0e^{11}$, which is an order of magnitude more accurate than that of any other kernel modeler.

The Parasolid API is C-callable, so that it can be integrated into any C or C++ application code. A binding for C# is also available, allowing you to call the Parasolid API from C# code.

2.2 Parasolid components support

Siemens PLM Software provides comprehensive technical support for the Parasolid library and all related products, via a dedicated team of specialists co-located with the Parasolid development team in Cambridge, UK. The Parasolid Support team are true subject matter experts who are experienced in both using and integrating the toolkits, as well as enabling customers to maximise the full potential that the Parasolid Toolkits offer.

All commercial Parasolid customers under a current maintenance agreement are entitled to a full range of world-class support services at no additional cost. These services include direct email and telephone access to Parasolid subject matter experts as well as the Siemens 24x7 GTAC on-line Support Centre. Siemens Parasolid Support is provided for all platforms, all releases (full releases and patch releases), bug fix resolutions to your reported problems and all cumulative bug-fixes implemented for all customers. Where appropriate and upon request, Siemens can also arrange for face-face or virtual meetings with support personnel.

Siemens takes a proactive and collaborative approach to support through establishing meaningful, long-term relationships with customers. Getting to know you and your company, your products, workflows, functionality requirements, project and technical problem resolution priorities and timelines enables us to provide you with the high standard of support that enables you to deliver maximum value to your own users.

Siemens also understands the frustration that dealing primarily with anonymous or automated helpdesk systems can lead to. Whichever way you chose to contact the Parasolid Support team, you can be assured that there will be a real and identifiable subject matter expert at the other end of the email or the telephone.

Siemens takes great pride in the consistently high scores that our customer support has received in our regular customer surveys and is constantly striving to improve the services offered.

Support is available to all registered Parasolid customers as well as evaluators. Regular update releases are made available to all customers when needed.

For more details please contact Parasolid Support by emailing

ps-support.plm@siemens.com

2.2.4 Fault reporting and debugging tools

In the event you do find a problem with one of Siemens' Parasolid Toolkits, the XML-based debug reporting system is simple and streamlined and provides all the information the Parasolid Support team requires to speedily replicate and investigate the problem you are experiencing.

Additional tools to aid the visual debugging of geometry problems are provided as part of our Parasolid Jumpstart Kit material. Our journaling system also provides a record of all API calls made to the Parasolid Toolkits which assist in the debugging of application problems.

2.2.5 Frequent product releases

Releases of Parasolid Toolkits are frequent, regular and predictable. As a customer, this provides you flexibility with your own release schedules and confidence that you can release at any time with an up-to-date release of Siemens' software.

Full releases of Parasolid containing major functional enhancements are made available once every six months on average. This comprises a complete release of the product and is available both via a secure HTTP site and on DVD. Service packs (patch releases) containing resolutions to customer reported problems and minor functional enhancements are delivered on a frequent and regular cycle.

Parasolid provides full version control which enables you to switch on or off behaviour changes made to the Parasolid kernel at any given release, thereby giving you full control over the functionality you choose to have in your application.

Parasolid incorporates a plug-and-play mechanism that allows you to use any newer version of Parasolid at run time, without rebuilding your application. This can be helpful for testing and deploying bug fixes in a newer Parasolid version with minimal disruption to your application.

Every release of a Parasolid Toolkit is rigorously tested and is guaranteed to have successfully passed our extensive suite of all-platform functional and regression tests. Each nightly build of Parasolid is tested for functional and performance regressions with over 2.5 million tests including all faults ever fixed.The run-time plug-and-play compatibility of the Parasolid Toolkit APIs provides a stable and reliable foundation for the straightforward, worry-free integration of incremental toolkit libraries.

Parasolid toolkits are supported on an extensive range of platforms, which is reviewed regularly as new platforms emerge.

The Siemens PLM Components "Level Playing Field" policy ensures that each Parasolid release is made equally available to both commercial and Siemens internal product groups simultaneously.

2.2.6 Learning material, training and consultancy

The extensive Parasolid standard materials can be supplemented by our instructor-led training and consultancy services. See Section 2.3, “Parasolid Jumpstart Kit”, for more information on these materials. These services are available on request for an additional fee and can be conducted either at your facilities or occasionally at our offices in Cambridge, UK.

- The instructor-led Parasolid training is a comprehensive, hands-on class covering the full breadth of functionality. It features both formal, taught material and interactive exercises which help to reinforce the knowledge gained and ensure you are ready to hit the ground running when applying this to your own development. Shorter, customised classes for users of Parasolid Editor or Parasolid Communicator are also available on request.
- Consultancy services take the form of in-depth implementation advice. A member of our Parasolid Support team works collaboratively with members of your organisation to understand your requirements and application workflows and provides advice and recommendations for an optimal integration of our toolkits into your application environment.

2.2.7 Prospective customers

Prospective customers are allocated a dedicated single-point-of-support contact who is personally available to answer all initial questions and help guide them through their evaluation of the Parasolid Toolkits.

The Parasolid Support team has extensive experience working with companies with a variety of objectives and backgrounds. For example:

- Experienced 3D users transitioning from another geometry kernel technology (in-house, commercial, open)
- Companies using 3rd party software component technology for the first time
- Start-up companies where cash flow is critical
- Companies transitioning from 2D to 3D
- Companies taking advantage of new technology to support new business needs
- Industries as diverse as CAD, CAM, CAE, AEC, CMM, EDA, Machine Tools, Medical and Shipbuilding

In conclusion, Siemens' goal is to support the successful development of customers' Parasolid applications. Our win-win business model means we have a mutual interest in providing assistance which allows you to shorten your time to market. We look forward to working with you.

For more details please contact Parasolid Support by emailing

ps-support.plm@siemens.com

2.3 Parasolid Jumpstart Kit

The Parasolid Jumpstart Kit provides material aimed at helping new users get up to speed self-sufficiently, quickly and straightforwardly with Parasolid. Using the Parasolid Jumpstart Kit as a structured learning aid, users can rapidly become adept at coding with Parasolid. Together with the support and documentation provided, this assists an accelerated implementation, and enables you to get your new products and enhancements to market faster.

The Parasolid Jumpstart Kit is equally well-suited to new customers, existing customers who need to train new employees and to experienced users exploring an unfamiliar area of functionality for the first time.

The material is coherently presented through a single web portal which leads users through the material in a structured manner, helping to make their initial experience with the toolkit a positive and constructive one and breaking down perceived barriers to implementation.

2.3.8 Functional overview

The overview documentation (this manual) provides a comprehensive, high-level summary of the nature and scope of Parasolid functionality. This material serves a valuable role for both development and business audiences when forming an initial understanding of the scope and depth of Parasolid functionality without introducing the specifics of the API. More detailed knowledge can subsequently be assimilated as required using further resources within the Parasolid Jumpstart Kit.

2.3.9 Training presentations

Three structured training presentations are provided to introduce the user to the key Parasolid concepts, functionality and the APIs in three essential subject areas: Parasolid Fundamentals, Modelling Functionality and System Design. These enable the reader to quickly become immersed in the capabilities and concepts that are central to hands-on development with Parasolid.

2.3.10 Example applications

Example Parasolid applications written in both C++ and C# provide a platform for understanding the architecture and integration of a Parasolid application. Consisting of full, commented source code and backed up by further documentation, these applications provide sample integrations of file and memory management, error handling, and display using both Direct-X and Open GL.

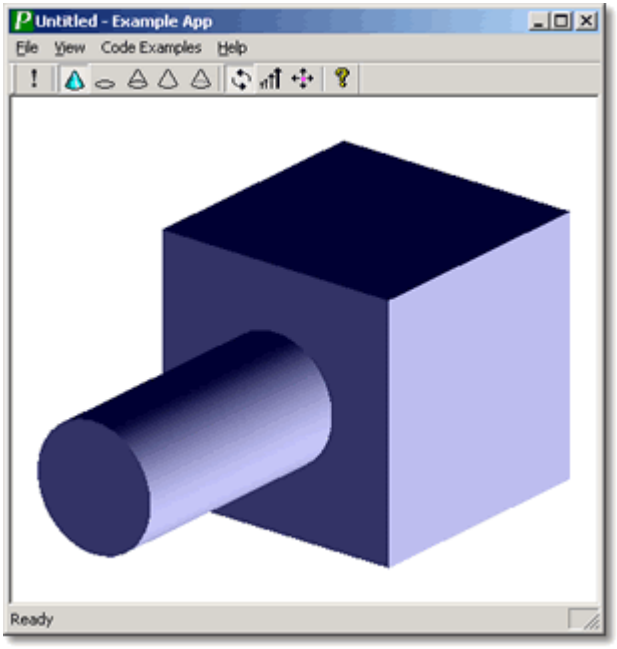


Figure 2–1 The Parasolid Example Application

2.3.11 Getting started

Getting started material leads the user through each of the initial steps involved when constructing a Parasolid application using the C++ Example Application as a contextual example.

2.3.12 Code examples

Working, fully-commented code examples in C++ and C# are provided to assist the user in calling many of the commonly-used Parasolid APIs. You can quickly discover how to call a particular API and learn about its most useful options using these code examples.

The logical grouping by functionality provides a hierarchy which is simple to browse and enables you to quickly locate examples which are relevant to a particular usage scenario. Each code example can be executed and stepped-through in a structured and consistent manner in one of the example applications, and the viewing window allows you to see the results of the API calls at each step.

2.3.13 Workshop.Net and the Parasolid Application Framework

Workshop.Net is a Parasolid-based application supporting model viewing, model analysis, and modelling operations. The Parasolid Application Framework is a collection of widely applicable, re-usable, modular classes and functions.

Both Workshop.Net and the Parasolid Application Framework ship with modular and fully commented source code and are readily extensible for prototyping and accelerated application development.

For more information on Workshop.Net, see Section 2.5, “Workshop.Net”.

2.3.14 Parasolid Documentation

The comprehensive Parasolid Documentation suite is also integrated into the Parasolid Jumpstart Kit. For more information on the documentation, see Section 2.4, “User documentation”.

2.4 User documentation

Extensive user documentation is provided for each of the Parasolid Components. The comprehensive information contained in the documentation is your first point of reference for anything you wish to know about the products (and more).

The user documentation presents often complex subjects in a simple and straightforward style, with frequent use of pictures helping you get directly to the key relevant facts.

Documentation is organised into two principal subject areas:

- Clearly structured API documentation containing all the information you need to know to integrate and program against the interface functions
- Comprehensive reference material describing the full breadth and depth of the functionality for when you need detailed information

Additional documentation covering other subject matters such as high-level overview information, getting started and integration material, new functionality in the latest release, guides on our example applications and the problem reporting process is also provided as part of our far-reaching core documentation sets.

The documentation is structured logically into key books, volumes and chapters - making it simple to rapidly locate the information you need. Full text search capabilities are also included, together with indices and glossaries of key terminology to further enhance the usability experience.

Our documentation sets are provided by default in both HTML and PDF formats, providing flexibility for both on-line and printed media to suit your preferred way of working.

In addition to the core documentation sets, extensive documentation also accompanies the various example code and other example applications provided as part of the Parasolid Jumpstart Kit.

All user documentation is used and reviewed by Customer Support personnel, and Technical Authors receive feedback and requests directly from customers.

2.5 Workshop.Net

Workshop.Net is an extensible Parasolid-based application supporting model viewing, model analysis, and modelling operations. Workshop.Net ships with full source code, and facilitates accelerated application development: you can implement Parasolid workflows in a ready-made and fully interactive environment.

Workshop.Net's source code shows how Parasolid functions may be used in practice.

The code is written in C# and is modular, fully commented and designed for maximum readability and maintainability. The user interface is built with Windows Forms, allowing easy customisation.

Workshop.Net includes a powerful plugin mechanism that lets you integrate your own C# code into the application. Workshop.Net ships with a number of example plugins:

- A plugin that provides functionality to create a primitive body. This plugin allows you to choose the dimensions of the primitive body, its reference direction, and axis. Full source code is shipped with this plugin.
- A plugin that provides functionality to perform boolean operations on general bodies. This plugin allows you to choose a target and tool body on which to perform boolean operations. Full source code is shipped with this plugin.
- A plugin that provides functionality to analyse a body. As well as returning information about a body, it allows you to check the body, perform mass properties calculations on it, and compare it with another body. Full source code is shipped with this plugin.
- A plugin that provides access to the healing functionality provided by Parasolid Bodyshop. In order to use this plugin, you must request an evaluation licence from

Parasolid Support. Existing Parasolid Bodyshop customers qualify for a permanent licence. No source code is shipped with this plugin.

- A suite of plugins that provide translation functionality for the following file formats: CATIA V4, Pro/ENGINEER, STEP, and IGES. In order to use each translator, you must request an evaluation licence from Parasolid Support. Existing Translator customers qualify for permanent licences of the translators they are licenced to use. No source code is shipped with these plugins.

To extend the application, you can create self-contained libraries (plugins) which communicate with the application via events, or you can adjust and augment the existing code.

2.5.15 Source code: general-purpose functionality

The source code provided with Workshop.Net includes implementations of many widely applicable, re-usable, modular classes and functions including the following features:

- Viewing models and underlying geometry
  - Facet-based, line-based or combined views
  - Simultaneous views of several models in separate or shared windows
  - Mouse interaction - pan, zoom, zoom-to-box, fit-to-window, rotate
  - Standard views and current view matrix
  - Full-screen anti-aliasing for high-quality screenshots
- Best practices for implementing visualisation with Parasolid
- Navigating models
- Parasolid session management
- Plugin (extensibility) mechanism
- Application settings - convenient storage of user preferences in the Windows registry

2.5.16 Source code: model analysis

The Workshop.Net source code provides further functionality which is particularly useful for analysis of Parasolid models.

- Entity selection and highlighting
- Easy navigation of model structures (topology and geometry) via tree views
- Model statistics
- Configurable Parasolid rendering and faceting
- Model validity checking
- Mass property calculations
- Model comparison
- Interactive model positioning via mouse drag
- Entity details such as orientation and geometry standard forms

2.5.17 Documentation

Workshop.Net is supplied with comprehensive documentation that includes:

- A *User's Guide* that describes the functionality available in the application
- A *Source Code Overview* that describes the structure of the source code, the core components of the applications, and the plugins that are provided
- A *Plugin Creation Guide* that explains how you can develop your own plugins in order to add functionality to, or prototype your own code in, the basic application.

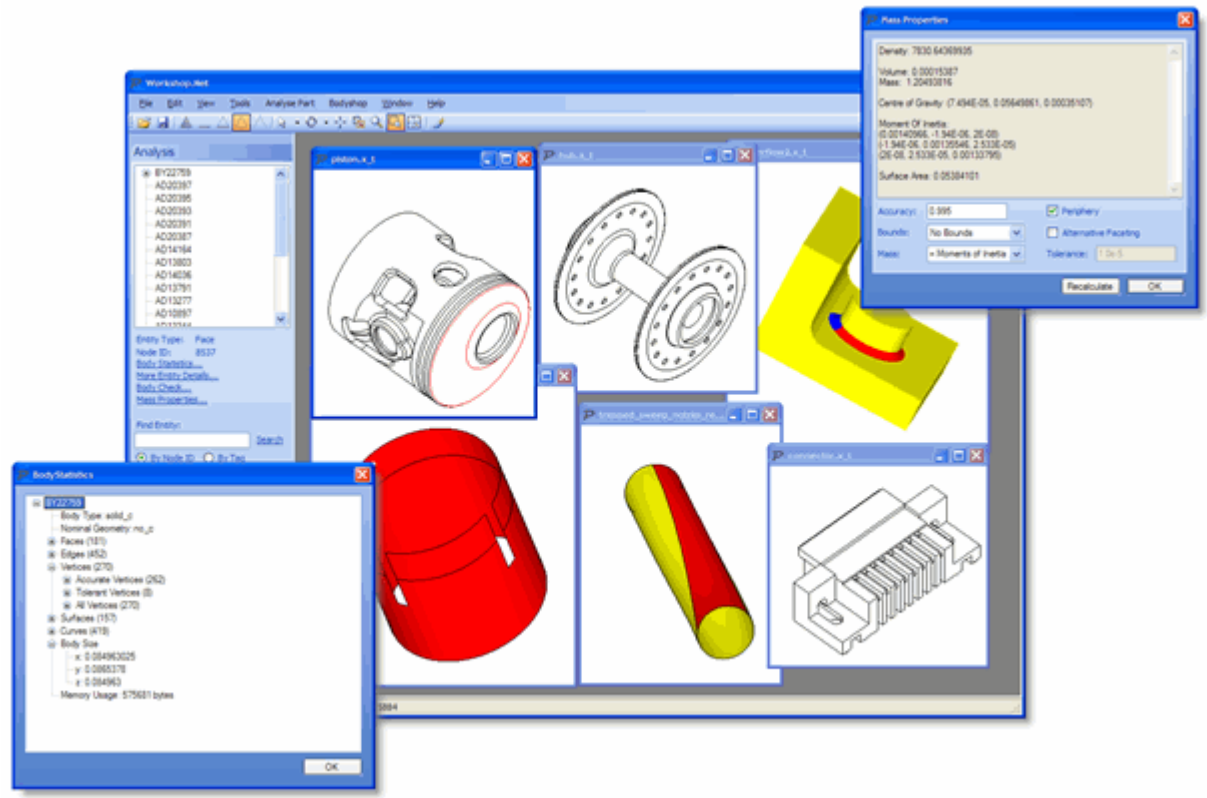


Figure 2–2 Parasolid Workshop.Net

Model Structure 3

3.1 Introduction

This chapter describes the major modelling entities that Parasolid uses to create its model structure, and how they are combined so as to construct a Parasolid model. The entities available fall into three broad categories: topological, geometric and associated data. Their general relationships are shown in the following diagram:

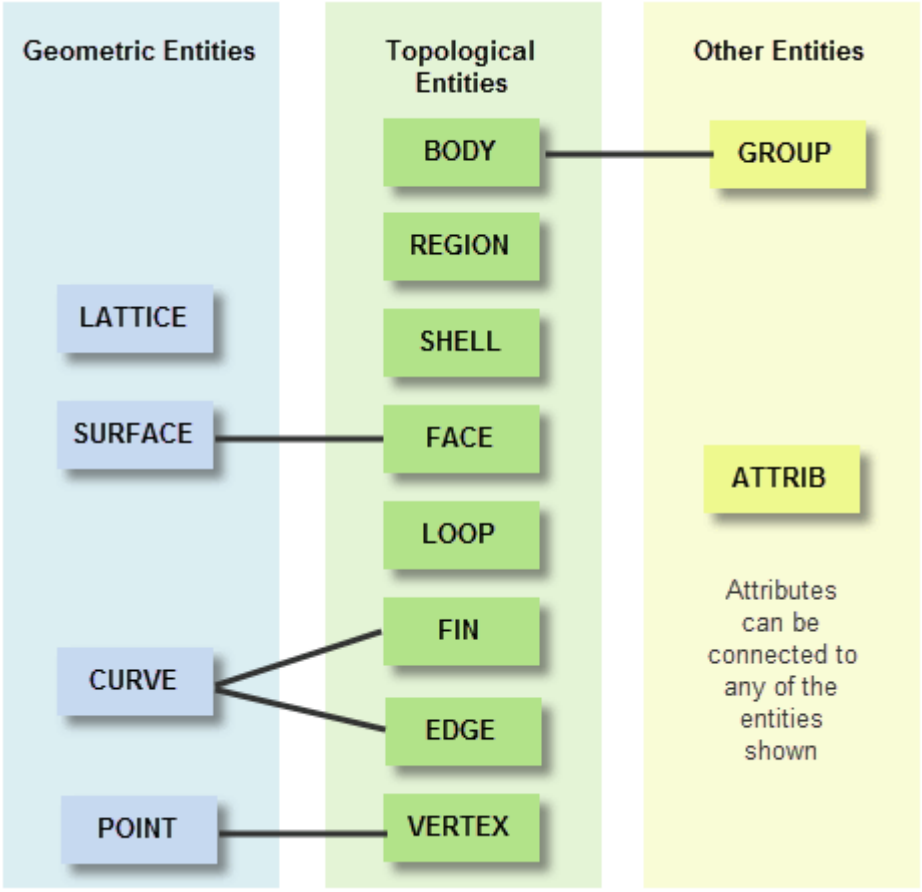


Figure 3–1 *Relationships between Parasolid entities*

- The major entities shown in *Figure 3–1* are described in Section 3.2, “Topological entities”, Section 3.3, “Geometric entities” and Section 3.4, “Other entities”, respectively.
- Section 3.5, “Assemblies and instances”, explains how you can combine topological information to create more complex structures.
- Section 3.6, “Accuracy of Parasolid models”, discusses the default precision that Parasolid uses for its models, and the support that it offers for importing and modelling with data from other, less accurate modelers.
- Section 3.7, “More about bodies”, explains more about the different types of body that Parasolid supports, and the level of checking that is performed on bodies.

3.1.1 Identifying entities

It is essential for both Parasolid and for your application to be able to identify different entities in a model. Parasolid provides two methods for doing this:

- **Tags** are used to identify particular entities within a Parasolid session. Every entity in a model has an associated tag that is created when the entity is created or enquired about (see Chapter 13, “Enquiring Model Data and Identifying Details”). Each tag is unique within a session. Tags are held in integer variables which Parasolid functions use as pointers to entities.
- **Identifiers** (IDs) can be used to identify particular entities between Parasolid sessions. Parasolid automatically attaches unique IDs to entities in a model because Parasolid does not preserve tags between different sessions: when the same part is loaded into two different sessions, its entities may have different tags.

For more information about Parasolid sessions, see Section 15.3, “Partitions and roll-back” and, in particular, Section 15.3.1, “Concepts”.

3.2 Topological entities

Topological entities comprise all the entities that are used to construct the structure or skeleton of a model.

3.2.1 Bodies

Bodies are fundamental to modelling with Parasolid. A body is composed of one or more connected entities, or *components*. Collections of related bodies that share some physical

aspects (for instance, different representations of the same part), can be represented as *compound bodies*. Bodies can contain the topological entities shown in *Figure 3–2*

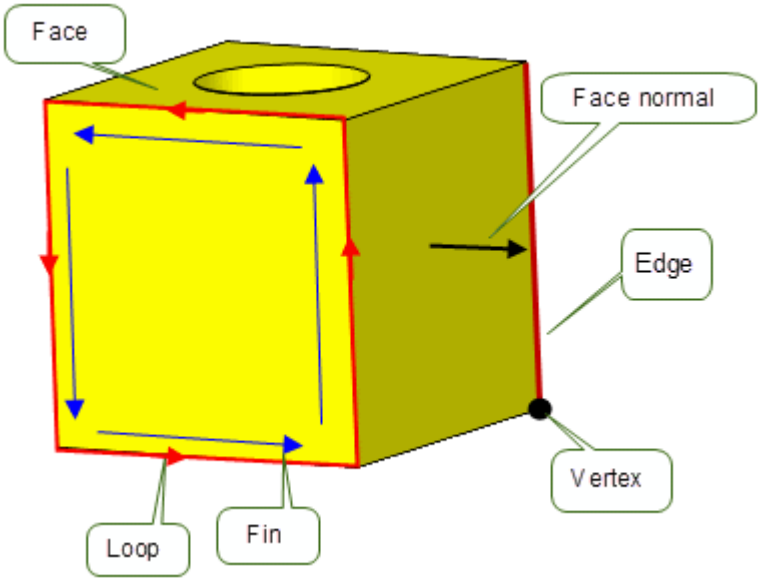


Figure 3–2 *Topological entities in a body*

Topology	Description
Face	A *face* is a bounded subset of a surface, whose boundary is a collection of zero or more loops. A face with zero loops forms a closed entity, such as a full spherical face.
Loop	A *loop* is a connected component of a face boundary. A loop can have: ■ an ordered ring of distinct fins ■ a set of vertices
Fin	A *fin* represents the oriented use of an edge by a loop.
Edge	An *edge* is a bounded piece of a single curve. Its boundary is a collection of zero, one or two vertices.
Vertex	A *vertex* represents a point in space. A vertex has a single point, which may be null

Bodies also contain regions and shells, as described in Section 3.2.3 and Section 3.2.4 respectively.

For a fully defined, valid model, every face, edge and vertex in a body requires an attached geometric entity: a surface, curve or point.

More information about bodies is given in Section 3.7, “More about bodies”.

3.2.2 Facet bodies

A facet body is one with Parasolid topology that references geometric data composed of **mesh** and **polyline** data (often referred to collectively as facet data), rather than classical Parasolid surface and curve geometry. A mesh is a surface subtype which encapsulates an application’s mesh data for one or more Parasolid face. A polyline is a curve subtype and describes a connected chain of linear segments. Parasolid can read in external mesh data in one of four forms; index, strip, fan, and vector, which are shown in *Figure* .

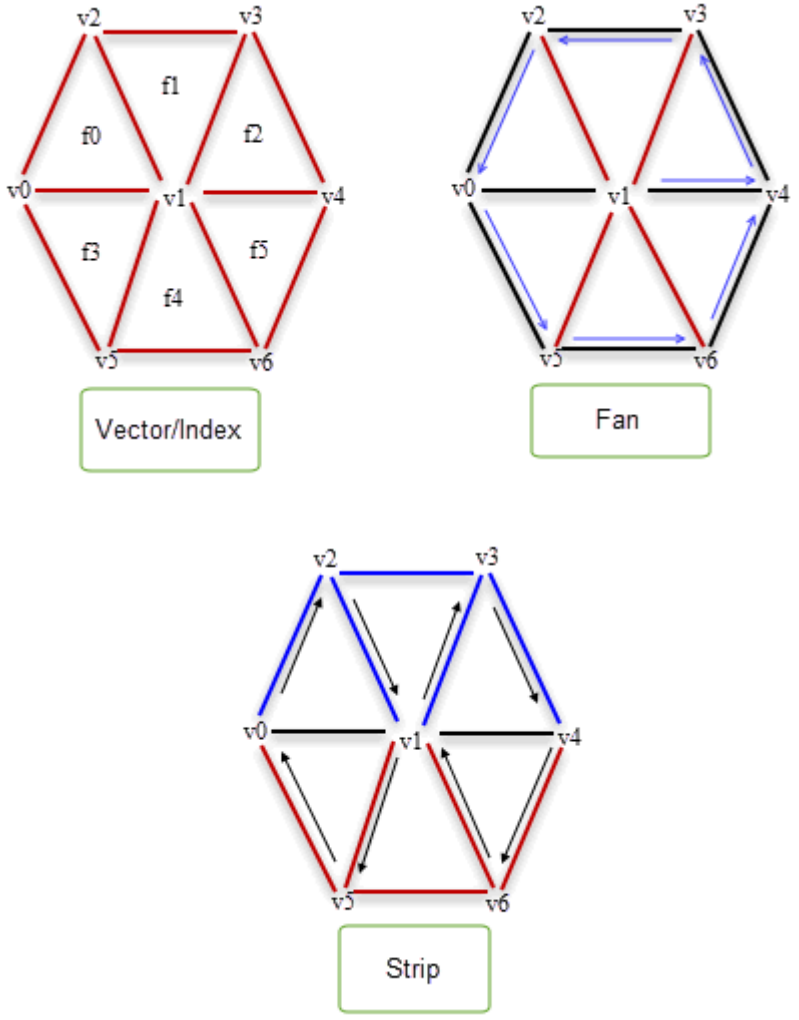


Figure 3–3 *Types of meshes that are supported*

Facet bodies can contain the mesh topologies shown in *Figure 3–4*.

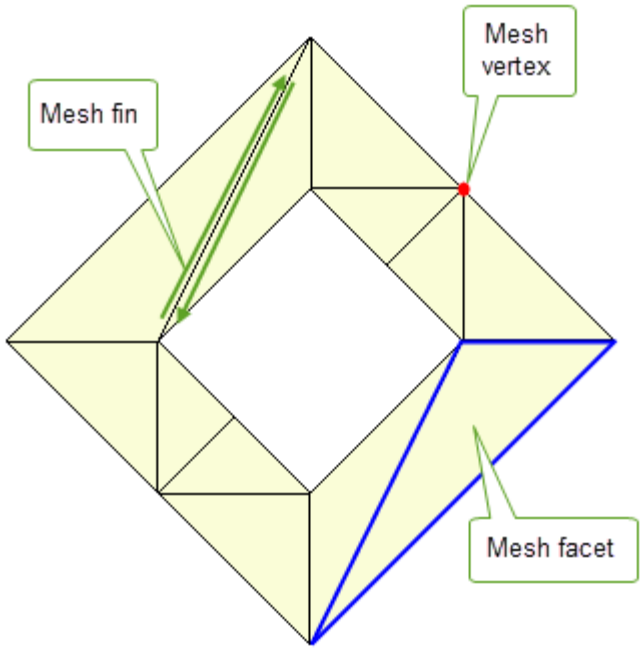


Figure 3–4 *Mesh topologies on a facet body*

The relationship between mesh topologies and Parasolid entities are shown in *Figure 3–5*

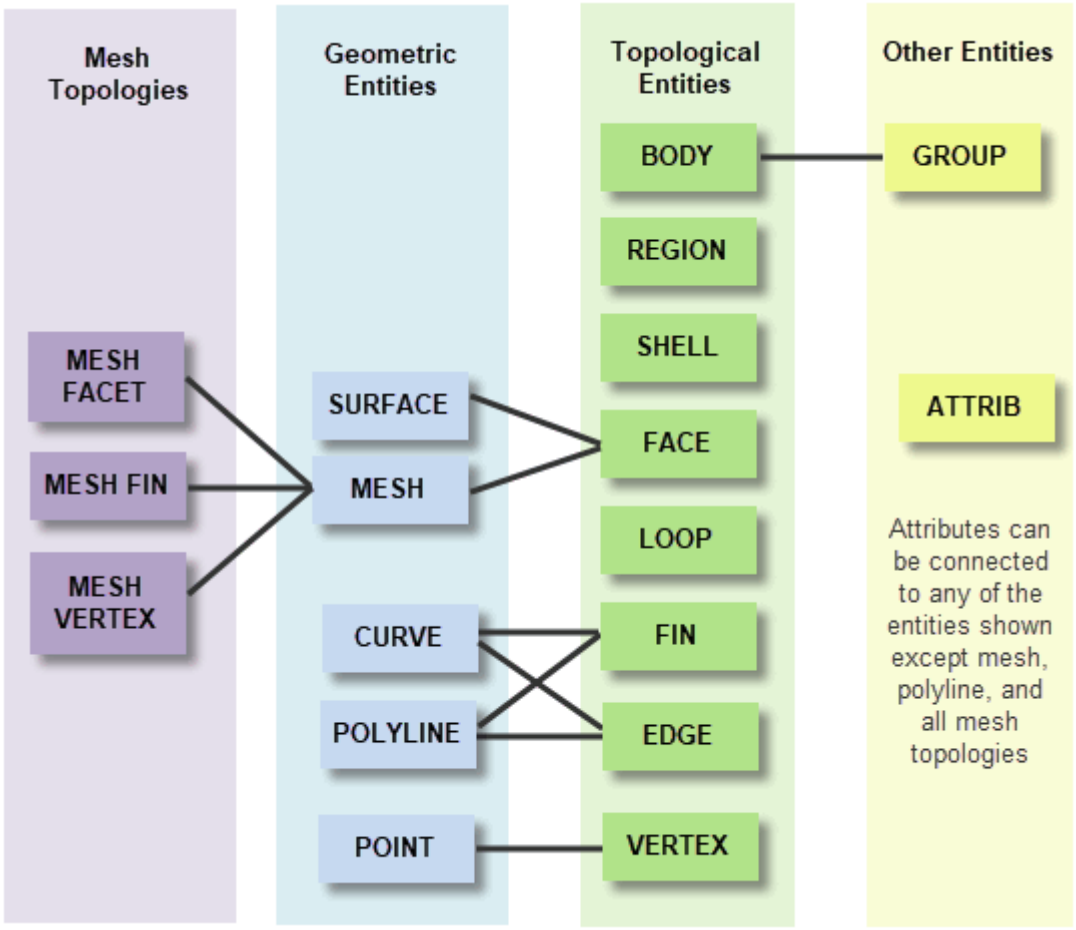


Figure 3–5 *Relationship between mesh topologies and Parasolid entities*

See Chapter 8, “Convergent Modeling” for more information on Convergent Modeling.

3.2.3 Regions

A *region* is a connected subset of 3-dimensional space bounded by a collection of vertices, edges and oriented faces. Regions are either *solid* (contain material) or *void* (empty).

Bodies always have an infinite void region, which you can think of as all the space outside of the body itself. The sum of all regions in a body comprises the whole of 3D space.

For example, *Figure 3–6* shows a hollow cube that contains three regions:

- The void region in the center of the cube.
- The solid region occupied by the material of the cube.
- The (infinite) void region outside the cube.

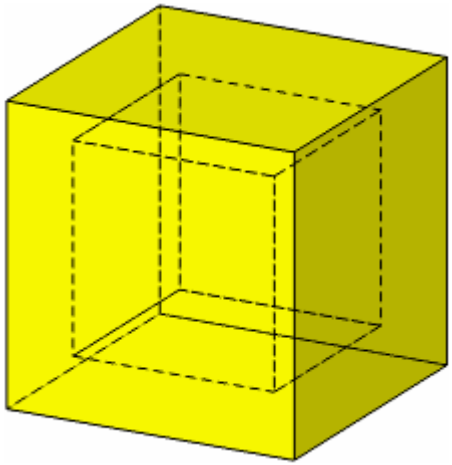


Figure 3–6 *Hollow cube consisting of three regions*

Regions contain a set of shells that define the topological structure of the parts in the body.

Note: A face is a two-dimensional analogy of a region. An edge is a one-dimensional analogy of a region. A vertex is the zero-dimensional analogy of a region.

3.2.4 Shells

A *shell* is a connected collection of oriented faces (each face used by the shell on one or both sides of the face) and edges. The shells in a region do not overlap or share faces, edges or vertices.

Note: A loop is the two-dimensional analogy of a shell.

3.2.5 Face normals

In a solid body, the normals of each face must always point away from the solid region:

- For an outer shell, the normals point away from each other.
- For an inner shell (i.e. a void inside a solid) the face normals point towards each other.

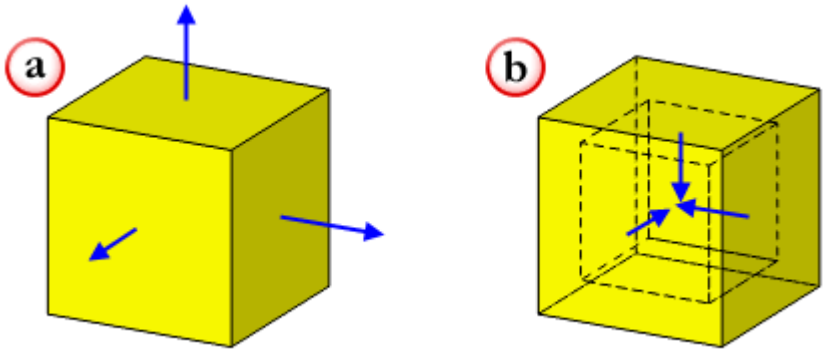


Figure 3–7 *Face normals of (a) solid body and (b) an inner shell*

Sometimes, a modelling operation may create a *negative body*, that is, a body whose face normals point the wrong way. Such bodies can easily be negated.

For sheet bodies, face normals must also point away from the body. The face normals in a sheet body must be consistent, as shown in *Figure 3–8*.

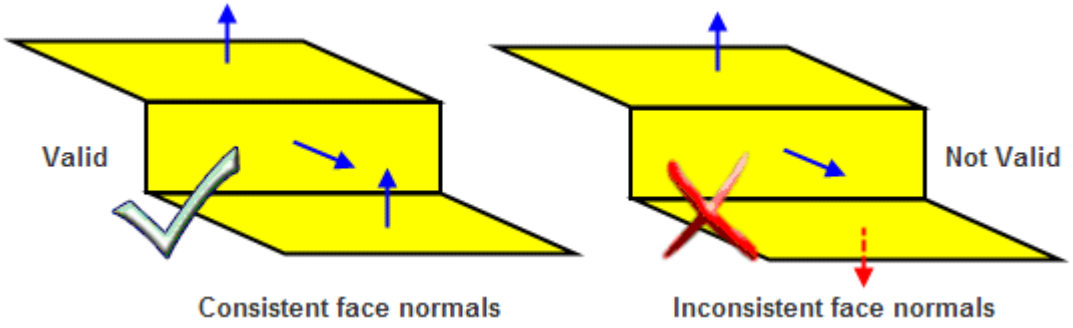


Figure 3–8 *Face normals of sheet bodies*

3.2.6 Direction of loops, fins, edges

Loops, fins and edges have directions, and Parasolid defines the following conventions for their relationships to face normals and to each other:

- The *forward direction* of a loop has the face on the left of the loop, when viewing the face down the face normal, as shown in *Figure 3–9*.

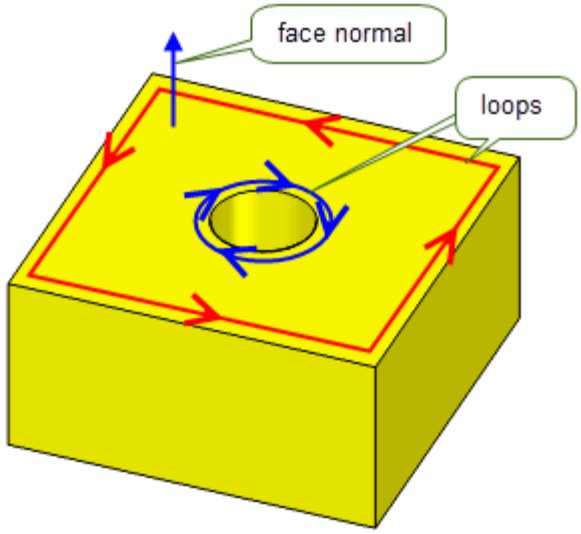


Figure 3–9 *Loop directions of the top face of a cube with a cylindrical hole*

The direction of the loop around a face determines the direction of the edges and fins of that face:

- A loop represents one boundary of a face as a closed set of fins, therefore the direction of the fin is the same as that of the loop that contains the fin.
- An edge, which (on a manifold solid) has a left and right fin, takes its direction from the left fin (which takes its direction from its loop)

3.3 Geometric entities

Geometries entities are used to specify the geometric shape of a body. These geometries are referred to as *principal geometry.* In a completely defined part, geometric entities are attached to topological entities according to the relationships shown in *Figure 3–1*.

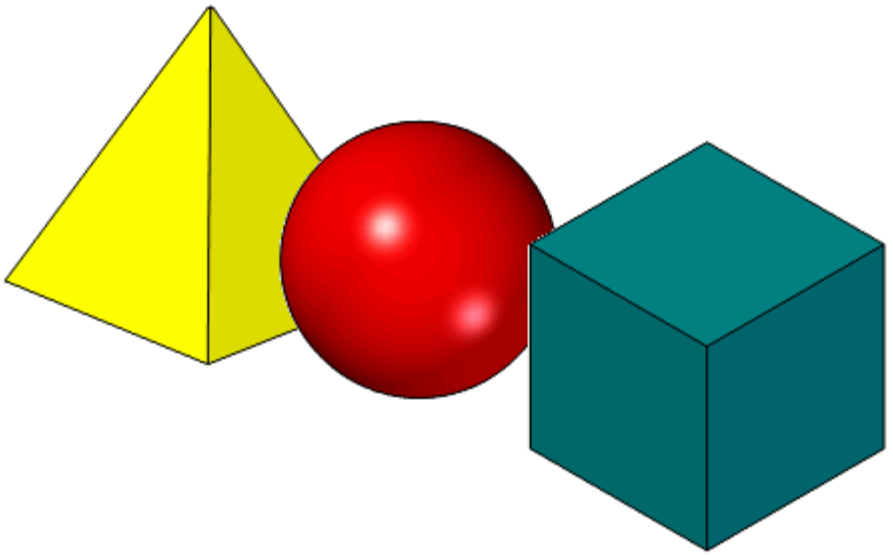


Figure 3–10 *Examples of 3D geometric shapes*

You can also attach geometry directly to a body to form *construction geometry*, as a means of keeping relevant entities with the body. For example, you might want to attach a point to a body that represents its center of gravity.

Parasolid also supports *orphan geometry*, which is not attached to any topological entity. This is useful if you want to create geometry prior to using it in a body, for example, to create a surface before incorporating it into a body.

There are four main classes of geometric entity:

Geometry	Description
Lattice	*Lattices* are a geometry type that can be created, enquired of rendered and attached to a part as construction geometry.
Surface	*Surfaces* are principally attached to faces. Normally every face has an attached surface, but it may be detached temporarily as the model is being built or modified. There are several subtypes of surface, such as plane, sphere cone and mesh
Curve	*Curves* are principally attached to edges or fins of the model. There are several subtypes of curve, such as circle, ellipse line, and polyline..
Point	*Points* are principally attached to vertices. All points are Cartesian

Only entities of surface, curve and point classes may also be attached directly to bodies as construction geometry.

In order to reduce the overall size of a model, geometry can be shared both within a single body, and across collections of bodies. Geometry can be shared in the following ways:

- Edges or fins with a common direction can share a common curve
- Faces can share a common surface

3.4 Other entities

Parasolid supports a number of other entities so that you can manipulate and attach additional data to a model. The most useful ones are:

- *Groups*, which let you group together arbitrary collections of entities within a body. See Section 15.4, “Groups”, for more information.
- *Attributes*, which are data structures containing arbitrary information that can be attached to any other Parasolid entity. See Section 15.2, “Attaching attribute information”, for more information.
- *Transforms*, or transformations, which are used to express geometric operations such translation, reflection, rotation or scale. You can use transforms to specify certain modelling and rendering operations, such as creating assemblies. See Section 3.5, “Assemblies and instances”.

3.5 Assemblies and instances

You can create more complex models in Parasolid by combining any number of bodies into larger *assemblies* that contain information about how the bodies contained in the assembly are structured. *Figure 3–11* shows a trivial example in which a toy car assembly is created by combining a body part with a wheel assembly, which is itself created from a wheel part and an axle part.

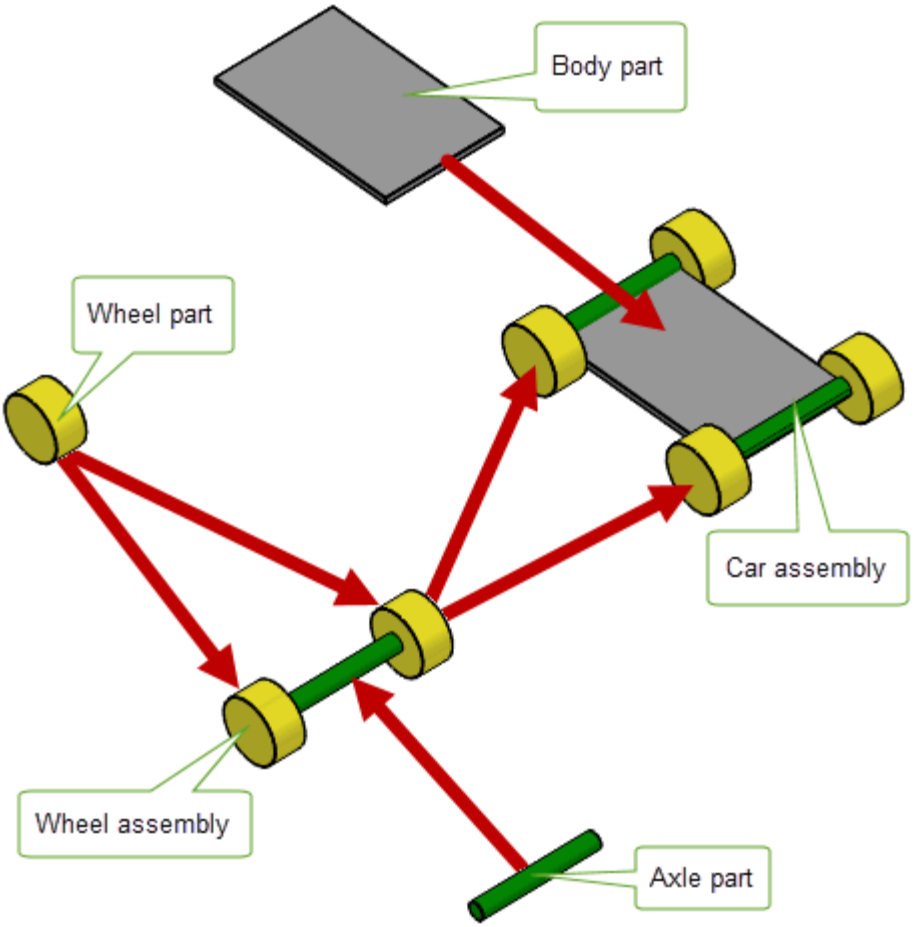


Figure 3–11 *Combining different bodies into more complex assemblies*

Each component in an assembly is an *instance* of a part:

- The car assembly contains an instance of the body part, and two instances of the wheel assembly
- The wheel assembly contains an instance of the axle part, and two instances of the wheel part.

An assembly is therefore just a collection of instances, where each instance is a pointer to a part used in the assembly that references:

- The *part* itself (either a body or another assembly).
- A *transform* that positions the part in the coordinate frame of the assembly.

To support reuse of modelling data, assemblies can contain any number of instances of the same part. You can also attach construction geometry to an assembly.

3.5.1 Restrictions on assemblies

Assemblies in Parasolid can only convey structural information about the components in the assembly. In particular, they do not in themselves constitute a complete assembly management system.

Assembly information in Parasolid is subject to a number of basic restrictions:

- Instances cannot exist outside an assembly.
- Instances cannot reference a null part.
- Instances must reference a **rigid** transform: scale, shear, reflection, and perspective are not supported within the context of assemblies.
- Assemblies cannot directly or indirectly instance themselves. Cyclic trees of assemblies and parts, as shown in *Figure 3–12*, are not allowed.

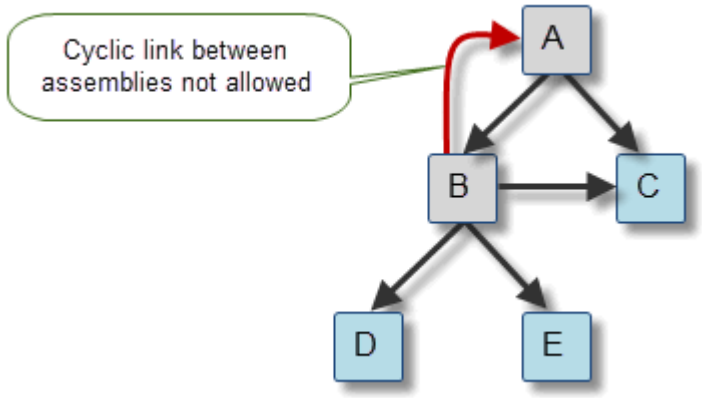


Figure 3–12 *Cyclic and acyclic trees of assemblies and parts*

3.6 Accuracy of Parasolid models

A Parasolid model is very precise. All calculations are performed to fixed accuracies called the *session precision* and *session angle precision*.

Precision	Description
Session precision	The linear precision of the modeler. Parasolid session precision is $1.0e^{-8}$ units. Distances less than this value are treated as zero and distances that differ by no more than this value are treated as equal.
Session angle precision	The smallest angle (in radians) that is treated as different from zero. Parasolid session angle precision is $1.0e^{-11}$ units. Angles less than this value are treated as zero and angles that differ by no more than this value are treated as equal.

To handle precision correctly, all parts of a body must lie inside the Parasolid *size box*. This is a box, 1000 units on each side and centered at the origin, that represents the whole of model

space. Typically, the default unit in your application is set to one meter, giving 1 kilometer as the maximum size, in any one direction, that can be modeled.

Parasolid also supports the use of large transformations in situations where a body lies within the size box but can be positioned by a large transform outside the size box. This can be useful for example in designing factories when deciding where to position objects (such as machinery) around the factory. Each component of the factory is “small” but the factory itself may extend to a few kilometers in size.

3.6.1 Tolerant Modelling

Parts created by other modelers can be imported into Parasolid and used in modelling operations. However, such parts will almost certainly have been created in a modeler that uses a lower precision. Once imported, they may contain inconsistencies between their topological and geometric data: edges which, topologically, should meet at a common point have attached geometric data that does not intersect.

To enable Parasolid to model successfully with imported parts, you can set a *local precision* on the edges and vertices of any externally created part. The use of such local precisions is known as *Tolerant Modelling*.

When using Tolerant Modelling, think of edges as tubes and vertices as spheres. If you lower the precision, the tubes become thicker and the spheres become larger. To use Tolerant Modelling effectively, you need to examine the topology for each part and, where there is inconsistency in the data, lower the local precision until the geometric data intersects.

Edges and vertices which don’t have a local precision set are considered *exact* by Parasolid, and have a precision of one half of the session precision.

Note: Tolerant Modelling is intrinsic to Parasolid, and its use spreads far wider than data import. Although the vast majority of calculations are performed to very precise fixed accuracies, so as to maintain session precision and consistency between topological and geometric data, many modelling operations (for instance, many involving B-surfaces) use tolerances (to a user-supplied precision) to allow operations to succeed where an accurate solution is not possible. All Parasolid operations work on both accurate and tolerant models.

For a typical example of how Tolerant Modelling is used within Parasolid, see Section 11.2, “Trimmed surface import”.

3.6.2 Nominal geometry

Nominal geometry is a mechanism whereby edges that are locally tolerant may also reference a notional accurate curve.

Any edge which has local precision can have nominal geometry that defines the Parasolid body attached to its fins. These nominal curves must all lie within a *precision pipe* whose diameter is the edge’s local precision. Exact edges cannot have nominal geometry attached, because to do so would be meaningless.

3.7 More about bodies

Bodies in Parasolid are divided into the following broad categories:

- Manifold bodies. See Section 3.7.1.
- General bodies. See Section 3.7.2.

Parasolid also provides functionality to check whether a body is valid. This is described in Section 3.7.3.

3.7.1 Manifold bodies

A manifold body is, broadly speaking, any body that can exist in the real world or could be manufactured. The following manifold body types are supported by Parasolid:

Body type	Description
Acorn	This is the simplest type of body: a body of zero dimensions comprising one or more points in space. An acorn body has one infinite void region, containing any number of shells, each of which contains only a single vertex. A *minimum body* is a special case of an acorn body that must consist of a single void region with a single shell consisting of a single acorn vertex.
Wire	A wire body is one step up from a minimum body: a topologically one-dimensional body. Each component in a wire body is a set of connected edges: ■ an *open* component in a wire body has two ends ■ a *closed* component in a wire body has no ends A wire body has one infinite void region containing any number of shells, each of which contains a single set of connected edges.
Sheet	A sheet body is topologically two-dimensional. Each component in a sheet body is either *open* (e.g. a bounded plane) or *closed* (e.g. a hollow sphere or torus whose walls have zero thickness). A sheet body that contains only open components has one infinite void region containing any number of shells, each of which contains a single set of connected faces. For each closed component, there is an extra bounded void region representing the interior of the closed component. Parasolid supports both single-faced and multiple-faced sheet bodies.
Solid	A solid body is three-dimensional and occupies a finite volume. The volume of each component in a solid body is continuous. Each solid component has a continuous bound volume. A solid body has one infinite void region, one solid region for each continuous solid volume, and one void region for each bounded continuous void volume.

You might frequently progress through all of the manifold body types in the process of making a solid. Beginning with a minimal body, you can turn it into a wire by scribing lines onto it. Closing the profile turns it into a sheet, which needs a new surface. You can then sweep or swing the sheet to form a solid.

Any type of manifold body may be disjoint, that is, comprising several disconnected pieces. Using a single body to represent disjoint components in this way is a useful method of tracking data in a feature modeler. For example, *Figure 3–13* shows the result of subtracting a cylinder from a block.

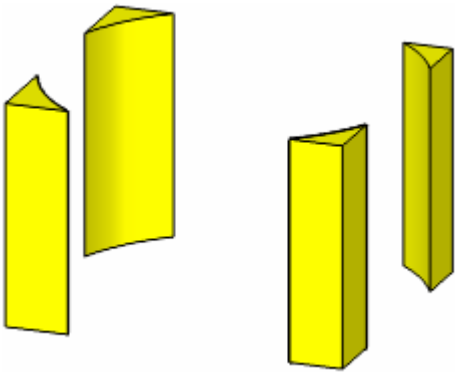


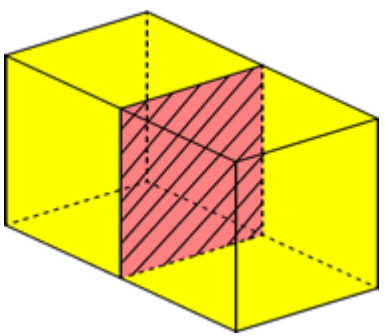
Figure 3–13 *Manifold body consisting of four disconnected pieces*

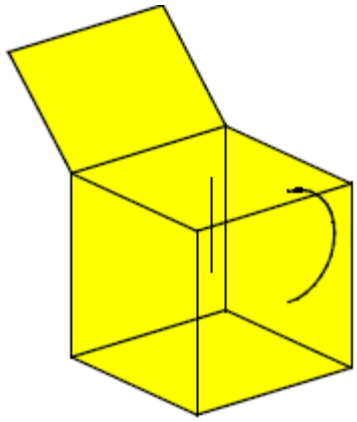
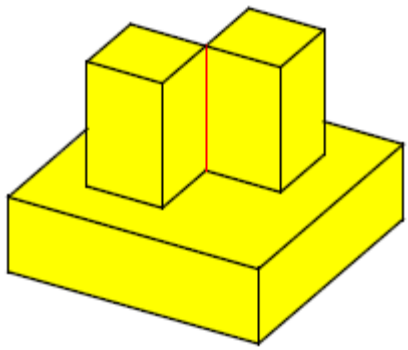
Note: The body in *Figure 3–13* could be represented as either a single disjoint solid or four separate solids.

3.7.2 General bodies

A general body is a collection of entities (faces, edges and vertices) and connected three dimensional regions into which space is divided by the entities. Each region is either solid or void, indicating whether or not it represents material. General bodies differ from manifold bodies in that they usually cannot exist in the real world.

General bodies can be any of the following:

Body type	Usage	Example
Bodies with internal partitions	You can use internal partition faces to subdivide solids for use in, for example, cellular modelling applications. Your application can also model bodies of mixed material using this body type.	

Body type	Usage	Example
Bodies of mixed dimensions	These let you create idealized representations of bodies. For example, a cooling hole in a body could be represented by a wire edge with no thickness.	
Non-manifold bodies	Non-manifold bodies can sometimes occur at an intermediate stage during a modelling operation, such as a boolean. In this example the edge between the two bosses is non-manifold: there are four faces meeting at this edge.	

3.7.3 Checking the validity of a body

When creating models, it is important to ensure that the model remains valid throughout (or, at least, at significant points in the design process). Parasolid supports functionality to let you check whether or not a body is valid, in order to:

- help you to debug your application code
- validate the results of modelling operations
- help you to create valid models from imported data
- help your users to repair invalid models

Parasolid provides two methods for checking whether or not a body is valid:

Type of check	Description
Full body checking	When a full body check is performed, every entity in the body is checked. Full body checking is called using a specific function call. By default, Parasolid performs all the checks appropriate to a given entity. However, you can control which checks are applied by supplying relevant options. Different checks are performed in groups and in a certain sequence, where each group of checks only proceeds if the previous checks have all passed; therefore, checks which rely on the results of previous checks are omitted if these have failed.
Local checking	Local checking only performs checks on entities that are affected by specific local operations, to test whether a body has been invalidated by the operation. Local checking is switched on using an option in the local operation call itself. If the body was already invalid before the operation, in a region not affected by the operation, local checking does not guaranteed to identify the body as invalid. When Parasolid performs local operations with checking off, only relevant simple checks are performed on the integrity of the body after the operation. This is useful if the resulting body is an intermediate stage which you expect to be invalid.

3.7.3.1 What gets checked

In the case of full body checking, Parasolid offers options to let you control exactly what gets checked. You can choose to check for problems such as:

- Inconsistencies in topology or geometry
- Self-intersecting faces,
- Corrupt data
- Violations of the size box

If local checking is on for a given function call, a standard set of checks are performed, as follows:

- Self-intersecting faces are identified
- Pairs of affected faces are selected and checked in turn to ensure that they intersect correctly
- Shells are checked, to determine if the body needs negating, and negations are performed as necessary.

3.7.3.2 When to use checking

Checking can be a time consuming operation. For instance, when local checking is on, each affected face is checked to ensure it is consistent with itself and with every other face in the body. Despite this, local checking is usually much more efficient than full body checking, especially if the local operation only affects a small proportion of the body's entities.

To save time, you can turn local checking off to when you are performing a sequence of local operations, or if you want one of the intermediate operations to create an invalid body. However, you should perform a full body check on the final body and, if this indicates an invalidity, you should roll back to a point before the appropriate local operation and try a different approach.

Booleans and Related Functionality

4

4.1 Introduction

Parasolid offers unparalleled boolean capabilities, allowing you to take sets of modelling entities and compare or combine them in a wide variety of ways. This chapter describes Parasolid's powerful boolean functionality, together with related functionality that either uses Parasolid's boolean capabilities or is similar to it.

- Section 4.2, "Booleans", describes the boolean functionality that Parasolid supports, including the basic functionality, some of the operations that are possible, and the low-level tools that are available for you to implement more specialized functionality.
- Section 4.3, "Instancing", describes functionality that uses boolean operations to provide a fast method of creating patterns on a model (for example, a pattern of bosses or a ventilation grill).
- Section 4.4, "Patterning", describes functionality that is related to instancing, but does not itself use boolean operations to create results. Patterning provides a good complement to instancing, and may be used in situations where instancing is not the best solution.
- Section 4.5, "Sectioning", describes Parasolid's support for dividing a body into different sections using surfaces, faces or sheets.
- Section 4.6, "Clash detection", describes the functionality that Parasolid provides for analysing two bodies and providing information about whether they clash with (i.e. overlap) each other.

4.2 Booleans

Boolean operations are a fundamental type of operation in Parasolid. Not only is boolean functionality available to you, the programmer, but it is used extensively by Parasolid itself. This section describes the boolean functionality that Parasolid supports:

- Section 4.2.1, "Overview of booleans", gives you a broad overview of the boolean process, describing the operations that are possible and the different ways that you can apply them.
- Section 4.2.2, "Common options and uses", explains some of the many options available to you when adding boolean functionality to your application, and the uses to which they can be put.
- Section 4.2.3, "Boolean tools", describes a number of tools that let you add your own boolean functionality to your application. This functionality can be useful if you have very specialized needs, or do not want to use the more generalized tools described in earlier sections.

4.2.1 Overview of booleans

Boolean operations involve the combination of two or more bodies so as to create a new result.

In Parasolid, boolean operations work on **target** bodies and **tool** bodies. A target is the body that you start with, the body that is modified by the operation, and the body that is returned by the operation; it is the body that persists throughout the whole of the boolean operation. The tool (or tools, since there can be several of them for some types of boolean) are the bodies that do the modifying. They are transient bodies that are destroyed at the end of the boolean operation.

Boolean operations in Parasolid are analogous to mathematical set operations. There are three basic boolean operations, as follows:

- A target and tool can be **united**, so that the result incorporates all the entities of the tool.
- A tool can be **subtracted** from a target, so that the result consists of just that material that didn't also form part of the tool
- A target and tool can be **intersected**, such that the only remaining material in the result is that material which was present in both the target and the tool prior to the operation.

Boolean operations can be performed on solid, sheet, and wire bodies (with the exception that wire bodies cannot be intersected). They can be performed using either manifold or general bodies (see Section 3.7, "More about bodies"), and the results may themselves be either manifold or general.

These operations are illustrated in *Figure 4–1*.

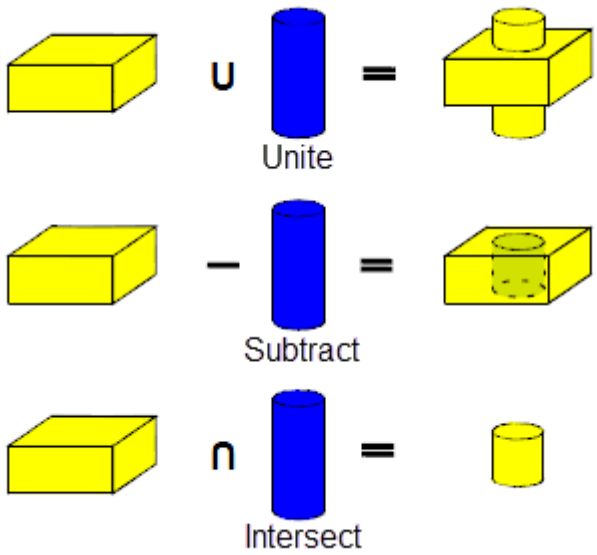


Figure 4–1 *Basic boolean operations*

There is always a single target in any boolean operation. As already mentioned, there can be many tools, in which case Parasolid unites them before the boolean operation proceeds, treating them as a single body from that point on.

Parasolid offers two broad categories of boolean operation: **global** booleans and **local** booleans. These are discussed in the following sections.

4.2.1.1 Global booleans

Global booleans perform boolean operations on whole bodies, taking a single target body, and multiple tool bodies, and applying the specified boolean operation to the target. Because the operation is performed at the body level (by comparing all pairs of faces in the target and tool), the resulting body is guaranteed to be topologically consistent. Global booleans, because of the nature of the face pair comparison, can be computationally very expensive.

4.2.1.2 Local booleans

Local booleans (sometimes known as partial booleans), tend to be very quick to perform compared to global booleans. Rather than working at the body level, local booleans use a set of faces from a target body, and a set of faces from a single tool body. You specify these faces yourself, and only the faces you specify are used in the boolean operation. Restricting the scope of the boolean operation in this way can drastically improve performance.

4.2.2 Common options and uses

Parasolid's boolean functionality supports a wide range of options that you can use to create a huge number of effects. Most of these options can be used with both global and local booleans, although some only apply to one or the other. This section describes some of the most common options.

4.2.2.1 Improving performance by matching coincident regions

Very often, there are regions on the supplied target and tools that are intended to be coincident, or **matched**. For some configurations (such as slightly mis-aligned geometry and topology), it is non-trivial for Parasolid to try and evaluate matched regions itself, and the success of this cannot always be guaranteed, leading to slow performance or failed boolean operations. In order to aid or enhance the result of a boolean operation, you can tell Parasolid which regions are to be matched. This is a flexible process: you can specify faces that should be matched, or edges, or a combination of both. *Figure 4–2* shows some examples.

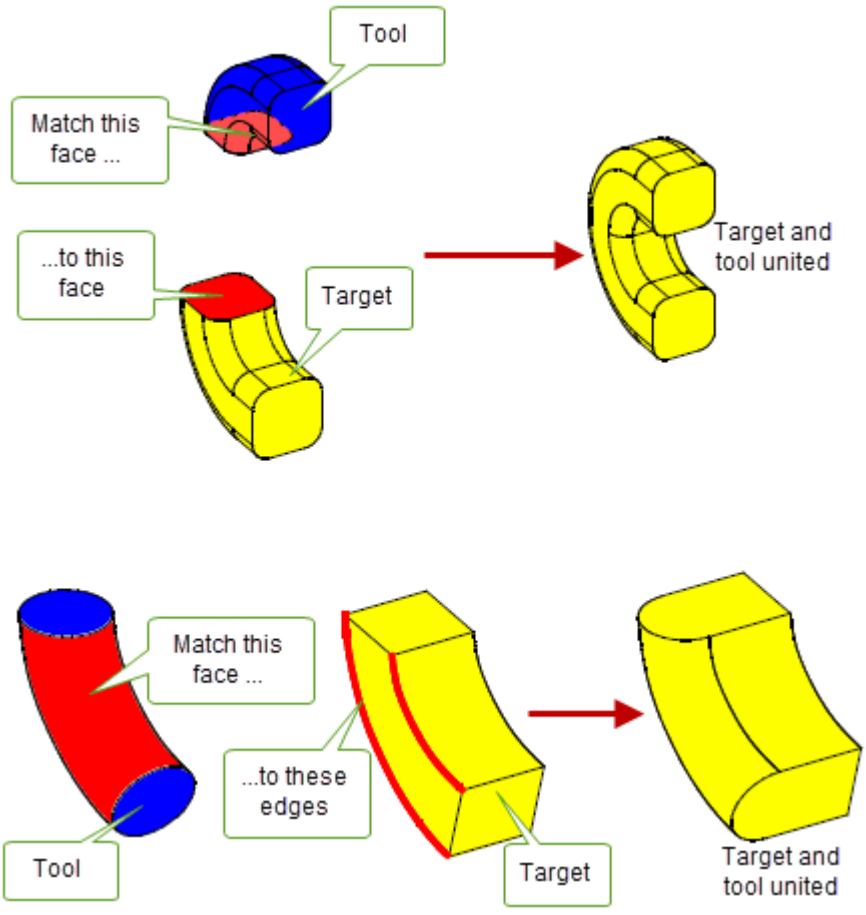


Figure 4–2 *Specifying matched regions to improve boolean operations*

4.2.2.2 Enclosing a solid region with a sheet body

You can enclose solid regions or add solid regions to a solid target by uniting it with a sheet tool. By choosing options to specify the resulting type of body, Parasolid ensures that the resulting body remains manifold.

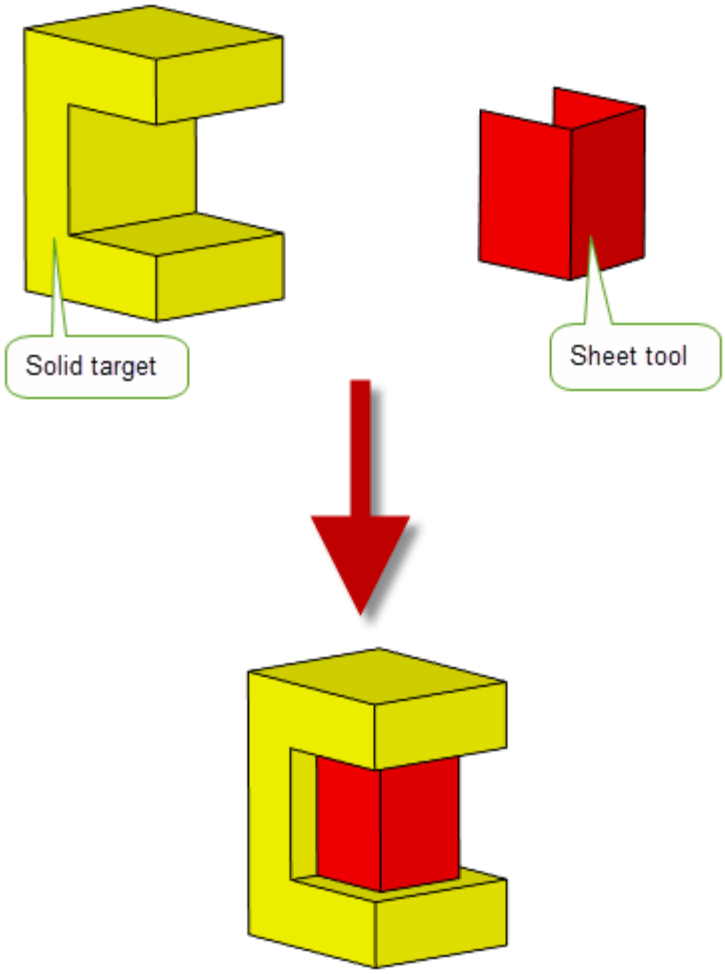


Figure 4–3 *Uniting solid and sheet bodies to enclose a solid region*

4.2.2.3 Punching sheets with solid bodies

By uniting a sheet target with solid tools, you can create a **punched** sheet, as shown in *Figure 4–4*. This functionality provides a simple way for you to add sheet tooling functionality to your application.

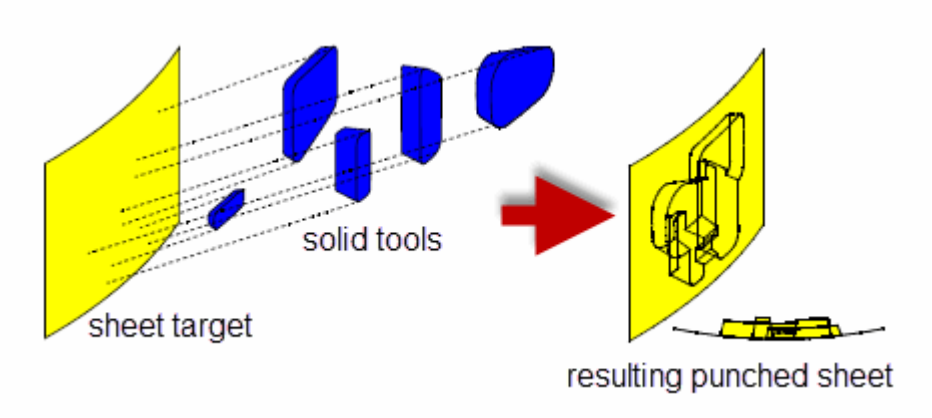


Figure 4–4 *Punching a sheet target body with five solid tool bodies*

4.2.2.4 Fencing off sections of a body

You can divide a solid target into sections by subtracting a sheet tool. You can then use Parasolid's **fencing** capability to choose which of these sections to retain after the boolean operation.

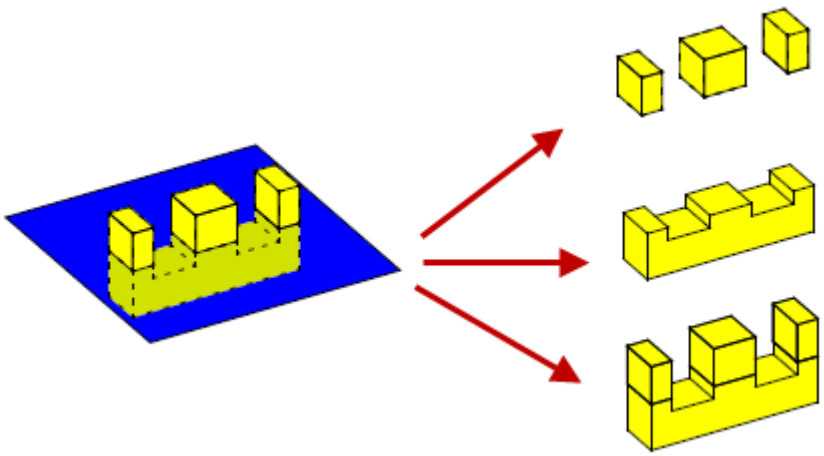


Figure 4–5 *Dividing a body into sections using fencing functionality*

4.2.2.5 Retaining topologies during a boolean operation

You can control which topologies to retain when solid regions overlap during a general boolean operation as illustrated in *Figure 4–6*.

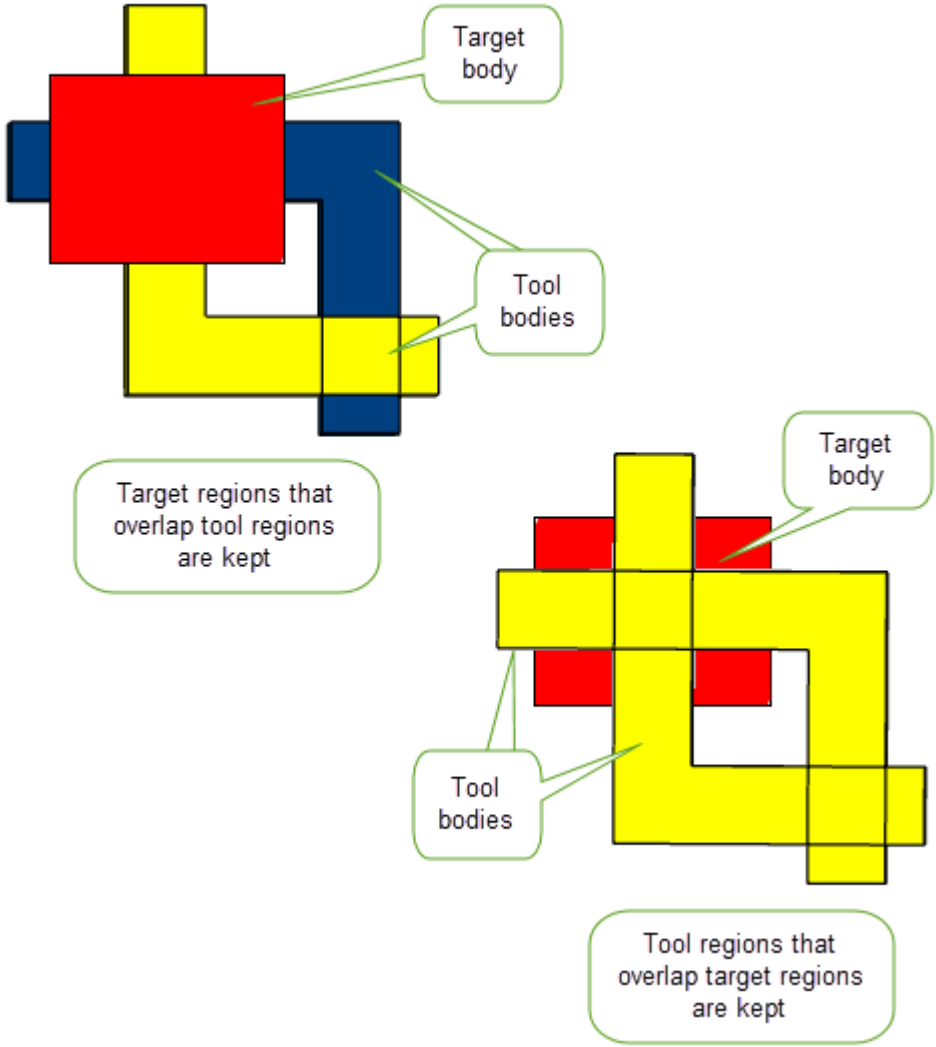


Figure 4–6 *Retaining topologies during a boolean operation*

4.2.2.6 Other options

Parasolid's boolean functionality supports a wide variety of other options that are not discussed here. These include:

- Options to improve the performance and reliability of boolean operations (such as an option to specify when tools do not intersect with each other)
- Options to control the resulting body (such as the ability to select regions to exclude from a boolean, and control over how edges in the result should be merged)
- Options to control the way the boolean operation is performed (such as the ability to extend edges that need to be imprinted in order to perform the boolean, and the ability to treat sheets as solids and vice-versa for tools and targets).
- Options that allow you to repair non-manifold edges created by booleans in which the target and tool share a common extrude direction

4.2.3 Boolean tools

Parasolid provides a number of boolean tools so that you can create your own specialized boolean functionality, in order to include unique features not available with the standard boolean operations. You could use these tools, for instance, to add preview functionality to your boolean operations, or to remove unwanted areas of overlapping sheet bodies before sewing them.

Boolean tools are available to let you:

- Imprint edges on a target and tool to show where the bodies intersect.
- Divide faces on a body (usually those created as a result of the imprinting operation) into distinct face sets so that you can decide which parts of the body to keep, and which to throw away.

To implement your own manual boolean functionality, you could imprint edges on a target and tool, identify the distinct face sets on the target and tool, and then remove the ones you don't want. Finally, you could use fuse the remaining face sets into a single body. *Figure 4–7* shows how you could use boolean tools to implement your own boolean subtract functionality.

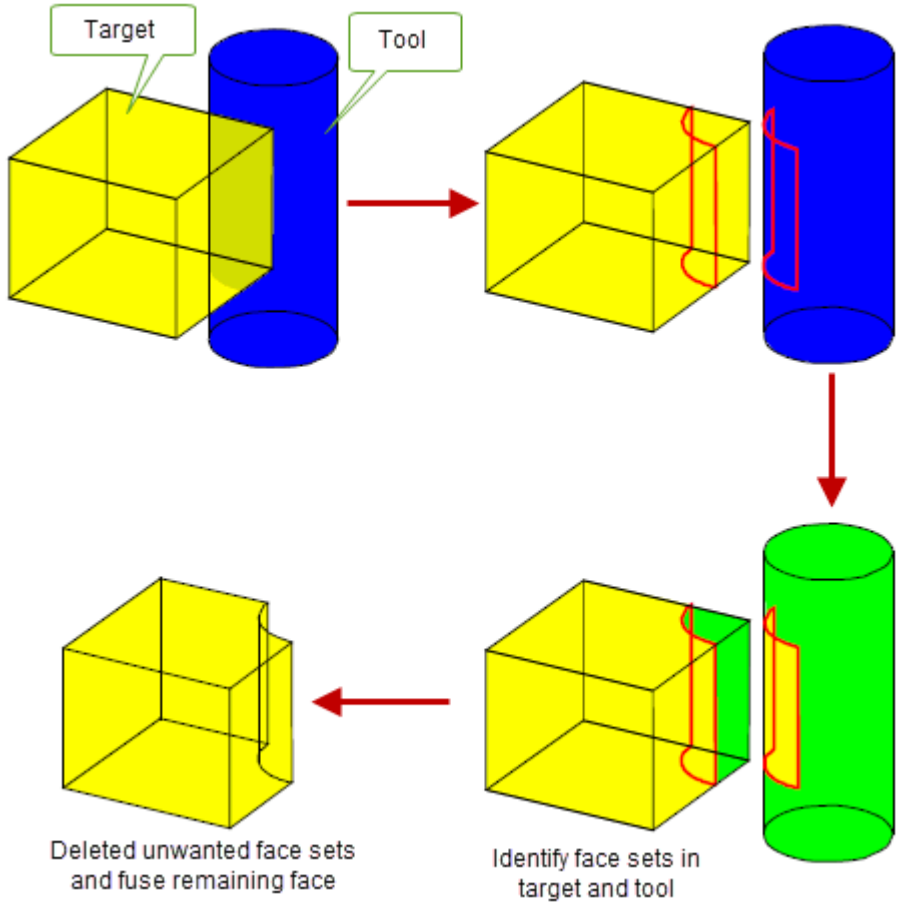


Figure 4–7 *Implementing your own subtract functionality using boolean tools*

4.3 Instancing

Instancing is a process that combines a single target and one or more tools with a local boolean operation and an array of transforms so as to repeat the boolean across target faces, thereby creating a set of instances of the specified tools, arranged according to the transforms you supply. It is a one-stop procedure for creating complex patterns, such as a series of holes in a plate, and offers significantly improved performance compared with creating the same effect using standard boolean operations alone.

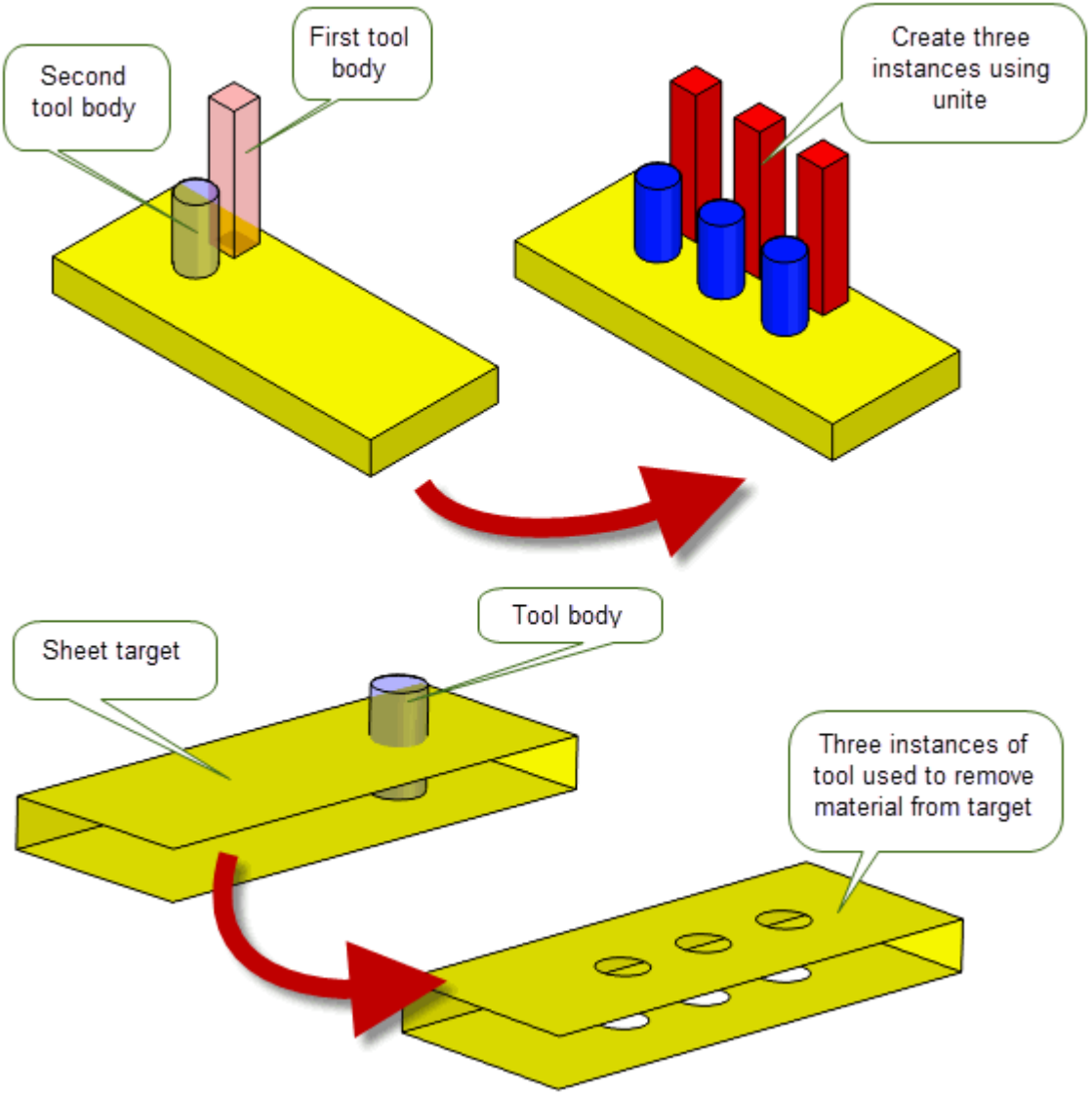


Figure 4–8 *Creating instances on solid and sheet bodies using single and multiple tools*

You can use many of the same options when creating instances of a tool as you can when performing local booleans.

4.4 Patterning

Parasolid also supports **patterning** functionality, which lets you create copies of a set of faces according to an array of specified transforms. Although patterning may initially appear very similar to instancing, it does not use any of Parasolid's boolean functionality. It therefore provides you with a useful alternative to instancing for cases where instancing may be difficult or impossible.

When creating patterns, you specify a set of faces on the target body, rather than a separate target and tool, together with an array of transforms that describes how the patterns are to be positioned. *Figure 4–9* shows a simple example, in which both the pocket and the boss on the body are to be patterned. A single transform is also given, resulting in the body shown. This particular example could not be performed using a single instancing operation, because it would require a boolean operation that could unite and subtract simultaneously.

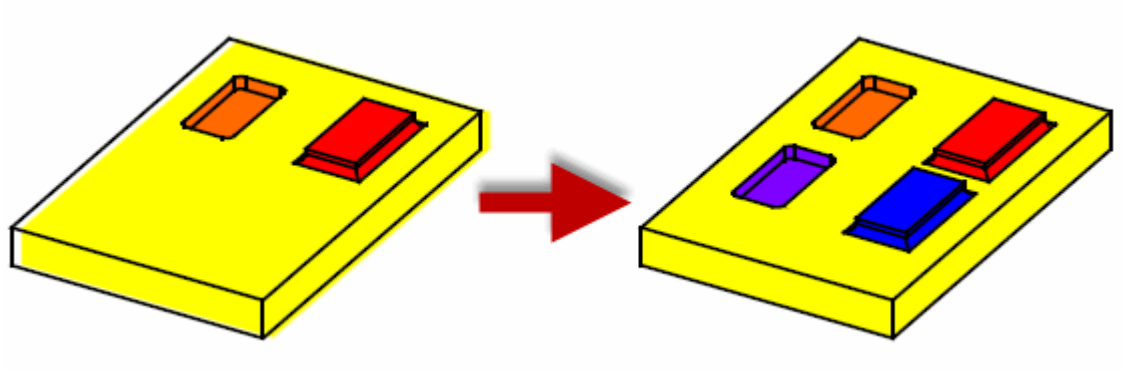


Figure 4–9 *Patterning example of one transform of both a pocket and a boss*

Patterning supports options to constrain patterned faces so that they do not overlap face edges, or do not intersect. In addition, you can create patterns that are repeated across different faces in the body. Parasolid also supports general patterning functionality, in which geometry for a set of patterned faces can be extended automatically in order to ensure the patterning is successful.

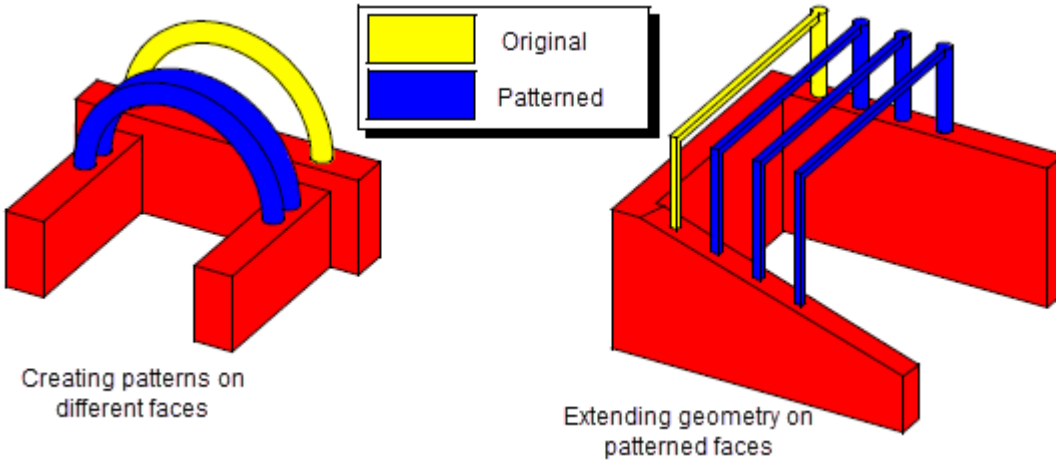


Figure 4–10 *Patterning within a face or to another face via rigid translation*

It is also possible to create patterns in different faces where the base of the original feature is not rigidly translated to the face containing the new instance; in some cases, it may straddle two or more faces.

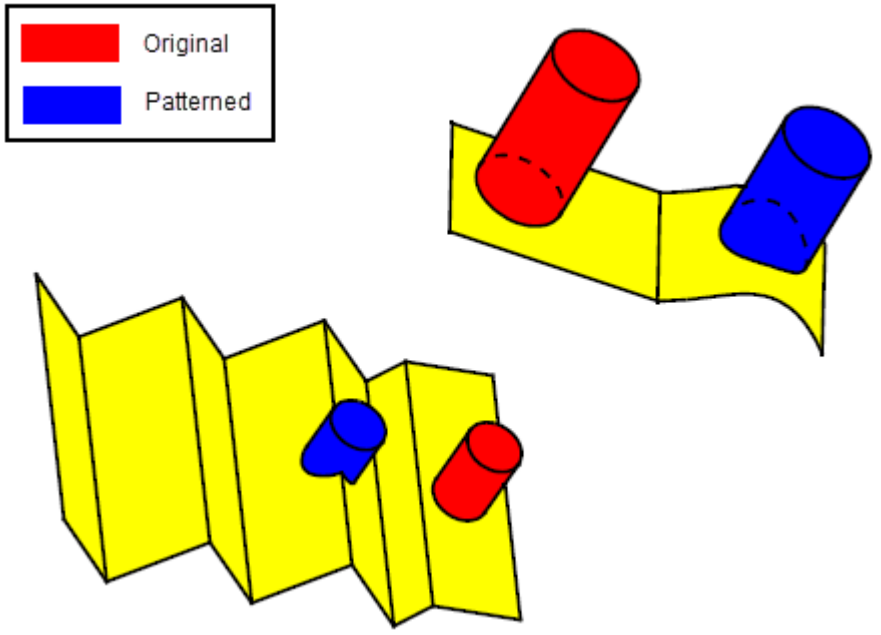


Figure 4–11 *Patterning to another face without rigid translation or to two faces*

Parasolid allows you to recreate blends around patterned instances: if a pattern face is a blend within a boundary loop, you can recreate that blend during the pattern operation in order to maintain the blend characteristic of that face.

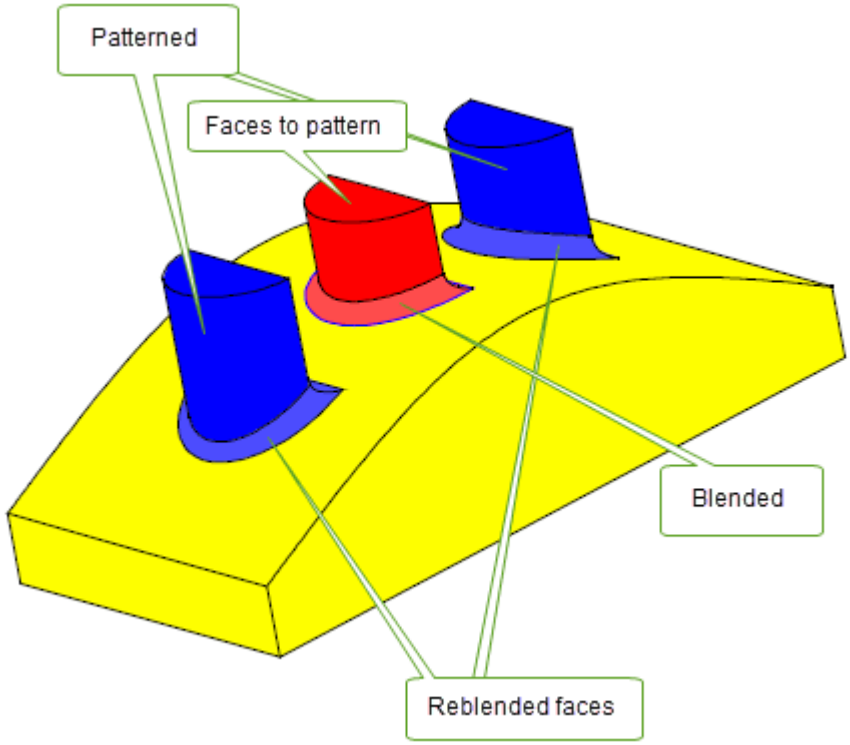


Figure 4–12 *Recreating blends on patterned faces*

4.5 Sectioning

Sectioning is the process of dividing a body into sections using faces, surfaces or sheets, and optionally throwing away unwanted sections. As well as providing some sectioning behaviour using the standard boolean functionality, Parasolid offers dedicated sectioning tools that work in a similar fashion to the standard boolean subtract operations (and, in particular, the fencing behaviour described in Section 4.2.2). Two broad categories of sectioning are supported by Parasolid: destructive sectioning and non-destructive sectioning.

4.5.1 Destructive sectioning

Destructive sectioning divides a single target body using a single tool body. As with booleans, the target body is modified and returned, and the tool is discarded. There are two types of destructive sectioning: global sectioning and local sectioning, and these are analogous to their boolean equivalents.

In a global sectioning operation, a body is divided into sections using a tool body that is either a surface or a sheet. Like global booleans, the resulting sectioned body is guaranteed to be topologically consistent, but because of the global nature of the operation, performance might be an issue in some circumstances.

In a local sectioning operation, selected faces of a target are divided using selected faces from a sheet tool. Like local booleans, the restricted focus of a local sectioning operation provides excellent performance, but can result in topological inconsistencies: locally sectioned bodies may need to be checked before continuing with modelling.

Both global and local sectioning functions support a similar range of options to global and local booleans.

4.5.2 Non-destructive sectioning

Non-destructive sectioning supports a different range of options to destructive sectioning, allowing you control over details such as the type of body returned by the operation.

In non-destructive sectioning, nothing is deleted after the operation and, instead of modifying the target, the specified tools are altered. You can section any number of bodies using any number of tools and both surfaces and sheet bodies can be used as targets.

With non-destructive sectioning you can also choose to create wire body sections from a set of offsets of a single tool plane as illustrated in *Figure 4–13*. This functionality can be useful when creating sections in additive manufacturing workflows where models are sliced in preparation for printing.

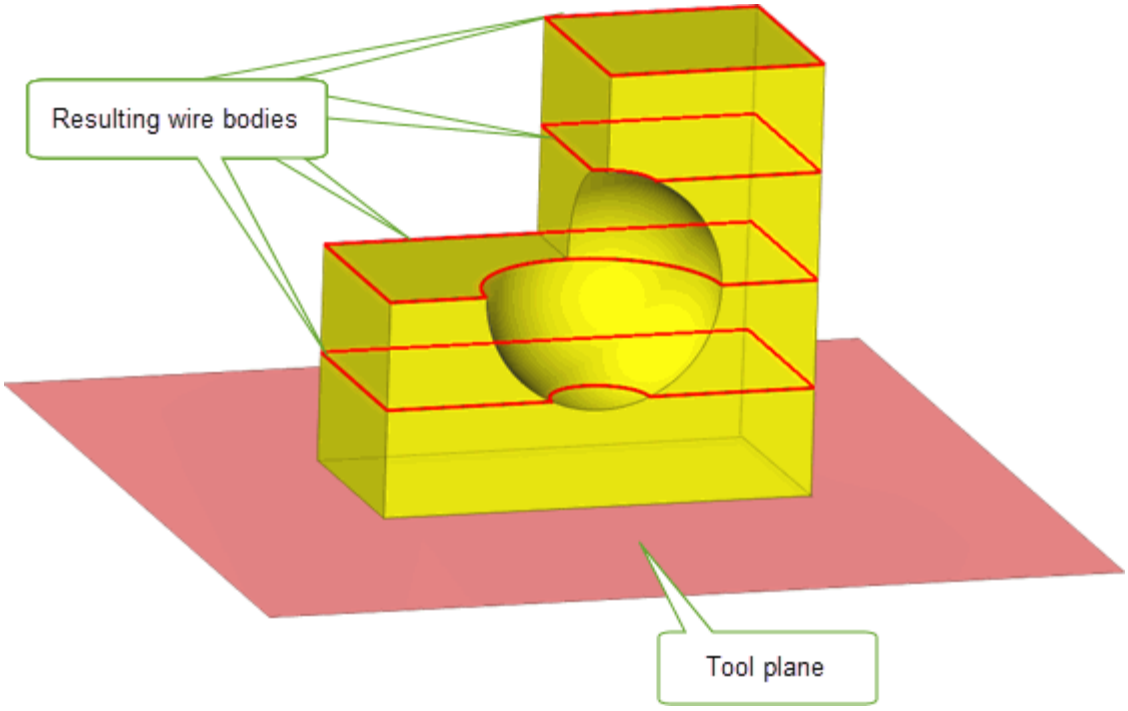


Figure 4–13 *Sectioning a body using a tool plane*

4.6 Clash detection

You can analyse any combination of solid, sheet and wire bodies to find out whether or not they clash; if they do, you can discover the nature of the clashes. Parasolid is able to distinguish between the following types of clash:

- **Interference:** bounding topologies of the entities cross with each other
- **Abutment:** bounding topologies of the entities touch each other
- **Containment:** one entity is entirely within the other, and their bounding topologies do not touch

The precise nature of these types of clash depends on the types of body that you are analysing. For example *Figure 4–14* illustrates the ways in which sheet bodies can clash.

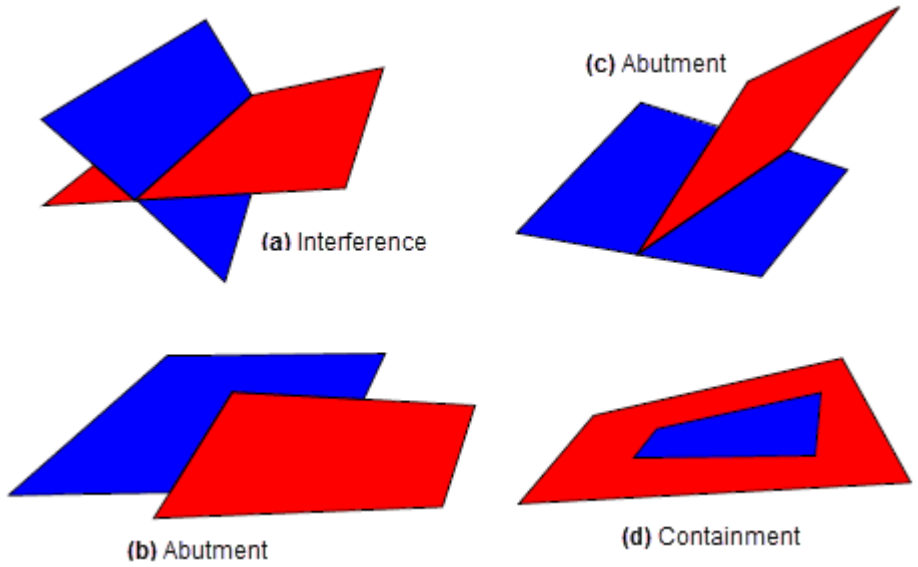


Figure 4–14 *Detecting clashes in sheet bodies*

Local Operations 5

5.1 Introduction

Local operations are modelling operations that are restricted to a specific area of a body. They are usually performed on a face or set of faces within a model. Because the operations are restricted to specific areas, local operations can usually be performed on general bodies as well as manifold bodies, so long as those general bodies are *locally manifold*, that is, manifold in the area that the operation is performed in.

You can use local operations in your application to make fast changes to local regions of large and complex models. Local operations are essential if you are developing constraints-managed or parametrically-driven applications.

Parasolid provides local operations functionality for:

- Offsetting and transforming
- Tapering faces
- Editing faces and healing wounds
- Editing the geometry of faces
- Spinning and sweeping entities
- Generic face change operations

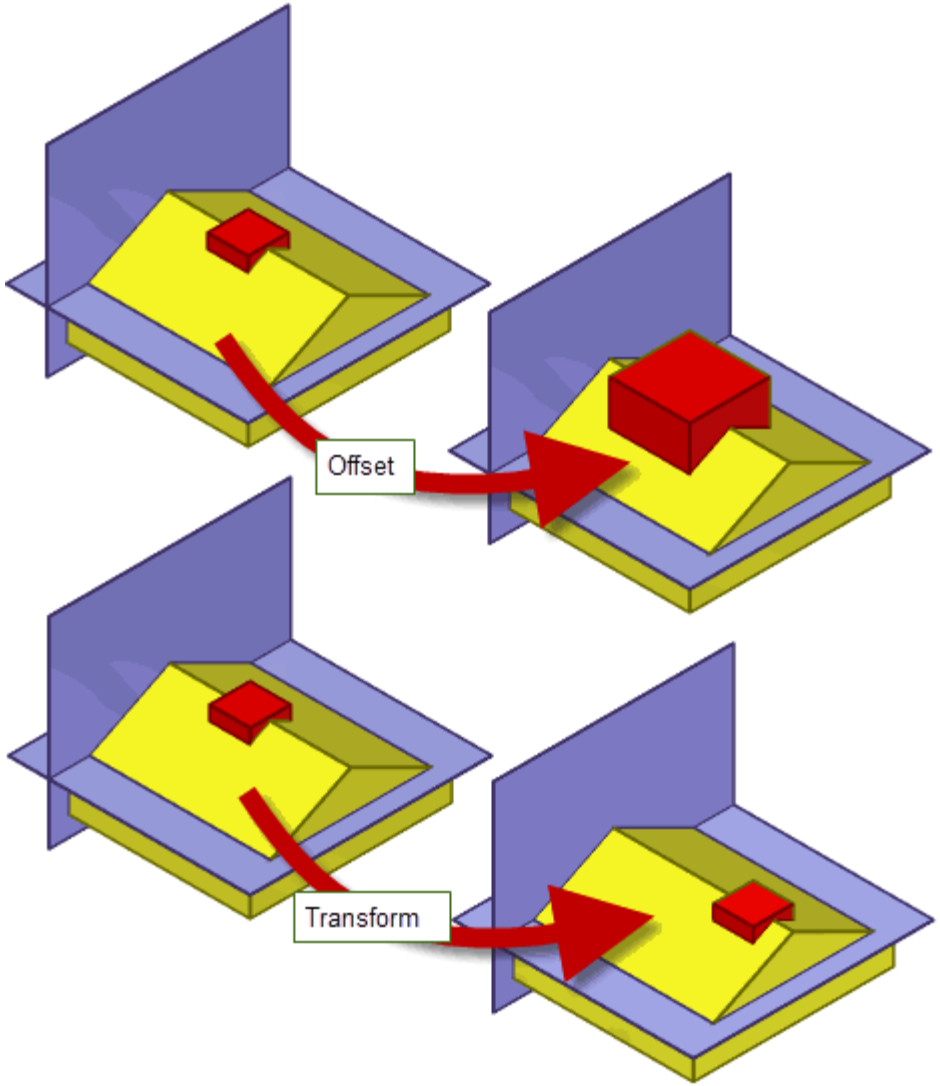


Figure 5–1 *Offset and transform local operations on general bodies*

5.1.1 Topological changes with local operations

Engineering changes to the geometry of a model often change the model's topology as well. Parasolid's local operations support many of these changes, including the removal of self-intersecting or clashing areas of the model, and the addition of new edges to the model at places where a vertex is split into two as a result of an operation.

Because of the nature of the changes made by local operations, however, not all topological changes can be supported, and result bodies cannot always be guaranteed to be topologically

consistent. It is therefore generally recommended that you check results before continuing modelling.

5.2 Offsetting operations

Offsetting, hollowing and thickening are related modelling operations that are commonly used in MCAD systems to model, for example, objects for injection moulding or casting, or to model the outer skins of aerodynamic parts. Parasolid provides powerful offsetting functionality which itself forms the “parent” technology for its hollowing and thickening functionality. You can use offset operations directly, as well as capitalize on them via hollowing or thickening (which are themselves not considered local operations). For all these operations, you can choose to repair any degeneracies on the surface of the body involved in the operation. Information about these repairs is saved in the Parasolid report.

5.2.1 Offsetting

Offsetting is the process of moving specified faces in a solid or sheet body outwards or inwards.

In *Figure 5–2*, two faces whose underlying geometry is free-form are offset. Parasolid has extended the underlying surface(s) so that the new offset surfaces meet correctly.

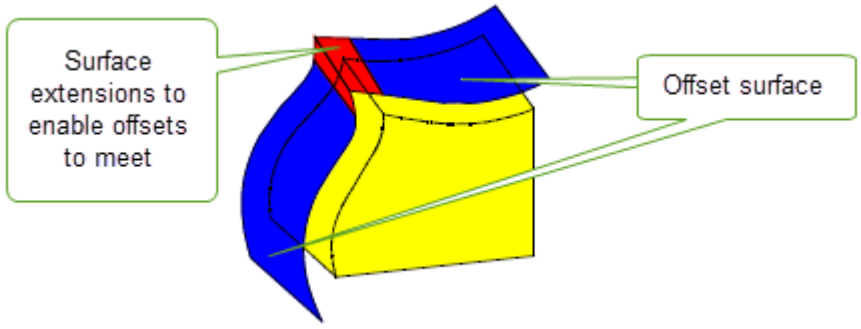


Figure 5–2 *Offset surface extended*

Because offsetting operations form the basis for hollowing and thickening, the functionality described here is also available to hollowing and thickening operations.

5.2.1.1 Removing self-intersections

Generating offset surfaces can sometimes cause self-intersections which, if left, would cause the body to become invalid. Parasolid can remove self-intersections automatically, generating new surfaces wherever holes remain as a result of removing the self-intersection. *Figure 5–3* shows some examples.

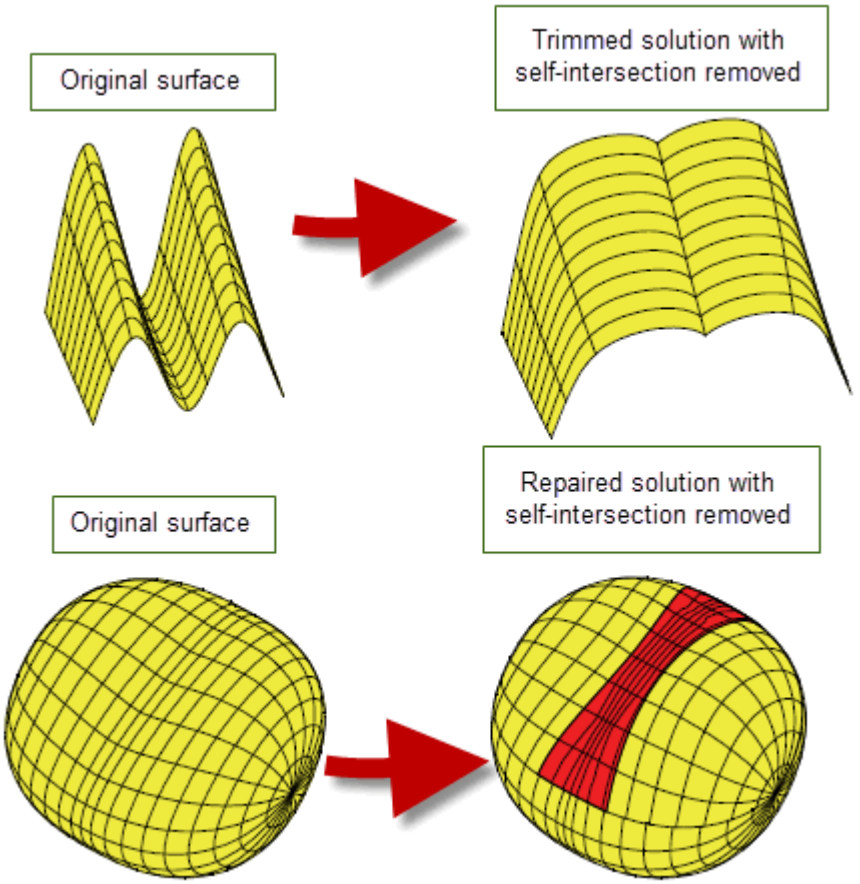


Figure 5–3 *Removing self-intersections during offsetting operations*

5.2.1.2 Step offsets

If required, Parasolid can generate additional side faces during offset operations, in order to ensure the validity of the resulting body.

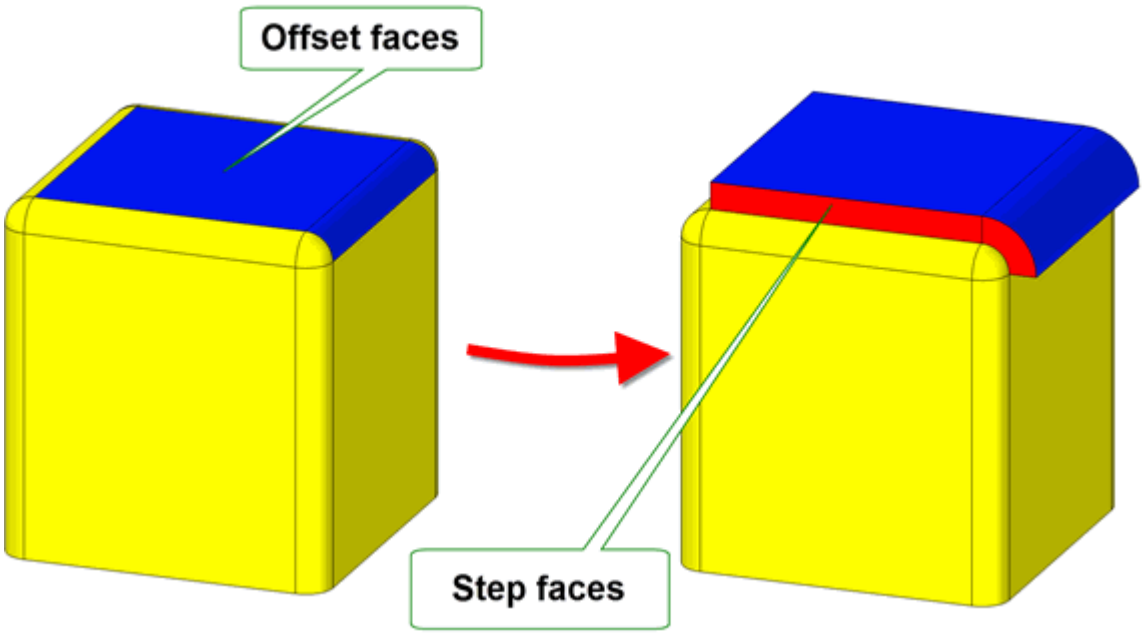


Figure 5–4 *Step offset faces during offset operations*

5.2.1.3 Creating blends from offset edges

Rather than extending adjacent offset faces to determine the new position of the edge, Parasolid can optionally round off the edges between adjacent offset faces during an offset operation.

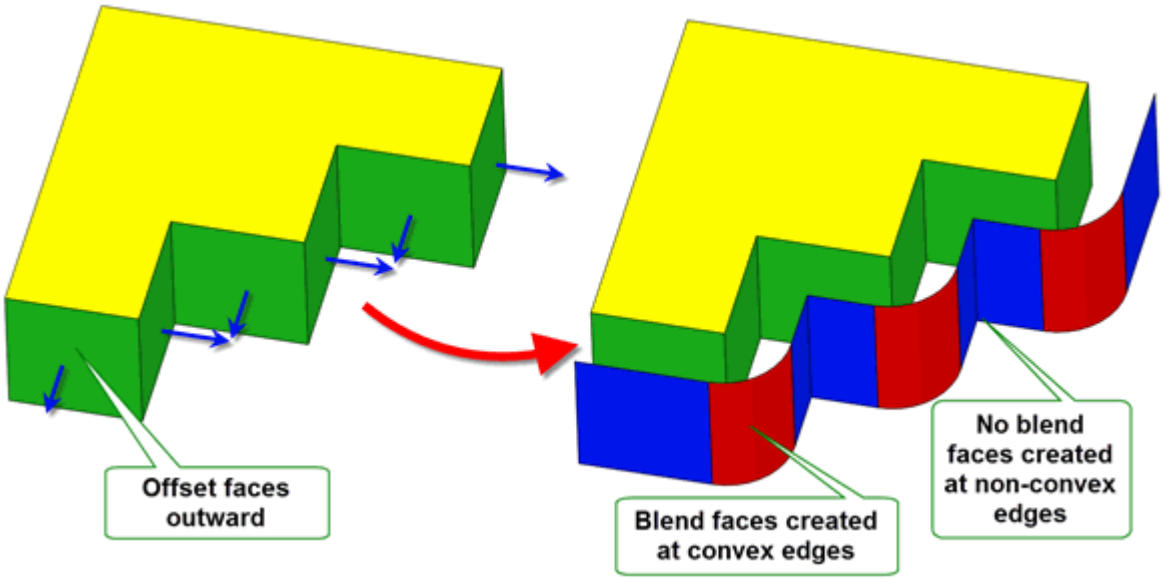


Figure 5–5 *Creating blends from offset edges*

5.2.1.4 Offsetting edges

Parasolid can also offset any set of connected edges in a given direction within their owning body. It can offset both laminar and non-laminar edges, and the resulting offset edges are imprinted on the body. *Figure 5–6* shows an example where a set of three edges have been offset on a sheet body that contains a hole and a sharp edge, such that the resulting set of four offset edges crosses both the sharp edge and the hole.

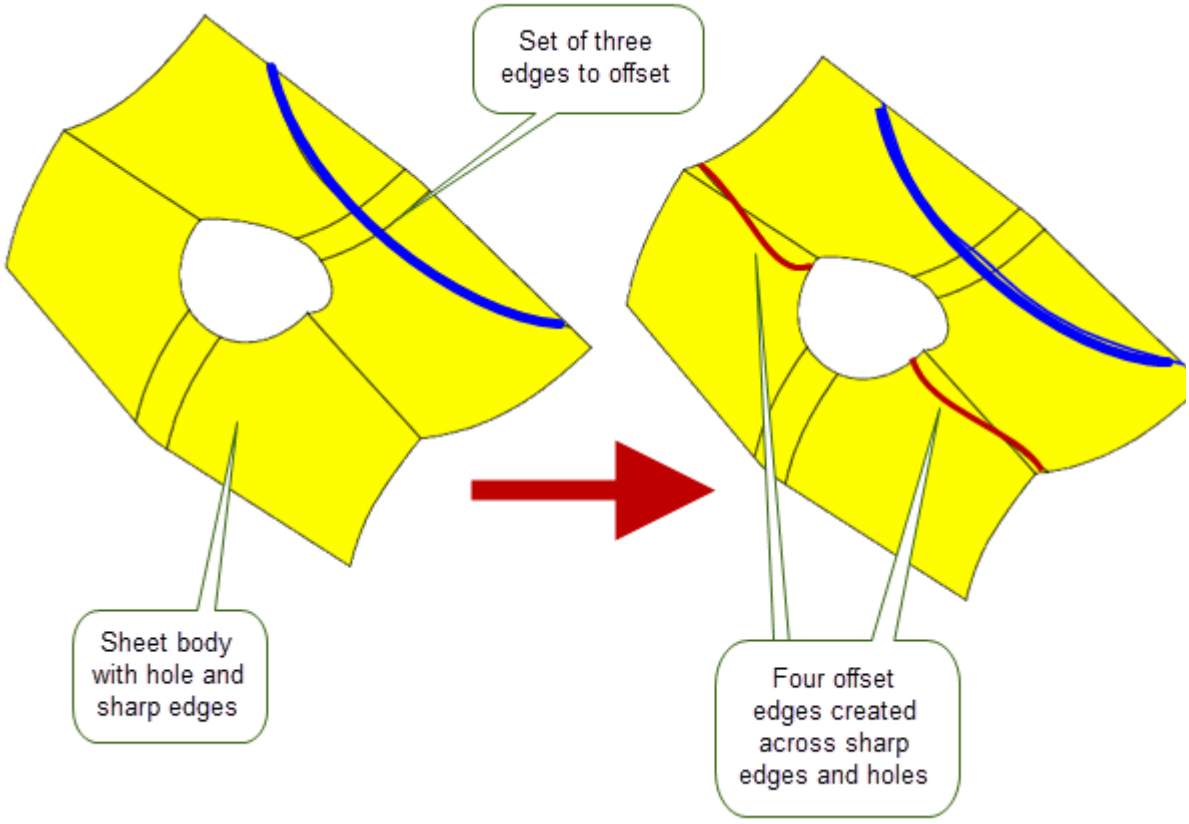


Figure 5–6 *Offsetting edges in a body*

5.2.2 Hollowing

Hollowing uses offsetting technology to create a void region in a solid body. In order to hollow a body, each face in the body is offset by a specified amount. Different faces may be offset by different amounts, and the hollow body can be opened up by specifying some faces as **pierce faces**.

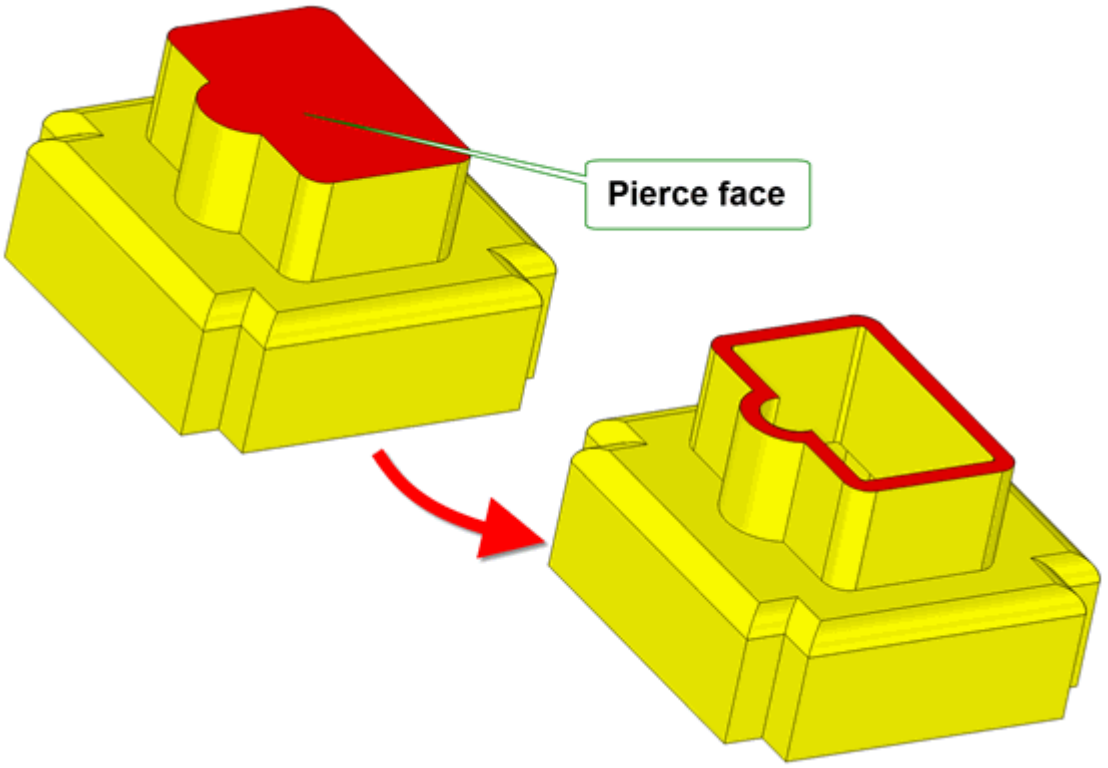


Figure 5–7 *Hollowing a body with two pierce faces*

Parasolid can also identify and deal with tangent pierce faces (pierce faces that share at least one smooth edge with another non-pierce face), creating the additional side faces required to hollow bodies in such a manner, as shown in *Figure 5–8*.

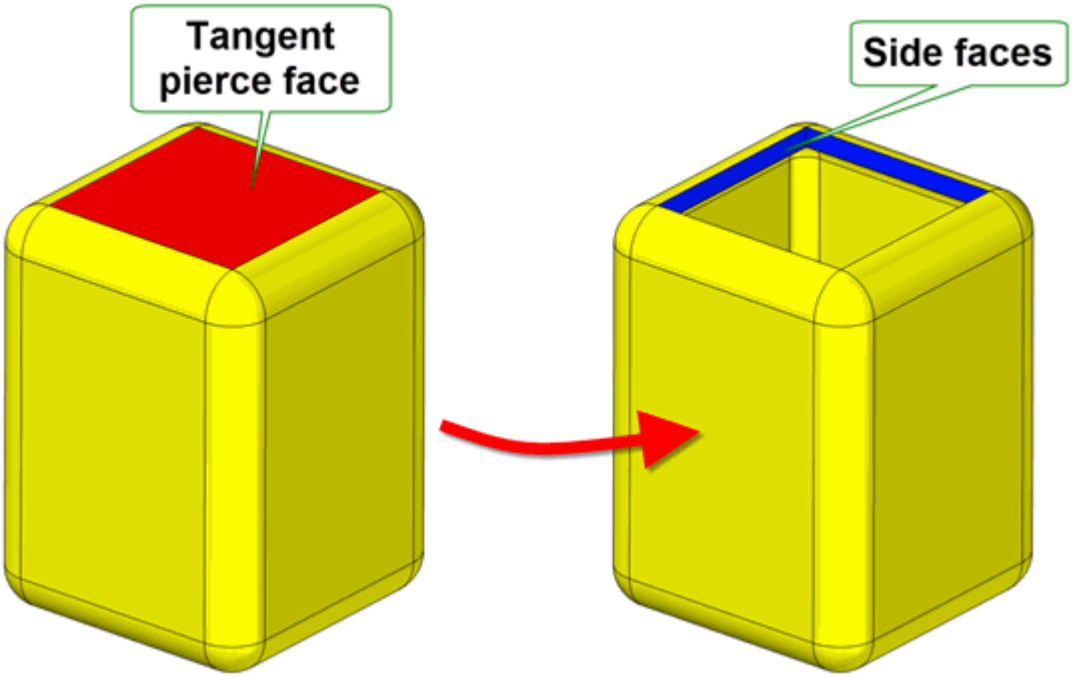


Figure 5–8 *Hollowing a body using a tangent pierce face*

Parasolid can also round off edges of the body that are offset by the hollow operation, as shown in *Figure 5–9*.

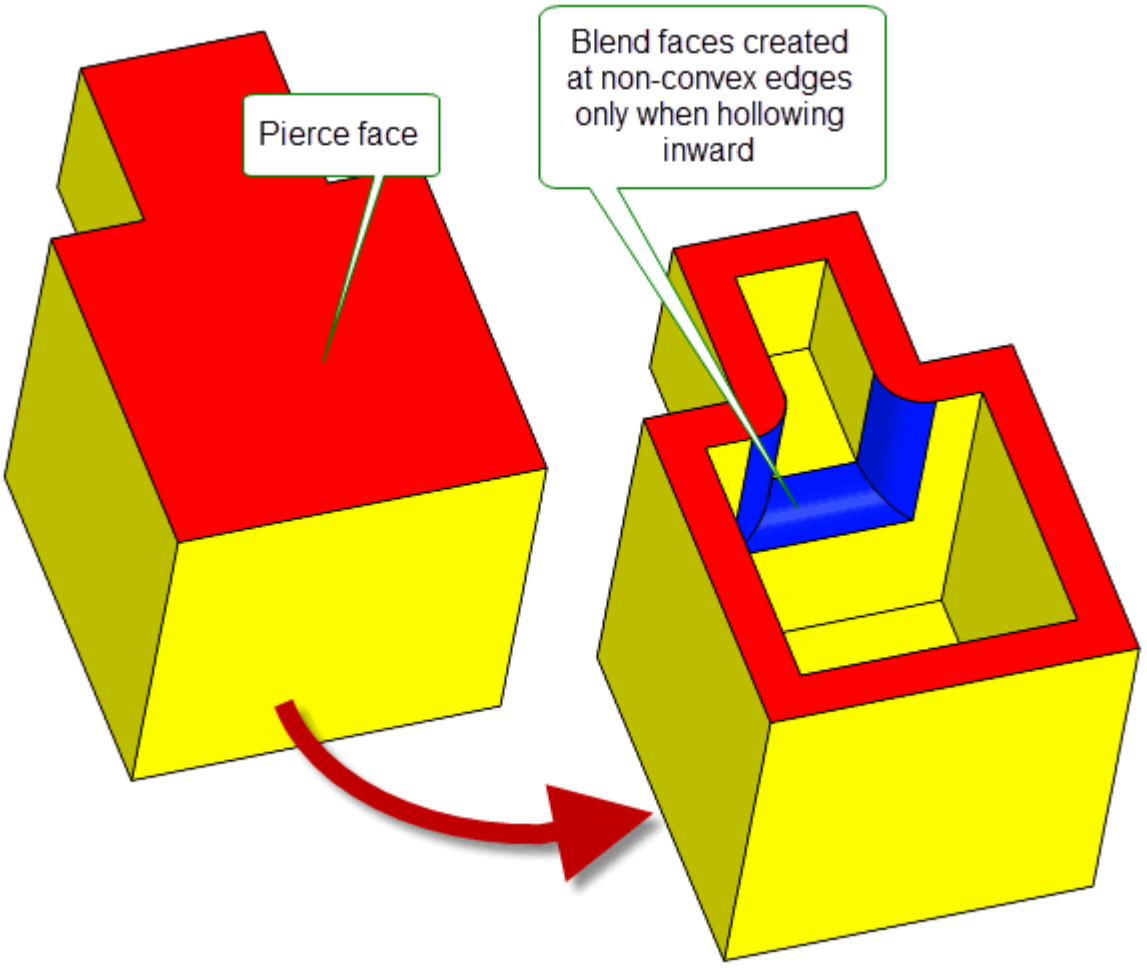


Figure 5–9 *Creating blends from offset edges during hollow operations*

Parasolid can also perform local hollowing operations, in which only part of a body is hollowed. For example:

- If you are designing a bath-tub with four legs, you can hollow out the tub while leaving the legs unhollowed.
- If you have added a feature to a body that has already been hollowed, you can hollow out just the new feature.

Figure 5–10 illustrates how the bowl of a wine glass can be hollowed out while the stem of the glass remains solid.

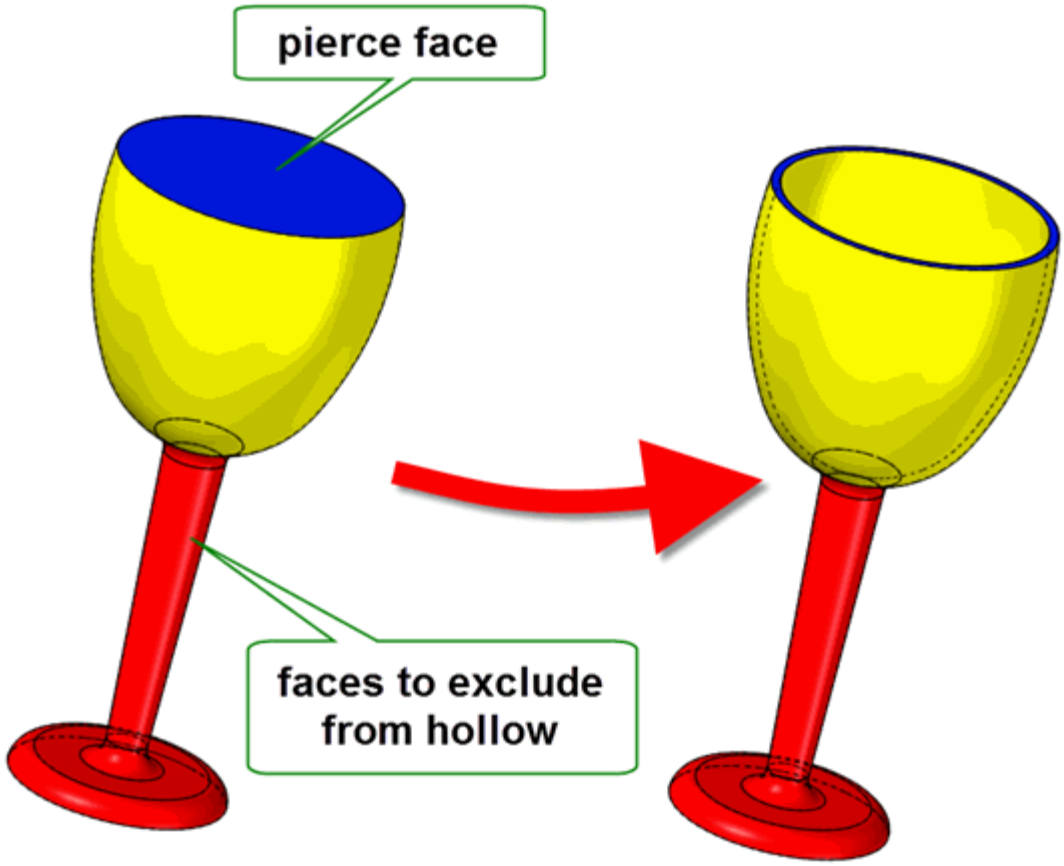


Figure 5–10 *Excluding faces from a hollow operation*

5.2.3 Thickening

Thickening is the process of taking a sheet body and thickening it into a solid body by offsetting it by a fixed amount. Although not strictly a local operation (because it works on a whole body, rather than specific faces in the body), Parasolid's thickening functionality also uses the offsetting technology used by hollowing. Like offsetting and hollowing, a thickened body may undergo a variety of topological changes, and Parasolid can remove self-intersections in either the side face that is created as the result of a thicken, or in the resulting offset face itself, automatically. Parasolid can also identify and repair degeneracies on the surface of the body to be thickened.

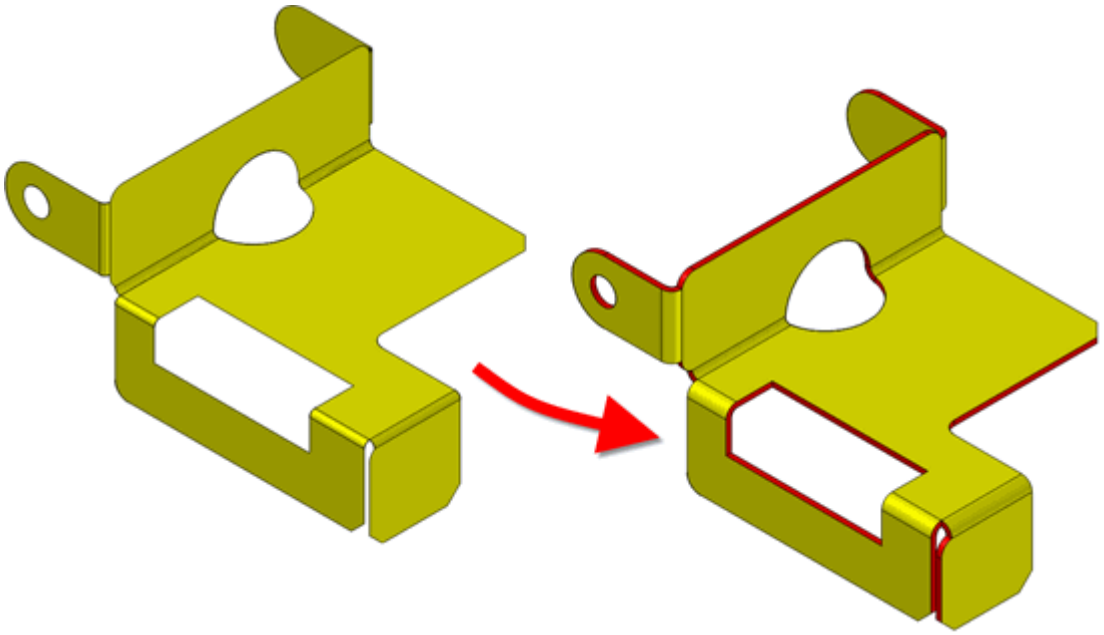


Figure 5–11 *Thickening sheet bodies*

During thickening operations you can:

- Specify a “punch” direction in which a sheet body is thickened.
- Specify a surface to attach to side faces.
- Round off edges that are offset by the thicken operation.
- Specify pierce faces to open up the thickened body, in a similar way to hollowing.
- Specify that Parasolid should generate step offset faces in cases where different faces are thickened by different amounts, in a similar way to offsetting.
-

Some of these options are illustrated in *Figure 5–12*.

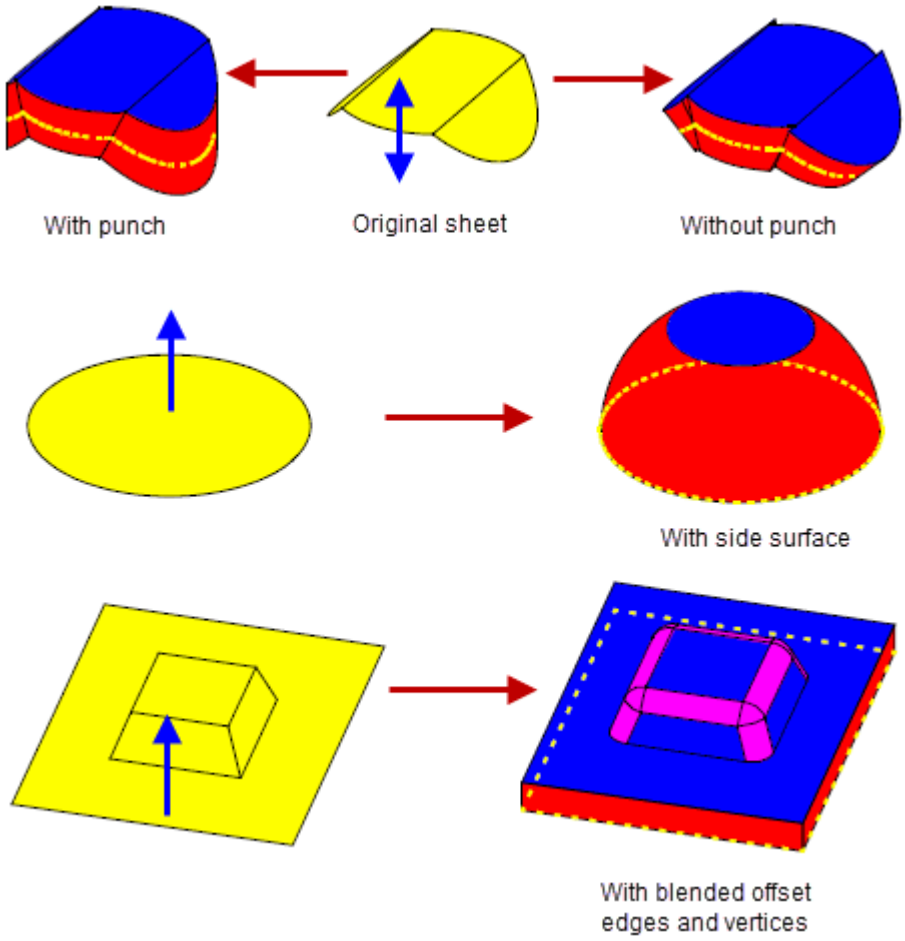


Figure 5–12 *Thickening options*

5.3 Tapering faces

Parasolid provides sophisticated support for adding **taper** (sometimes known as **draft**) to models. You can add taper to either specific faces in a body, as described in this section, or to an entire body.

During the tapering process, specified faces in a body are modified so that they form a minimum angle with a given direction. This is commonly required when designing molds: faces that are parallel to the direction in which a part is drawn out of a mold need to be tapered inwards slightly with respect to the direction they are drawn so as to facilitate easy removal from the mold, as shown in *Figure 5–13*.

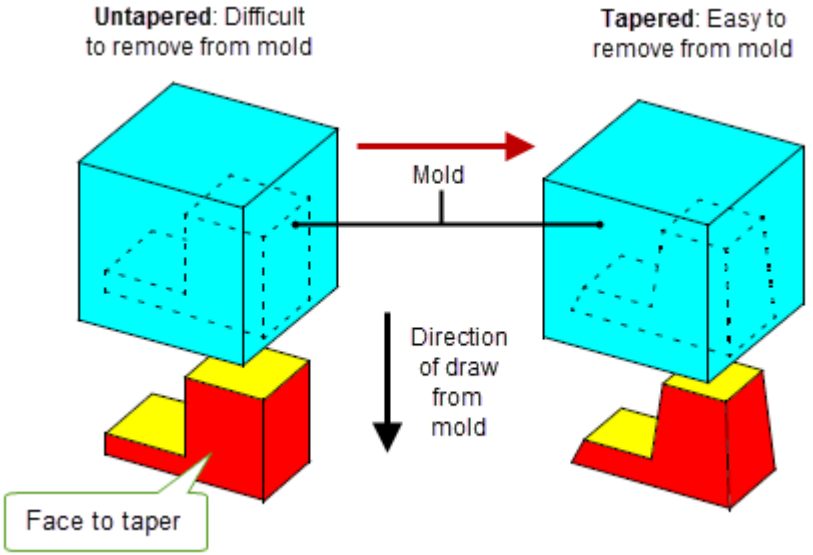


Figure 5–13 *Tapering vertical faces to improve molded parts*

Tapering is also very useful for improving the aesthetic appearance of a part. Parasolid's tapering functionality lets you offer your users an unprecedented level of control, letting them design a part and then add taper to discrete areas afterwards, rather than requiring the designer to build tapered faces into the part from early on in the design process. You can apply different angles of taper to different faces in the body in a single operation.

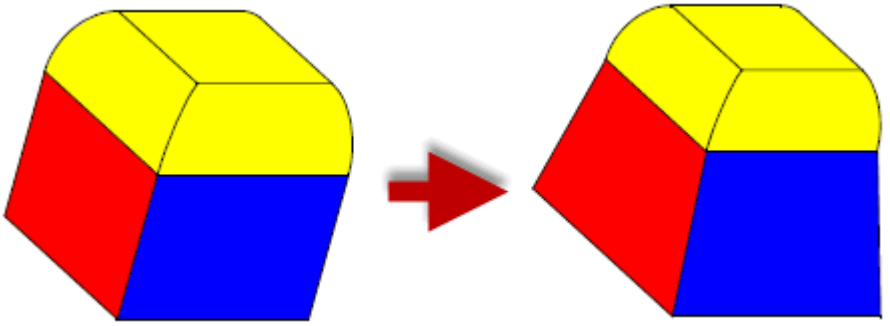


Figure 5–14 *Tapering different faces by different angles*

As well as providing the functionality to create tapered faces, Parasolid provides tools to help you decide where taper need be applied on a body. This is particularly useful in mold design, where a small taper may need to be added to a body at all places where the face is steeper than the required taper angle, with respect to the direction of draw.

Figure 5–15 illustrates the workflow that you would typically use to analyse and add taper to a body.

- First, the face to be tapered is analysed with respect to the taper angle that you want to apply.
- Next, isocline curves are added to the body to divide the face into regions that are steeper than the taper angle, and regions that are not. Only the steep regions require taper to be added.
- Finally, taper is added to the body.

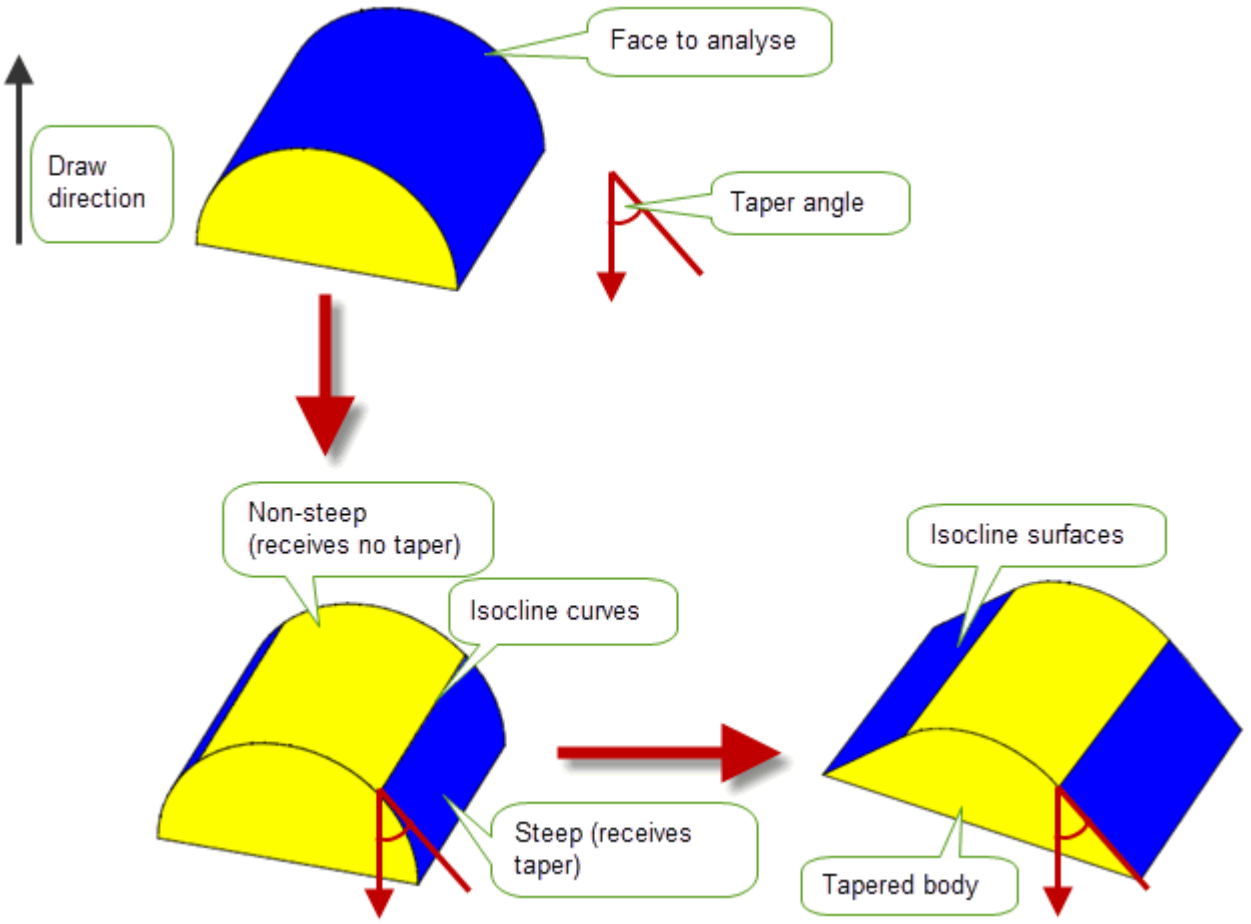


Figure 5–15 *Dividing a face into steep and non-steep regions*

Parasolid provides three methods of taper to choose between:

- Isocline-based tapering is suitable for adding draft when creating molds.
- Curve-based and surface-based tapering are both particularly suited to aesthetic tapering.
- Normal-to-surface tapering produces tapered surfaces that are not dependent on the draw direction but vary with the normal of a reference surface.

5.3.1 Step tapering

Sometimes, when a face to be tapered contains edges that are not on the taper plane, regular tapering produces less than ideal results. In such cases, you can perform a step taper, in which edges of the taper face that do not lie in the taper plane are not themselves tapered: instead, a step face extending from the edge to the tapered face is created. Step tapering is illustrated in *Figure 5–16*.

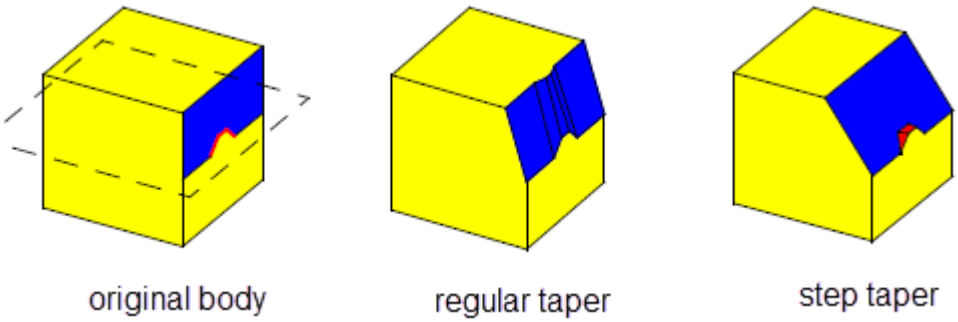


Figure 5–16 *The effect of regular tapering against step tapering*

There are two types of surfaces that you can use for step tapers:

- **Tapered surfaces** are themselves tapered with respect to the taper face.
- **Normal surfaces** are ruled surfaces perpendicular to the taper face.

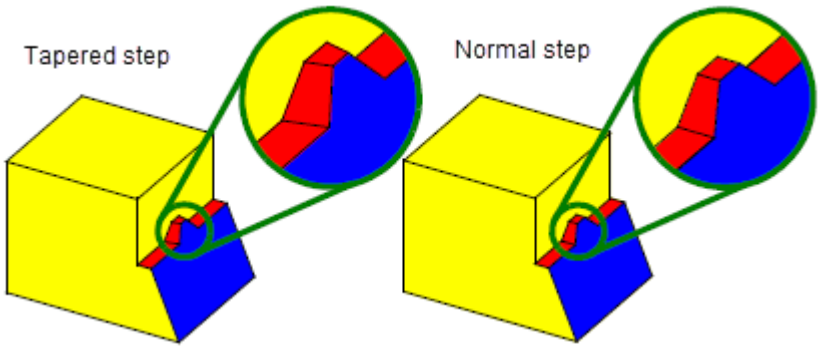


Figure 5–17 *Tapered and normal step faces*

If you wish, you can ask Parasolid to create a normal step automatically wherever a taper face meets a non-taper face smoothly, such as the case shown in *Figure 5–18*.

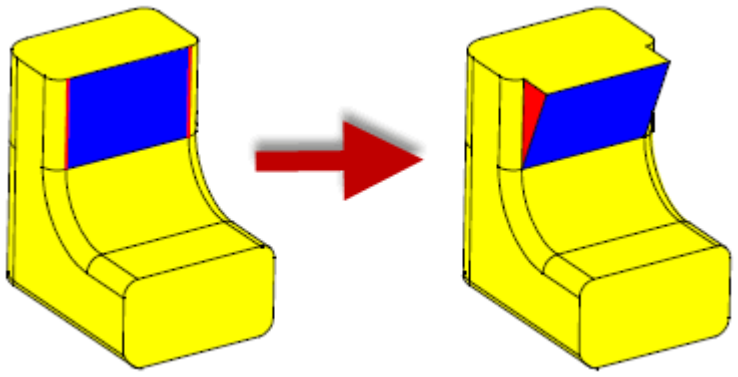


Figure 5–18 *Automatic step tapering*

In addition, Parasolid can automatically add a step face when adjacent faces become separated as a result of tapering them by different angles, as shown in *Figure 5–19*

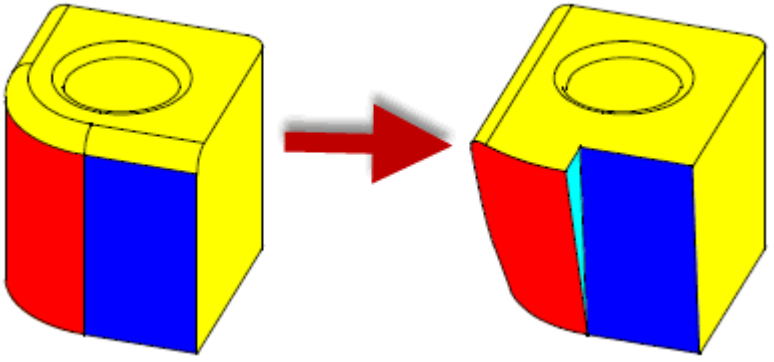


Figure 5–19 *Creating step faces between taper faces*

Rather than allowing Parasolid to generate the step face, you can provide one yourself using a parting body, as shown in *Figure 5–20*.

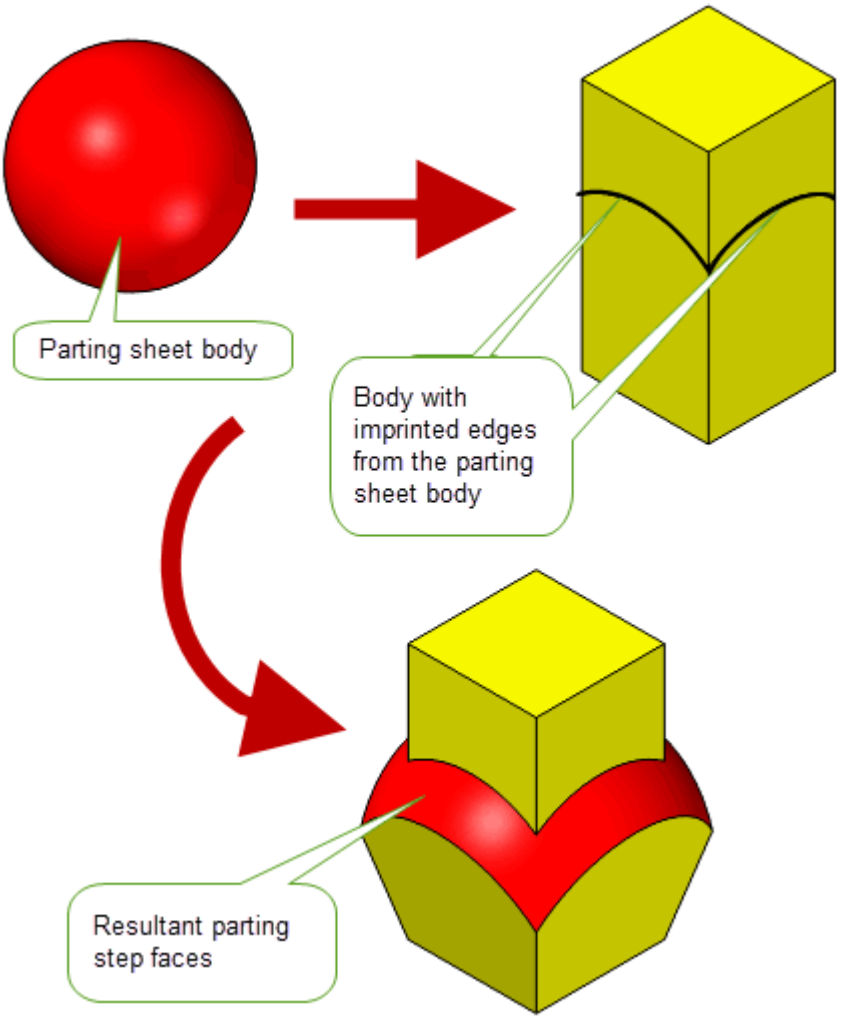


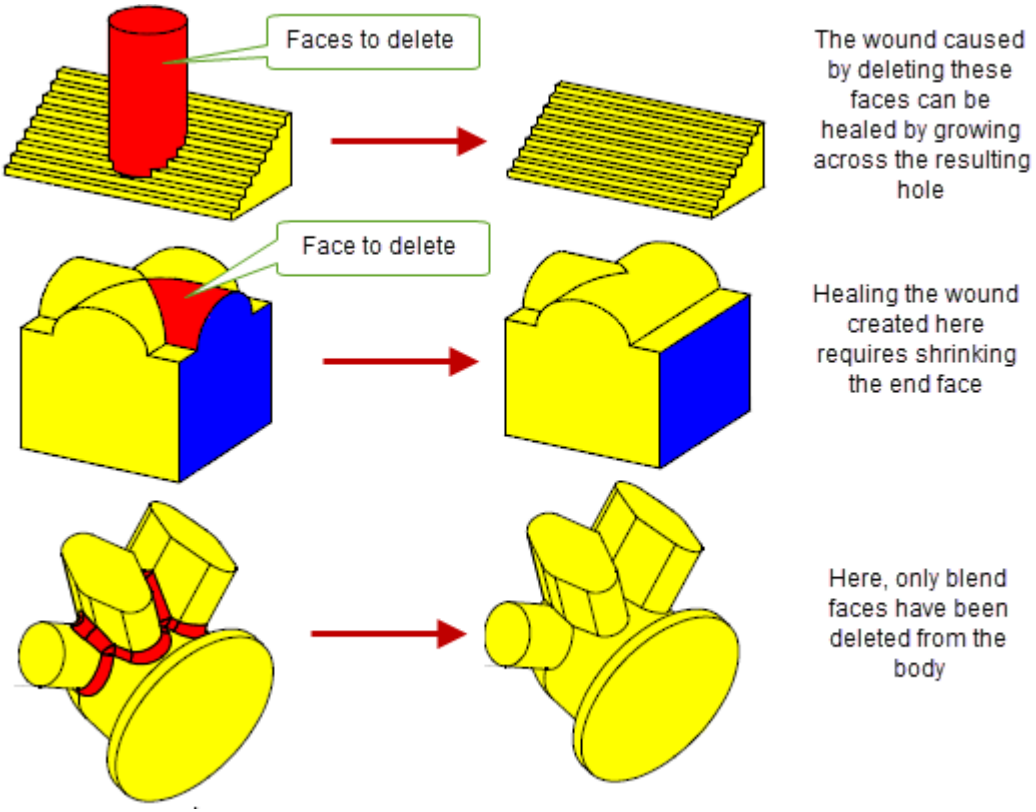
Figure 5–20 *Creating step faces from a parting sheet*

5.4 Editing faces and healing wounds

Parasolid includes tools for performing simple edits to faces, such as:

- Deleting arbitrary faces from a body
- Deleting blend faces from a body
- Copying specific faces and creating a new body from them
- Removing specific faces and creating a new body from them
- Removing trimmed boundary features from a body

A major issue with all of these operations is the wound that is usually left as a result of editing faces on a body; when a face is removed, a hole usually remains. Parasolid can repair these wounds automatically, using a number of strategies that you can control to get the precise results you want. *Figure 5–22* shows a variety of face edit operations, and illustrates some of the healing processes that Parasolid performs automatically.



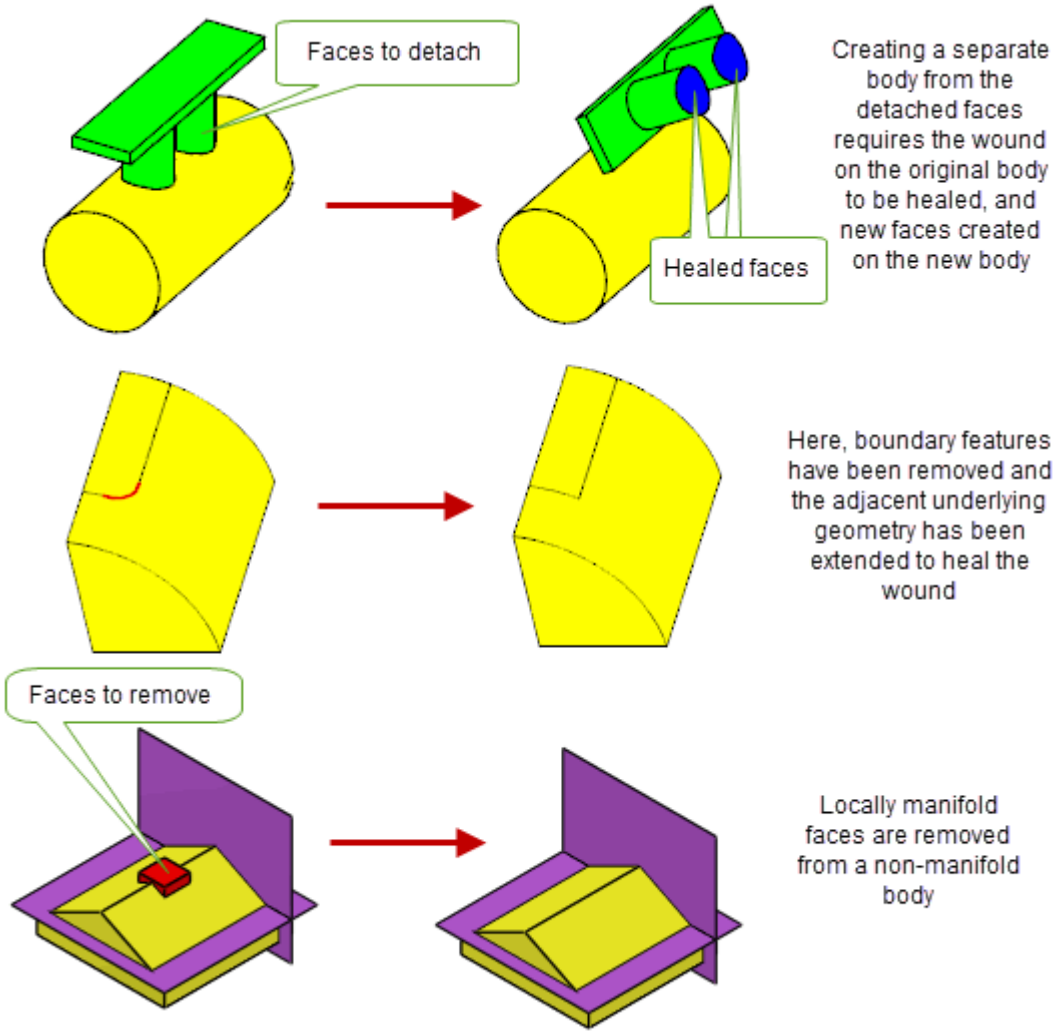


Figure 5–21 *Healing wounds after face edits*

As well as removing faces from a body, Parasolid provides tools for adding faces to a body or replacing existing faces in a body. This can be used for:

- Replacing missing geometry after importing data from a non-Parasolid based application
- Simplifying models
- Filling holes in a body

You can supply a patch yourself or, more commonly, let Parasolid generate one for you. Whichever option you choose, Parasolid fuses the patch with the original, as shown in *Figure 5–22*. The last illustration in the image, shows how position data can be used to aid Parasolid in building patch data.

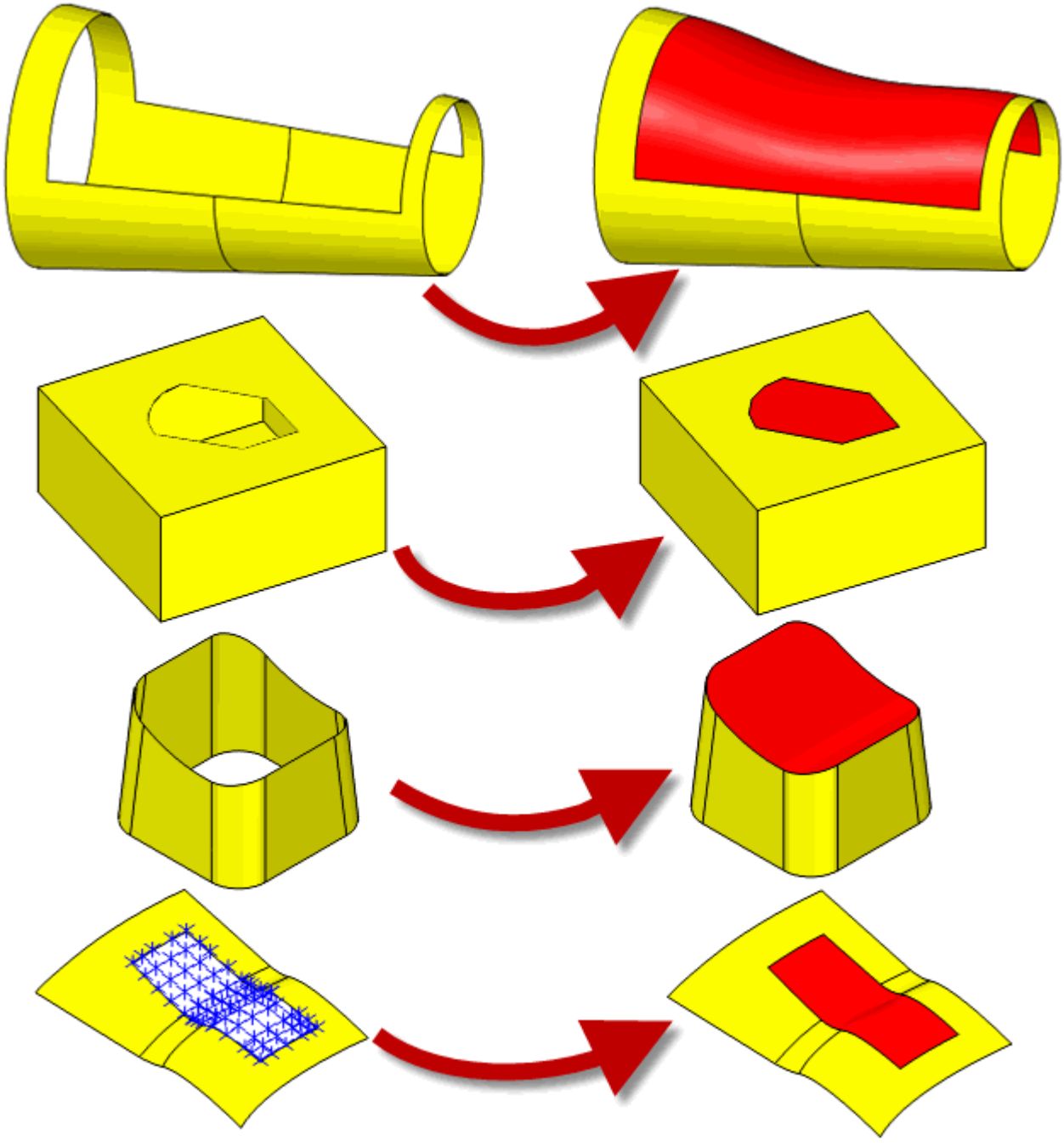


Figure 5–22 *Patching open regions in sheet bodies*

5.5 Editing the geometry of faces

As well as changing or creating faces in a body, Parasolid provides tools for creating and editing the geometry related to specific faces in a body: primarily the surfaces that are attached to the faces themselves.

If you have a model that contains rubber surfaces (that is, a face that has no geometry attached), you can create and attach a surface that fits it. Rubber surfaces can occur when scribing lines on a minimum body, for example, and this functionality provides a simple way to remove them.

You can also replace the surfaces attached to selected faces in a body, as shown in *Figure 5–23*

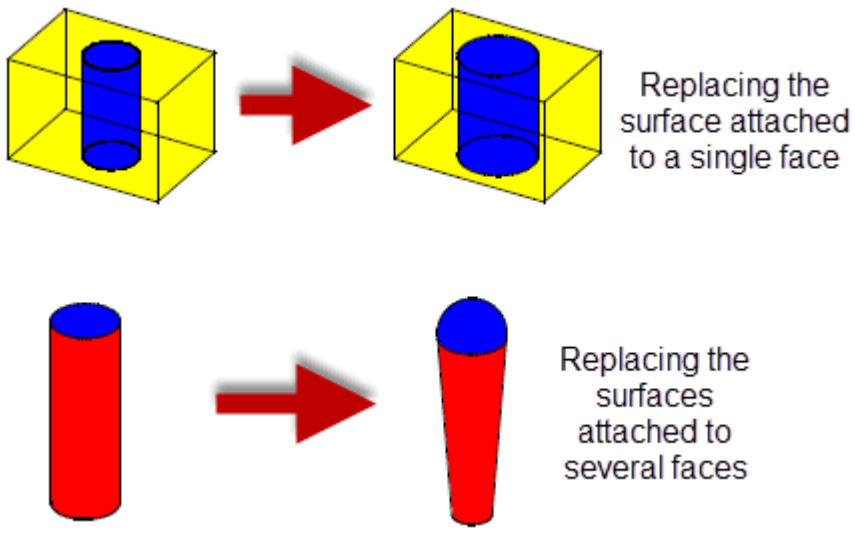


Figure 5–23 *Replacing surfaces attached to faces*

When replacing surfaces, Parasolid lets you:

- Choose whether adjacent faces are merged together where possible.
- Choose which curves in the body are recalculated by Parasolid as a result of the operation.
- Supply replacement curves in cases where the existing geometry is not considered accurate.

These options are particularly useful when cleaning up parts that have been imported into Parasolid, for example as a second stage after fixing up some of the geometry of the part.

Parasolid can also transform the geometry of sets of faces in a body. *Figure 5–24* shows a body in which one face has been translated and two have been rotated.

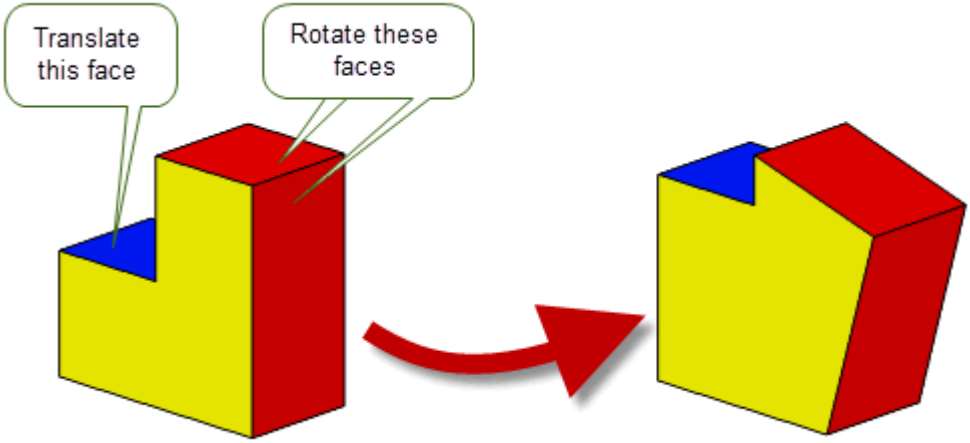


Figure 5–24 *Transformations applied to a list of faces*

5.6 Spinning and sweeping entities

You can spin many types of entity around an axis or sweep them in a specified direction, in order to create a new entity. For example, you can create a cylinder by sweeping a circular sheet along a path. Using spin and sweep (together with other operations) on profile bodies is a technique commonly used to create bodies from scratch.

Spin and sweep are described in more detail in Chapter 9, “Building Bodies from Profiles”.

5.7 Generic face change operations

Parasolid provides a generic face change operation that lets you perform any combination of the following local operations on any set of faces in a body in a single call:

Operation	Description
Offset	Offset any face by a specified distance.
Taper	Replace the surface of any face with a taper surface.
Transform	Transform the geometry of a face or set of faces.
Replace	Replace the geometry of any face with a new surface.
Blend	Reblend any face to remain consistent with its underlying faces.
Patch	Replace faces by patching in a set of replacement faces.
Deform	Locally deform a set of target faces on a body while preserving and adjusting existing features.
Radiate	Radially displace a rotational surface of a face.

Providing a generic operation has two major advantages:

- You can perform modelling tasks that would not be possible if you used a series of separate local operations, because the model would become invalid in between those operations.
- If you are performing a series of local operations, performing a single combined operation is often faster.

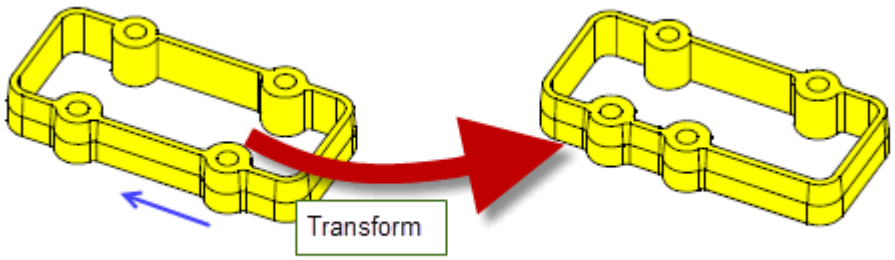


Figure 5–25 *Performing operations to several faces within a single body*

Working with Sheets and Wires 6

6.1 Introduction

Many Parasolid operations work equally well with sheet, wire and solid bodies. This chapter describes Parasolid functionality that is specifically aimed at modelling with wire bodies and sheet bodies, or whose behaviour with wire and sheet bodies is significantly different to its behaviour with solid bodies. Although Parasolid's primary focus is creating solid models, there are many occasions when you need to work with sheets or wires. For example:

- You may need to import data from another modeler that is expressed in terms of surface data, rather than solid. This data needs to be imported, cleaned up and then represented in such a way that the user can continue to model with it in Parasolid.
- You often need to create wire or sheet profiles that are going to be used as the basis of further modelling operations, as described in Chapter 9, "Building Bodies from Profiles".
- You may want to provide your users with a workflow that lets them work with sheet data, converting it to solid data only at specific points in the modelling process.

The functionality described in this chapter makes all of these techniques possible.

- Section 6.2, "Sheet modelling", describes functionality specifically aimed at sheet modelling. This is particularly useful when importing surface data into Parasolid.
- Section 6.3, "Wire modelling", describes functionality aimed at wire modelling. Much of this functionality is analogous to the functionality available for sheets.
- Section 6.4, "Creating profiles", explains how you can create profile bodies. These are acorns, wires, or sheets that you use to create more complex bodies using further Parasolid functionality. There are many ways of modelling with profiles (and these are described in Chapter 9, "Building Bodies from Profiles"), but for now, think of them as representing a cross-section through a body that you want to create.

6.2 Sheet modelling

Despite the superiority of solid modelling applications such as those based on Parasolid, there are still CAD applications that rely on surface modelling techniques, particularly for parts containing free-form geometry. Models created in these applications often need to be imported into Parasolid-based applications, as described in Chapter 11, "Importing Foreign Data", and the sheet bodies resulting from such imports need to be cleaned up and reworked so that modelling on the body can continue.

Parasolid provides you with a wide variety of operations specifically aimed at sheet modelling, making it easy to create, modify, blend and "solidify" sheets. These complement the core solid modelling capabilities offered, enabling your application to tightly integrate the two so as to

provide advanced modelling systems that combine the strengths of both paradigms. This section describes this functionality:

- Section 6.2.1 describes how you can create and modify sheets, respectively.
- Section 6.2.2 describes how sheets can be blended.
- Section 6.2.3 describes how you can use sewing and knitting techniques to join sheets together.
- Section 6.2.4 describes other miscellaneous sheet operations that are available.

6.2.1 Creating and modifying sheets

Parasolid provides operations to let you create sheet bodies from scratch from a collection of faces. You can also create primitive sheet bodies such as circles, polygons and rectangles directly, by calling efficient dedicated functionality.

You can modify existing sheets by trimming them (making them smaller) and extending them (making them larger).

You can create a set of neutral (mid-surface) sheets from a solid body, as shown in *Figure 6–1*, or from faces of sheet bodies.

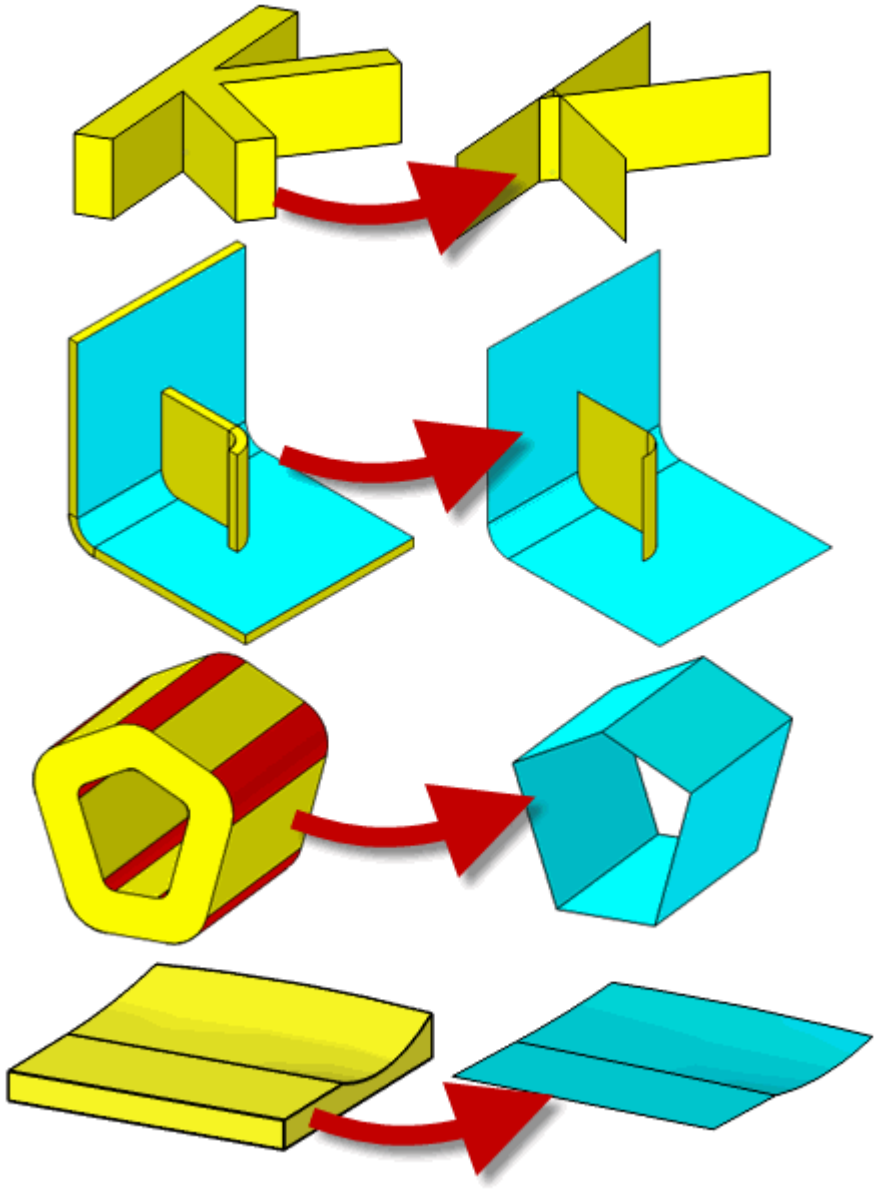


Figure 6–1 *Creating neutral sheets from a solid body*

6.2.1.1 Trimming sheets

Trimming sheets to a specified boundary is generally a two-part process:

- Imprint additional curves on the sheet to divide the sheet into at least two distinct sets of faces.
- Trim the sheet to the specified boundary, selecting which faces should be kept.

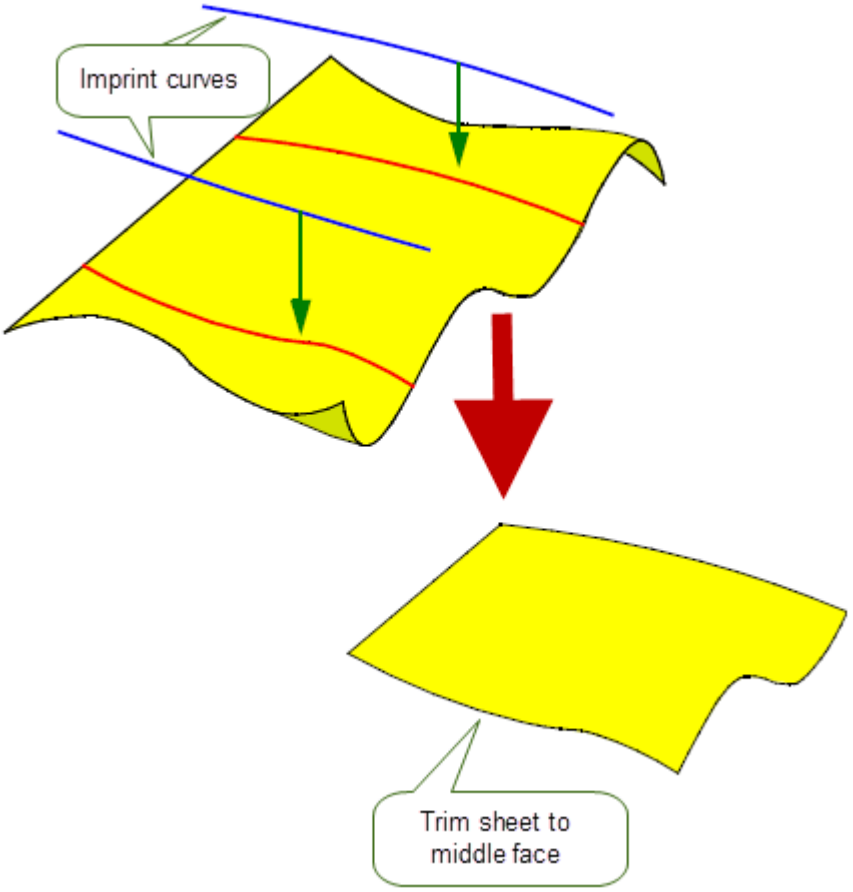


Figure 6–2 *Trimming sheet bodies*

There are many ways that you can imprint curves onto a body. The most common methods involve:

- Specifying the curves explicitly
- Imprinting a set of faces onto the sheet
- Intersecting a plane with the sheet

See Section 6.4.1 for more methods of imprinting. Parasolid can also join together disconnected parts of an imprinted curve automatically, thereby ensuring that sheets are reliably divided into face sets ready for trimming.

6.2.1.2 Extending sheets

As well as trimming sheets, you can extend sheets beyond their current boundaries, following the underlying surface wherever possible. When a sheet is extended to a point where no underlying surface is available, you can control the shape of the extension that is created.

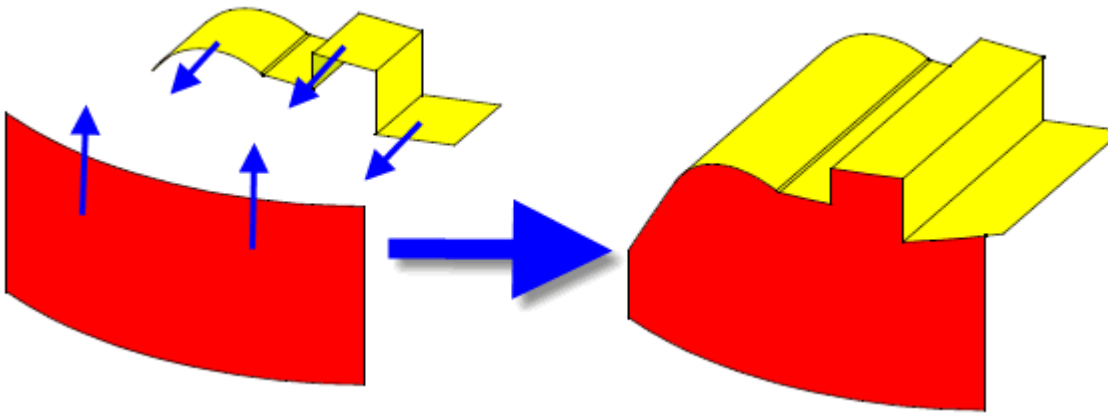


Figure 6–3 *Creating sheet bodies by combining sheet extension and boolean operations*

Parasolid can extend sheets either by a specific distance, or by extending until the sheet reaches a specified target body.

When extending by a specific distance, you can control whether to create a precise or a loose boundary. Creating a loose boundary can improve performance in cases where the exact boundary shape is irrelevant for your needs, for example if you intend to trim away some of the extension using subsequent modelling operations.

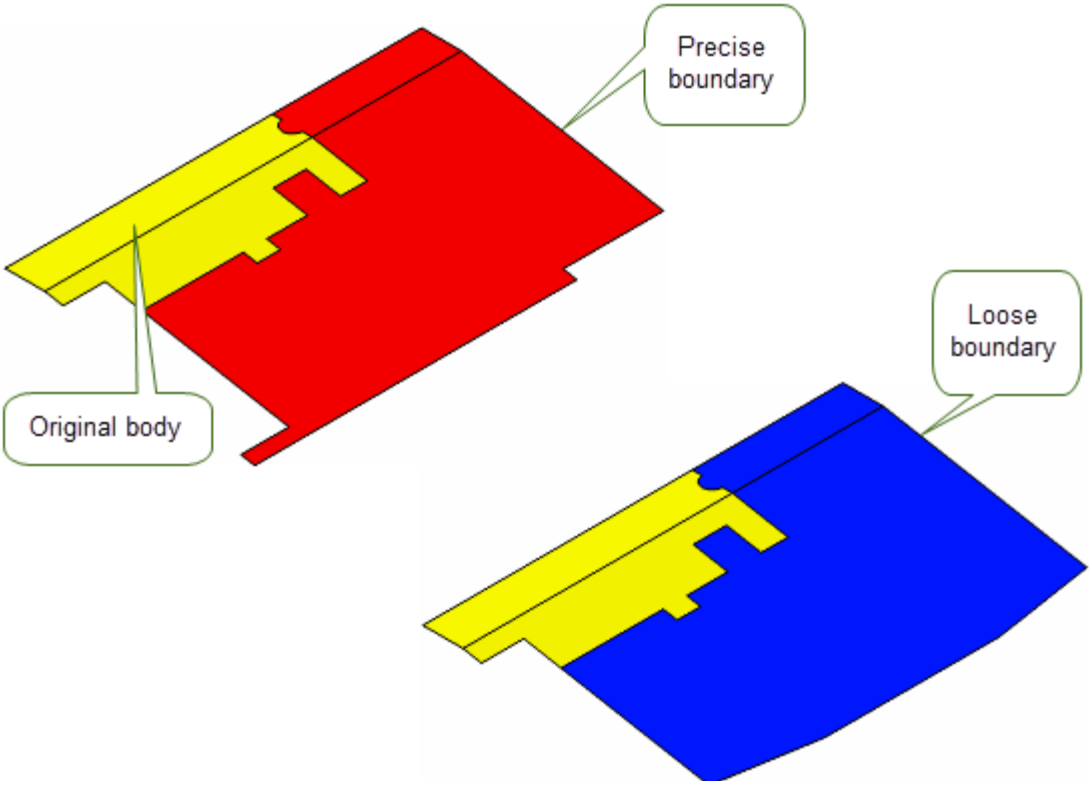


Figure 6–4 *Creating precise and loose boundaries when extending sheets*

When extending to a target, you can control whether to extend until the sheet first reaches the target, or until the sheet leaves the target, as shown in *Figure 6–5*

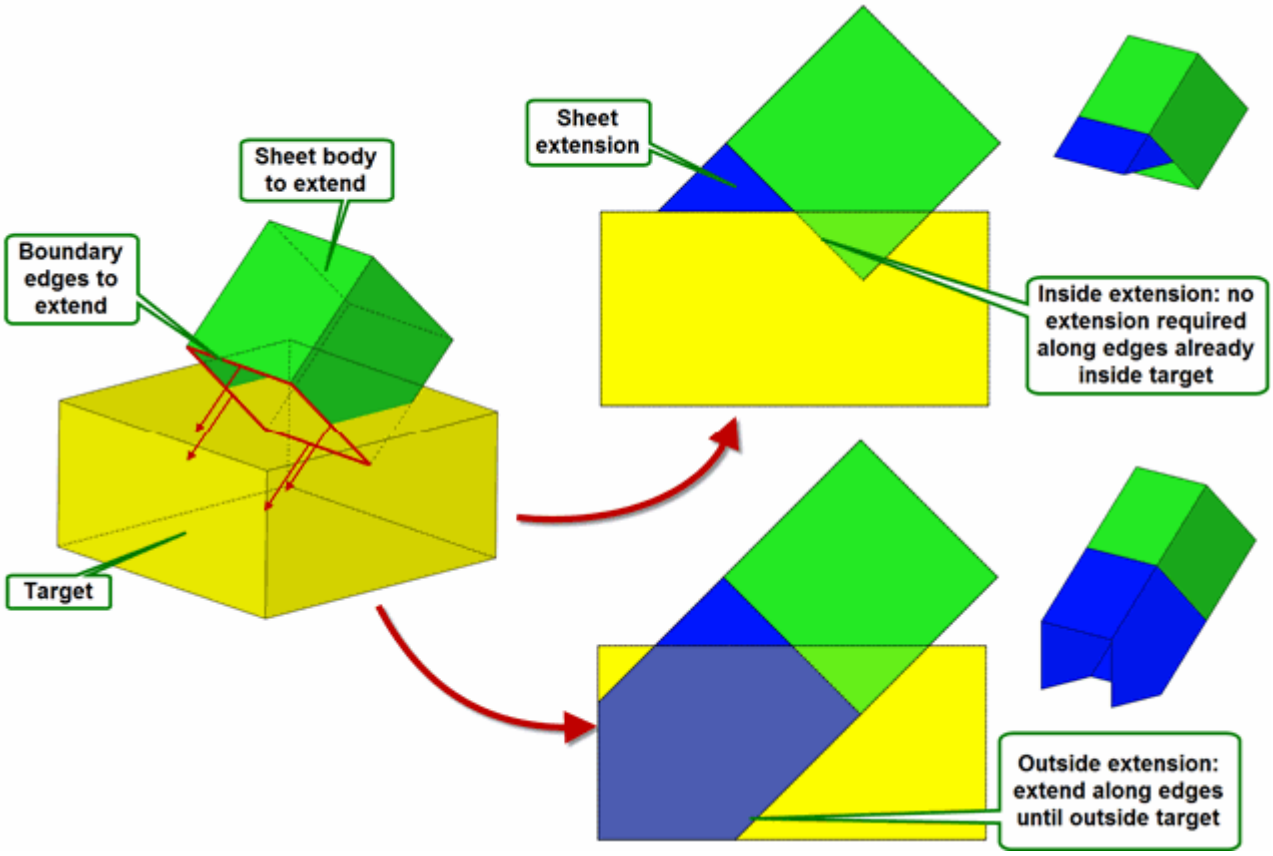


Figure 6–5 *Controlling the result when extending a sheet to a target body*

6.2.2 Blending sheets

Blending operations (described in more detail in Chapter 10, “Blending”) can be performed on sheets as well as solids.

- Separate sheets can be bonded by creating face-face blends between them, as shown in *Figure 6–6*.

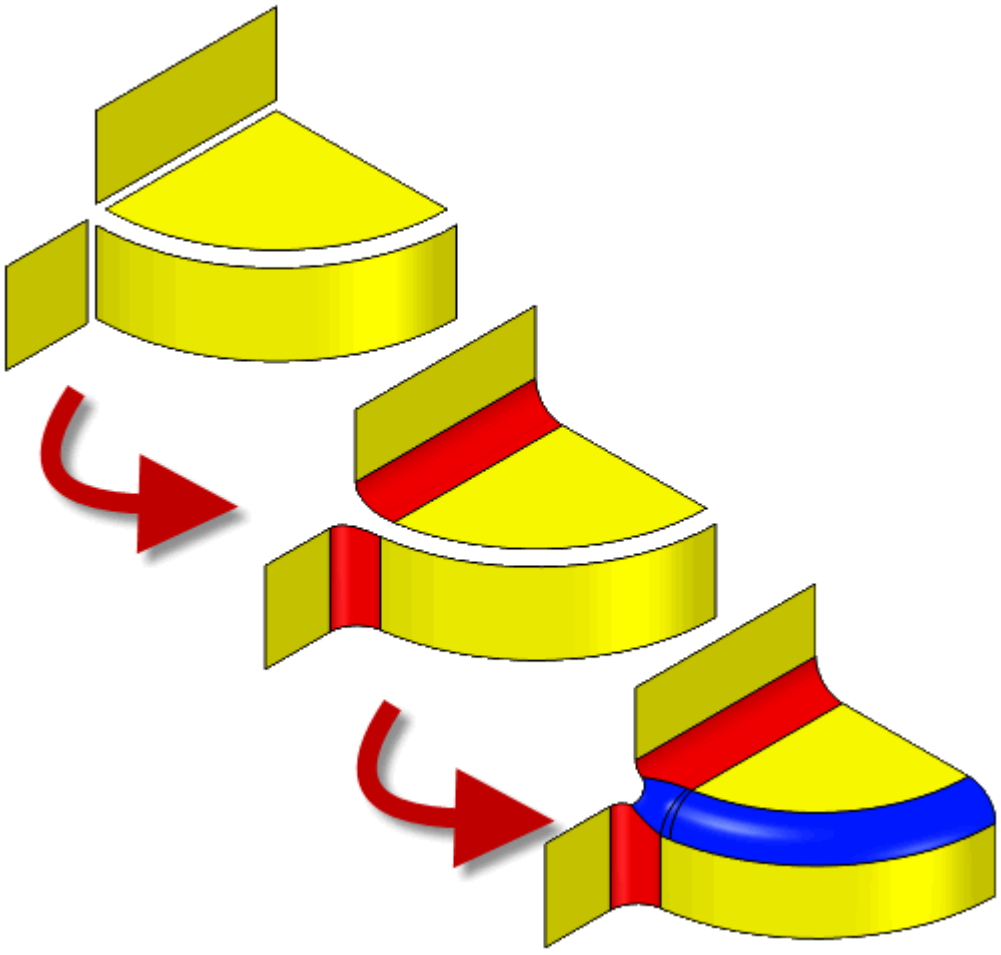


Figure 6–6 *Creating blends between sheets*

- The shape of a sheet can be modified by blending vertices at non-smooth points on a sheet boundary, as shown in *Figure 6–7*.

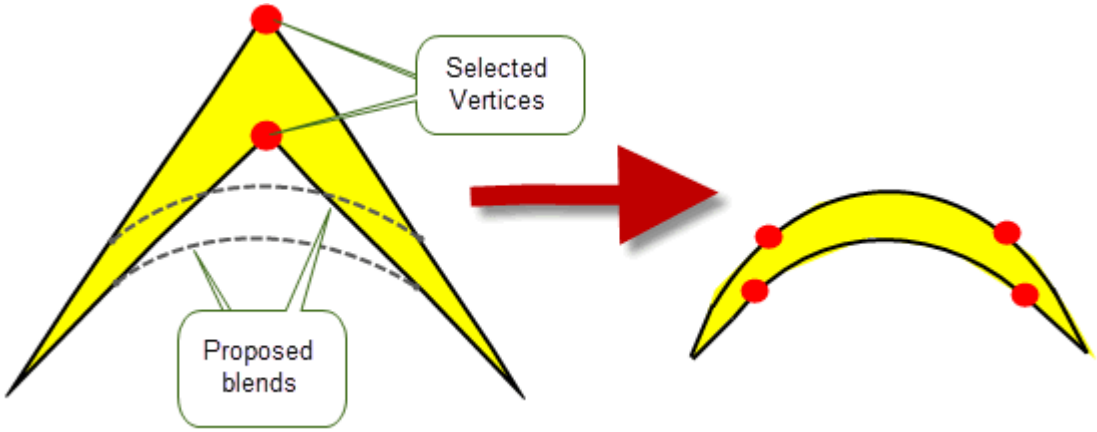


Figure 6–7 *Creating blends between sheets*

6.2.3 Sewing sheets

You can sew together sets of sheets that are geometrically close. This is particularly useful when importing trimmed surface data from another, non-Parasolid, application (see Section 11.2). In such cases, because Parasolid expects extremely accurate geometry and rich topological information, the result of an import can often be a set of disconnected sheets with no information available about how each sheet should connect to the others. Parasolid make it possible to heal such bodies without requiring any topological information.

A collection of sheet bodies can be sewn together by gluing them where their edges meet, resulting in a single connected body. Parasolid sews together all pairs of edges which are less than a specified distance apart, as shown in *Figure 6–8*.

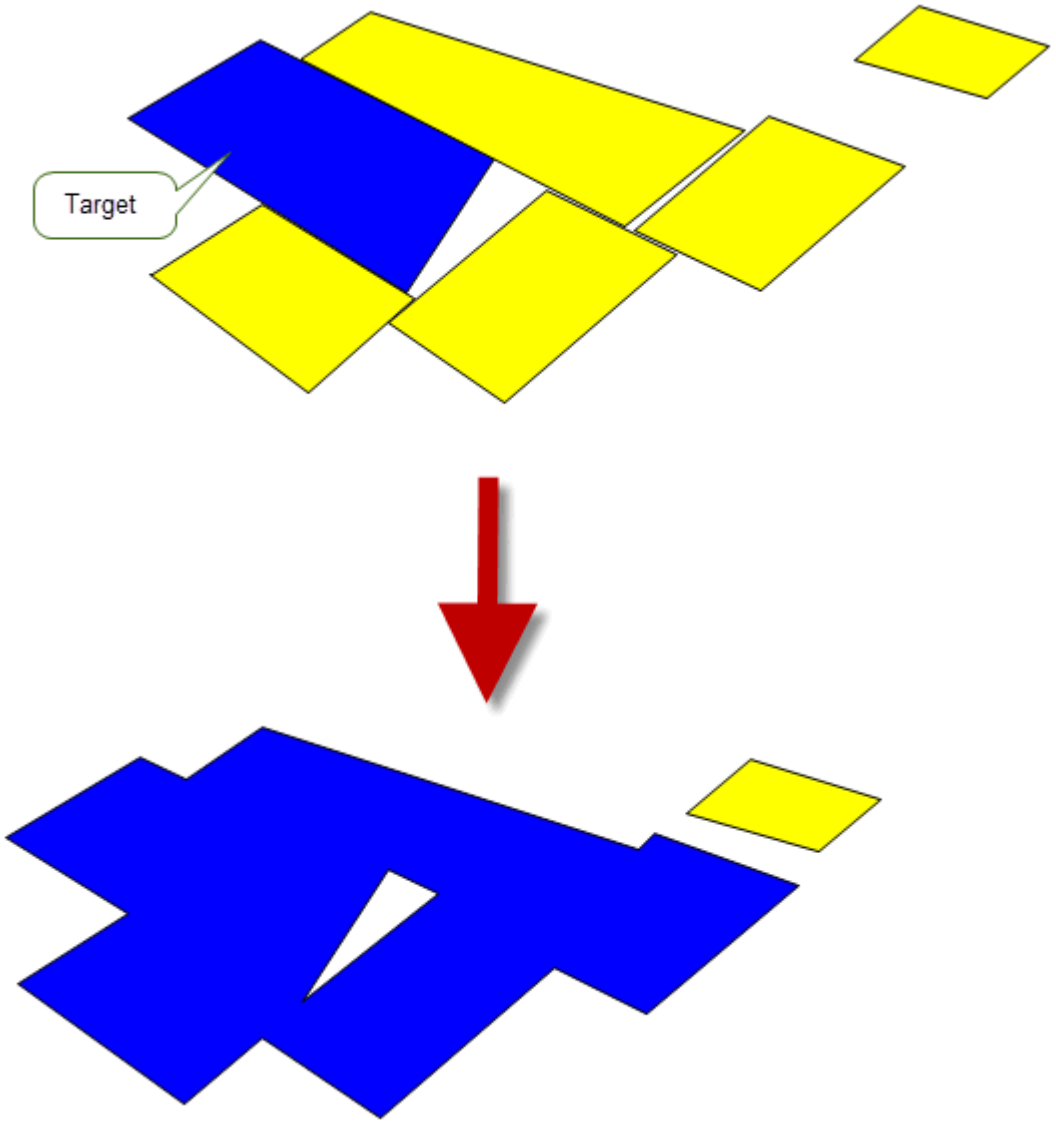


Figure 6–8 *Sewing together collections of closely spaced sheets.*

Several options are provided, including:

- Whether to remove duplicate sheets in the result.
- Whether to return sheet or solid bodies (in cases where sewing creates a void region).
- The ability to optimise performance and reliability when sewing sheets that are part of an assembly

In addition, Parasolid supports **incremental sewing**. This is a technique whereby sewing is repeated several times on the same set of sheets, increasing the maximum gap that will be sewn across with each iteration. The effect of this is to maximise the preservation of small details on parts where significant details are present over a wide range of scales.

6.2.3.1 Knitting bodies together

Parasolid also supports **knitting** operations. Knitting is similar to sewing, but can be used in cases where you want to combine sheets with solid bodies, or combinations of bodies. Parasolid can knit entities that are coincident to within a tolerance set by your application.

6.2.4 Other operations with sheets

Parasolid supports a number of other miscellaneous operations with sheets. These include:

- Thickening sheets so as to produce solid bodies (described in more detail in Section 5.2.3)
- Deleting holes
- Replacing the surfaces of faces in sheet bodies.

6.3 Wire modelling

As with sheet bodies, Parasolid provides specific functionality for modelling with wire bodies. This includes:

- Creating wire bodies from sets of curves or edges.
- Creating faces to fit wire bodies.
- Blending vertices on wire bodies.
- Splitting wire bodies.
- Offsetting planar wire bodies.
- Performing boolean unite and subtract operations on wire bodies.
- Uniting wire bodies with sheet bodies.

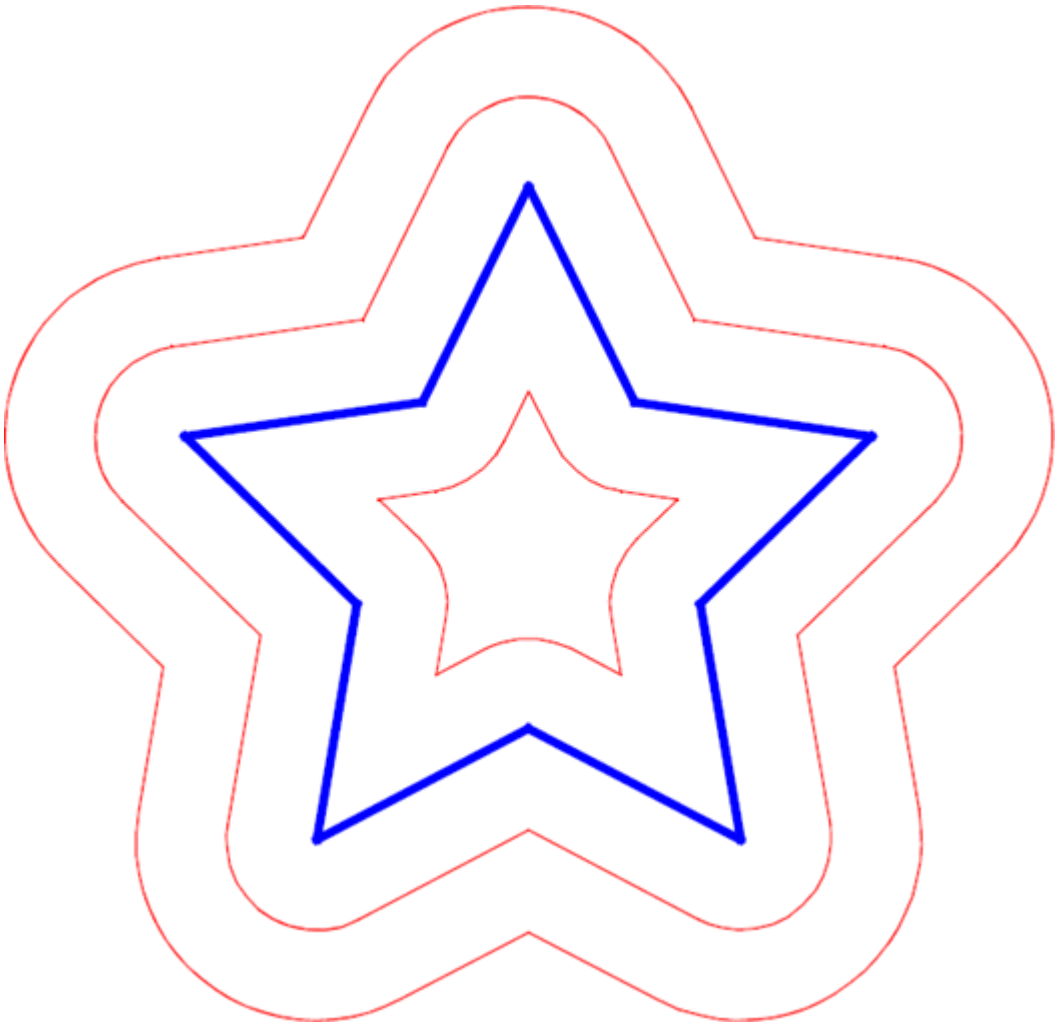


Figure 6–9 *Various offsets of a star-shaped wire body*

6.4 Creating profiles

A profile is a general term for a simple zero-, one- or two-dimensional body that is subsequently used to create new bodies with free-form geometry. Creating profiles is an important technique in solid modelling, and Parasolid provides functionality to make it simple.This section describes how you can create profiles from a set of curves, by creating sheet bodies directly or from existing entities.

There are four general approaches to creating profiles:

- Imprint a series of curves to create a sketch. See Section 6.4.1.
- Create primitive sheet bodies directly using built-in functionality. See Section 6.4.2.
- Create profiles from existing edges or curves in a body. See Section 6.4.3.
- Create profiles by finding the outline of an existing body when viewed from a particular direction. See Section 6.4.4.

Chapter 9, “Building Bodies from Profiles”, tells you about how you can use profiles to create new bodies.

6.4.1 Imprinting

The most basic way of creating a profile is to build it piecemeal by imprinting a series of curves:

- **You can scribe new curves directly onto existing entities.** You can create new faces by scribing onto a solid body, or you can extend a wire body by scribing onto the wire body or onto its region.
- **You can project existing curves onto a body.** Different effects can be achieved, depending on whether you project the curve along the normal of the faces onto which you are imprinting, or whether you project the curve along a specified vector.
- **You can also imprint projected curves onto a body.** Curves are first projected then imprinted on the body.

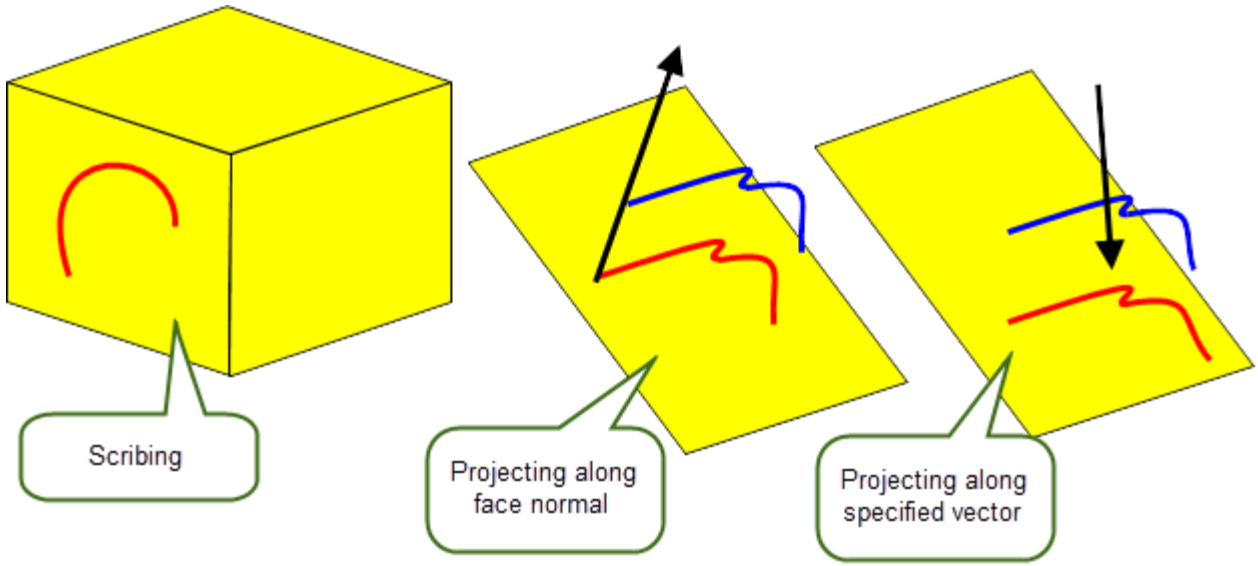


Figure 6–10 *Imprinting curves by scribing and projecting*

6.4.2 Creating primitive sheet bodies

As mentioned in Section 6.2.1, Parasolid provides a range of functionality to let you create primitive sheet bodies such as circles, polygons and rectangles.

6.4.3 Using existing entities

Parasolid provides tools to create a wire profile from a set of existing edges or curves in a single operation, and to turn a closed wire into a sheet body if required.

6.4.4 Outline curves, perspective outlines, spun outlines and shadow curves

Another way of creating profiles is to use outlines. This is specialised functionality that creates a profile from the outline of an existing body. Parasolid supports the following types of outline:

- An **outline curve** is the outline created by drawing around the boundary of a set of bodies, as seen from a specified view direction. This is illustrated in *Figure 6–11*. Parasolid also supports the creation of outlines of wire components or sheet components that are edge-on when viewed in a specified direction as illustrated in *Figure 6–12*.

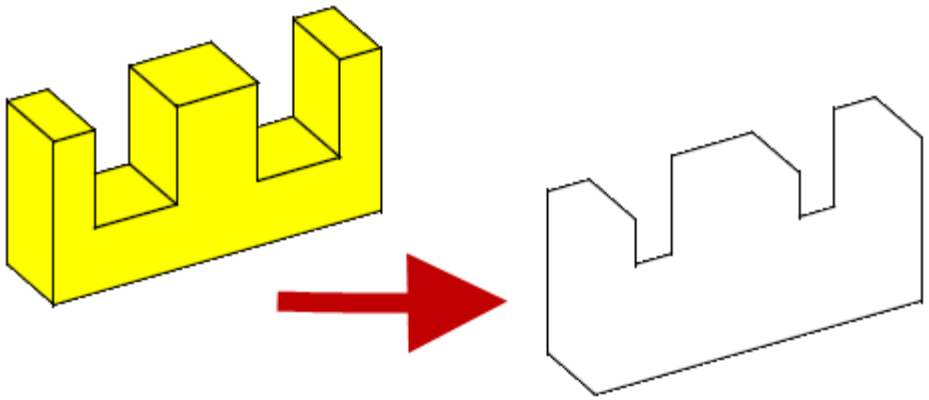


Figure 6–11 *Creating outline curves for a body*

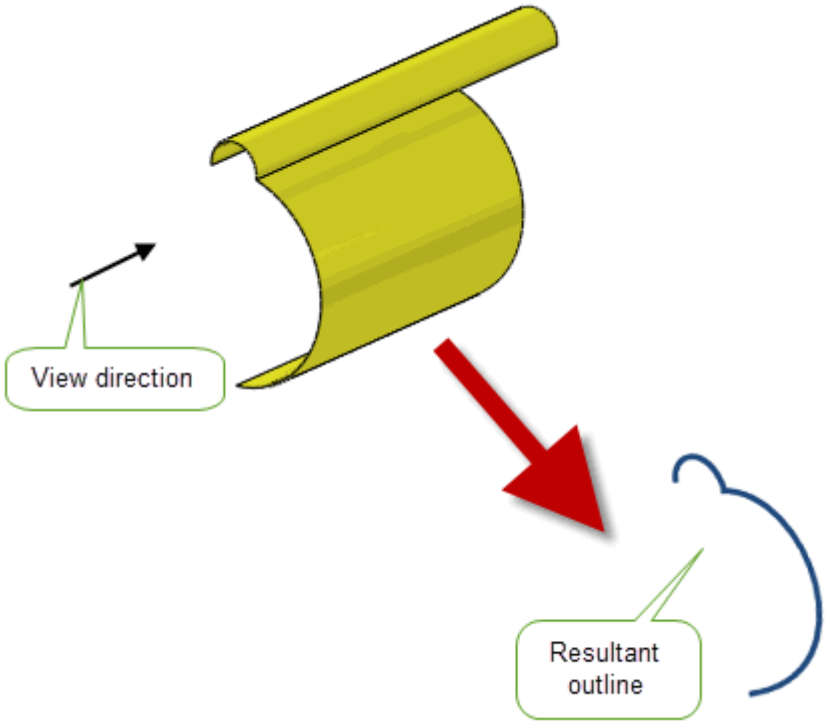


Figure 6–12 *Creating an outline of an edge-on sheet body*

- A **perspective outline** is created by drawing around the boundary of a body or a set of bodies, as seen from a perspective view point as shown in *Figure 6–13*. In addition, Parasolid gives you the option to clip these outline curves so only those that fall within a specified range are generated as shown in *Figure 6–14*.

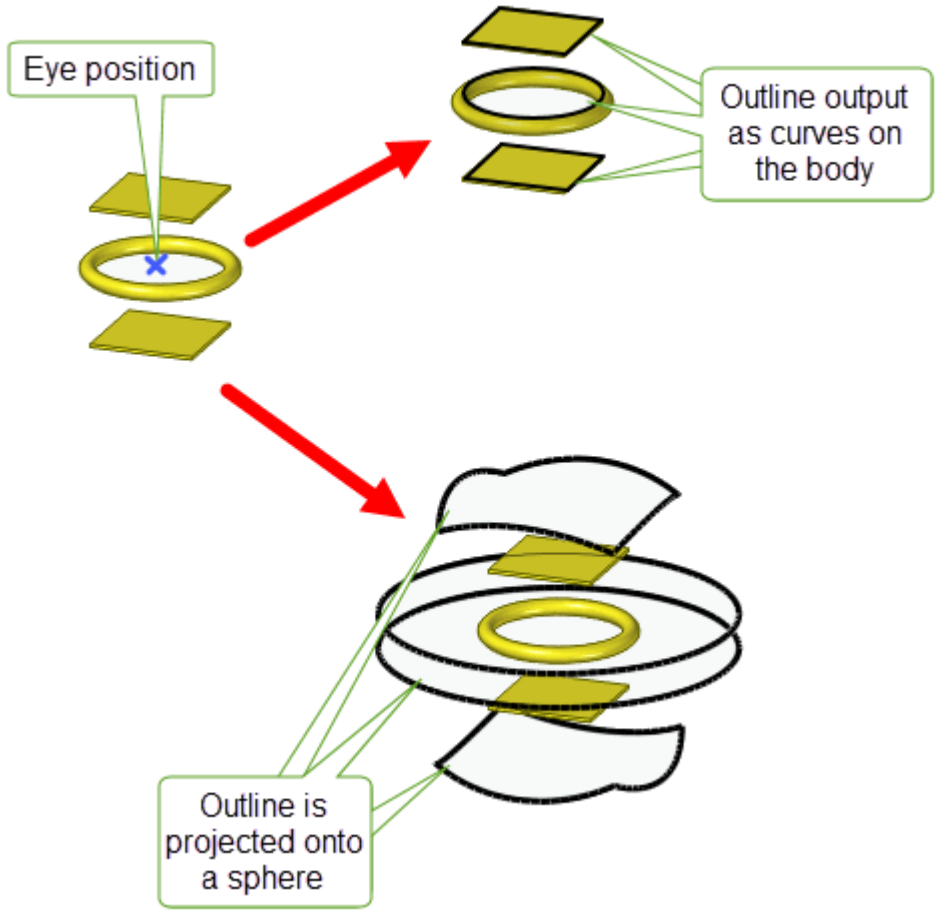


Figure 6–13 *Creating a perspective outline for a body*

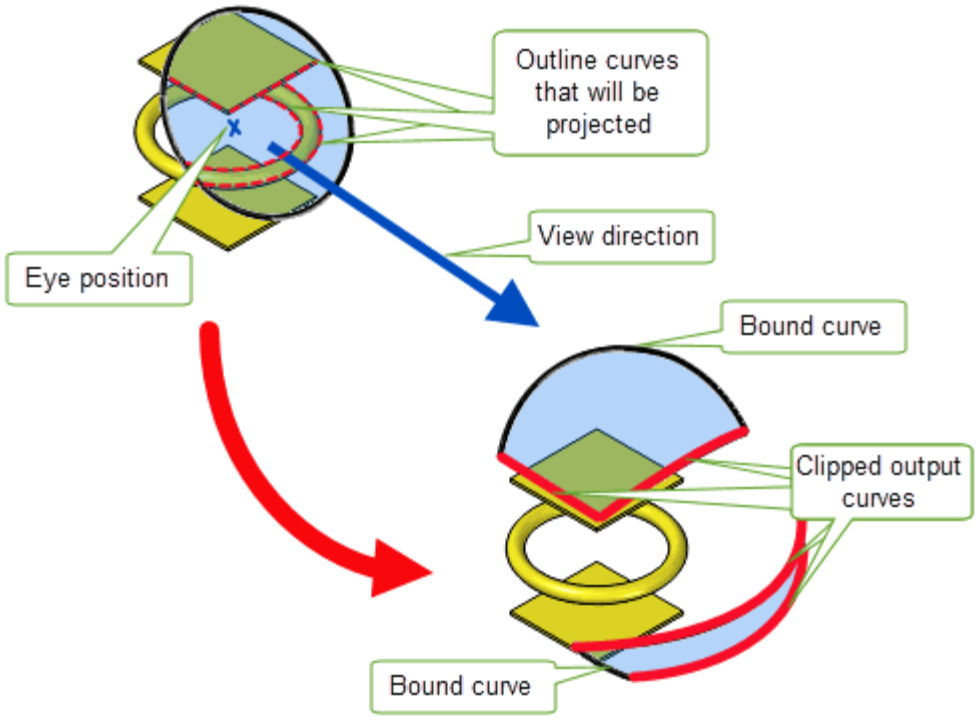


Figure 6–14 *Clipping perspective outline curves*

- A **spun outline** is the rotational equivalent of an outline curve. Consider spinning a set of bodies around an axis: the spun outline represents a cross-section through the resulting spun body. If the spun outline is projected onto a plane that passes through the spin axis, its resulting profile describes the hole that would need to be cut in order for the bodies to pass through, as shown in *Figure 6–15*. This functionality also enables you to create outlines of features which are hidden by other portions of the model as shown in *Figure 6–16*.

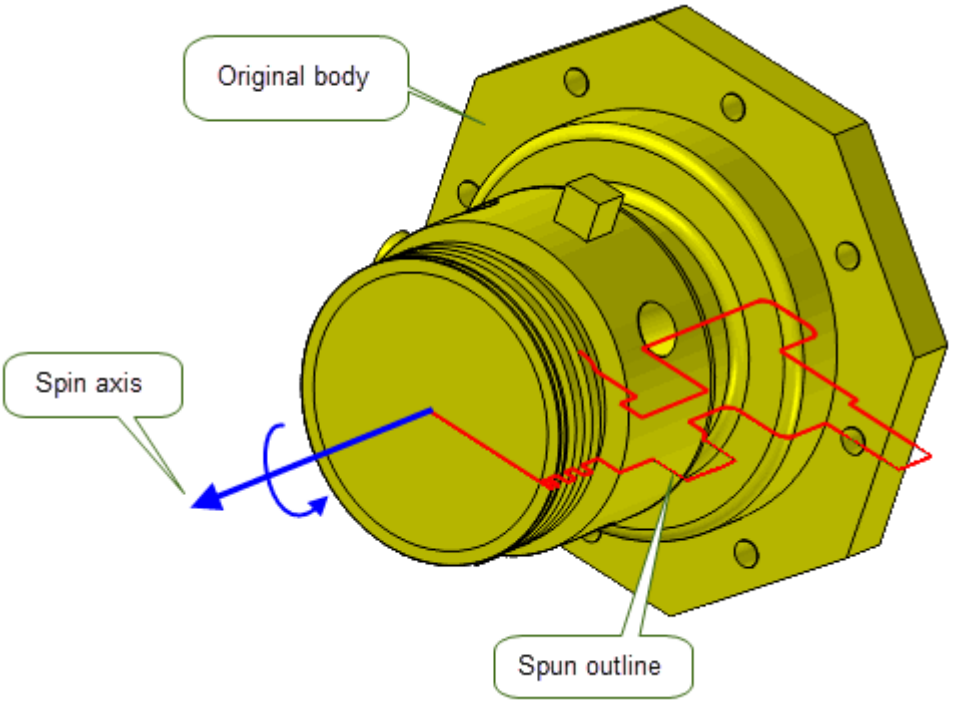


Figure 6–15 *Creating spun outlines for two bodies*

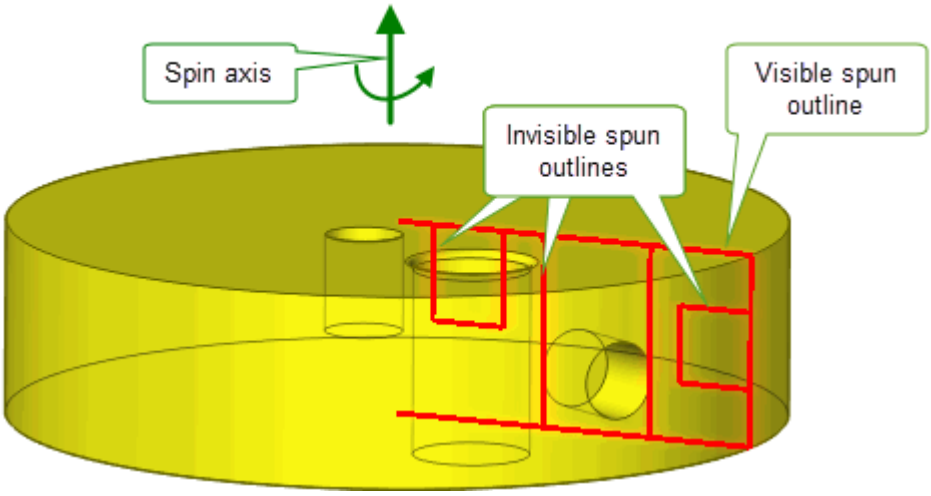


Figure 6–16 *Creating invisible outlines of a body with holes*

- A **shadow curve** is basically an outline curve that is projected directly onto a body. Shadow curves are the curves that are projected onto a set of bodies when viewed from a specific direction, splitting the bodies into visible and invisible areas when they are viewed from that direction. They are known as shadow curves because they represent areas of light and dark if you assume that a light source is placed at the view direction.

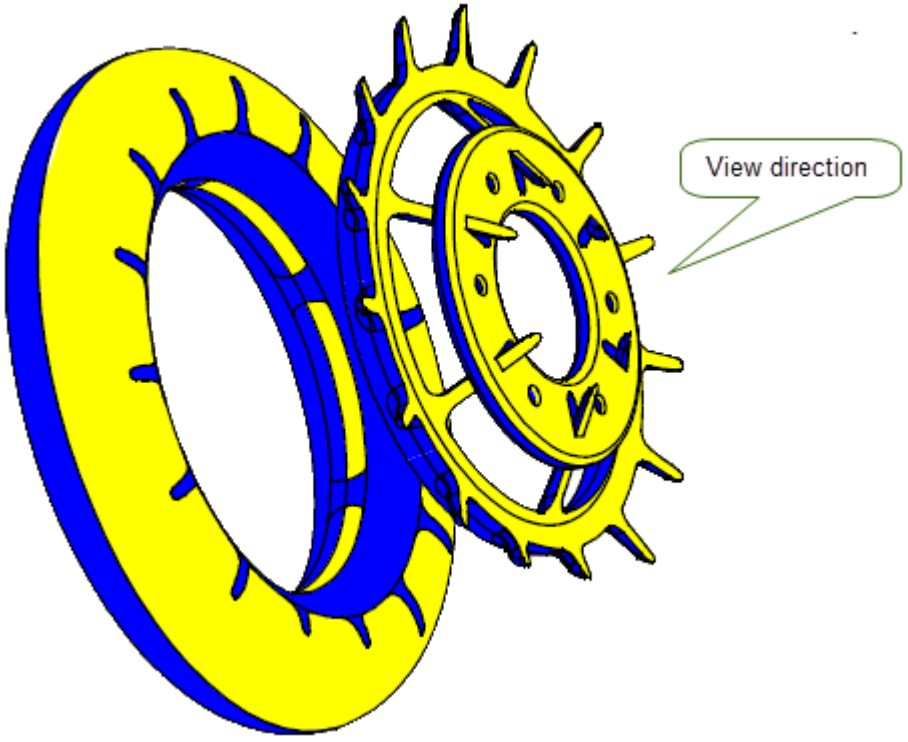


Figure 6–17 *Shadow curves on a set of bodies*

B-Spline Curves and Surfaces

7

7.1 Introduction

Wherever possible, Parasolid uses analytic geometry in its models. For example, curves such as circles, ellipses and lines, and surfaces such as spheres, tori and cones are represented using geometry that can be described by an equation. Using analytic geometry as much as possible gives Parasolid unparalleled speed and economy of storage.

There are times, however, when geometry cannot be represented by analytic entities. If possible, other types of geometry are used (such as swept or spun surfaces), but when even this is not possible, NURBs-based (**N**on-**U**niform **R**ational **B**-**s**plines) **B-geometry** is used instead. Parasolid offers two forms of B-geometry: **B-curves** and **B-surfaces**. Unlike analytic geometry, B-geometry is defined over a finite region of space.

B-curves and B-surfaces are fully integrated into Parasolid, so you can attach them to edges and faces, and apply any relevant Parasolid operation to them, just as you would any other type of curve or surface. This integration is so tight that in most circumstances there is no reason why users even need be aware they are using B-geometry. However, in addition, Parasolid provides functionality to let you work explicitly with B-geometry. This chapter describes the functionality that is available.

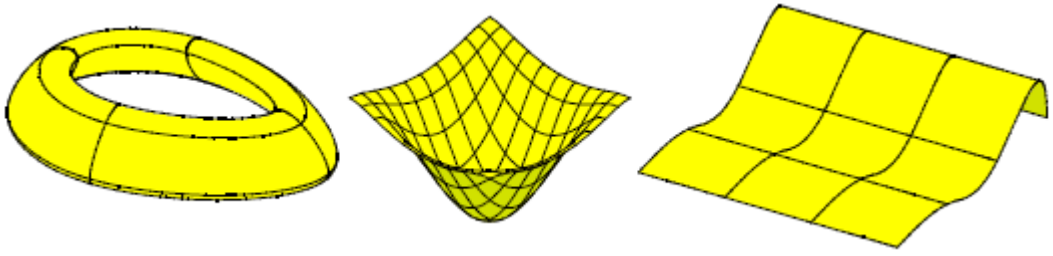


Figure 7–1 *Examples of B-surfaces*

7.2 Creating B-geometry

Parasolid's use of B-geometry is kept to a minimum; wherever possible, other types of geometry are used during operations that create geometry. However, if you explicitly need to create B-geometry, Parasolid provides the following methods:

Method	Description
Supplying B-spline data	You can supply B-splines (in the form of control points and knot vectors), and Parasolid can create B-curves and B-surfaces directly from this data.
Supplying piecewise data	You can supply independent segments of surface or curve data, in either Bezier, Hermite, polynomial or Taylor series format. Parasolid can piece these individual segments together to create the appropriate B-geometry.
Splining	You can create B-geometry with data that is not as rich as B-spline or piecewise data by using Parasolid's general splining functionality. You supply a series (for a B-curve) or a mesh (for a B-surface) of points, and Parasolid creates a curve or surface either by interpolating through the supplied points or by fitting to them within a specified tolerance. You can control the final appearance of the geometry using controls such as clamping conditions.
Curve and surface fitting	You can build B-curves and B-surfaces from a set of points sampled from existing curves or surfaces. This functionality is principally used to improve the quality of existing geometry; for example, after importing curve or surface data into Parasolid, you may decide to rebuild data that is not C2-continuous using these functions. You can also create B-geometry using only the definition of a curve or surface, rather than actual geometric data.
Constrained surfaces	You can create B-surfaces from a cloud of points and, optionally, normals, which constrain the shape of the resulting B-surface. To further constrain the surface, you can also supply parameterisation information.
Converting from other geometry	You can convert any piece of geometry into B-geometry. This can be useful if you want to output your data for use in non-Parasolid based applications that use less efficient methods of storing data.

7.3 Modelling with B-geometry

While performing modelling operations, Parasolid treats B-geometry just like any other type of geometry, allowing you to apply relevant operations to models that have attached B-curves and B-surfaces. In addition, Parasolid offers a range of modelling functionality that is intended specifically for use with B-geometry.

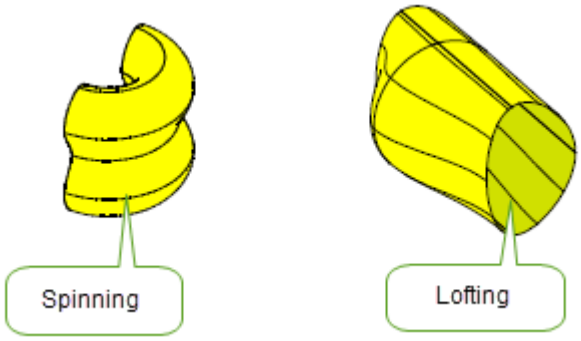


***Figure 7–2** modelling with B-geometry*

Some of the modelling operations that can be performed with B-geometry are as follows:

Operation	Description
Sweeping and spinning	Parasolid can create B-surfaces by sweeping or spinning a previously created B-curve.
Lofting	Parasolid can create B-surfaces by lofting between a set of pre-defined B-curves. Parasolid's generic lofting functionality is described in more detail in Chapter 9, "Building Bodies from Profiles". Parasolid contains a variety of options to control the shape of the lofted surface.
Extracting isoparam curves	Parasolid can create B-curves along constant parameter lines from a B-surface.
Curve joining	Parasolid can join a number of B-curves to form a single B-curve.

Parasolid allows you to simplify the geometry in a body by converting rational B-curves and B-surfaces to non-rational B-curves and B-surfaces. You can also convert non-rational B-curves to lines or circles, and non-rational B-surfaces to planes, cylinders, cones, spheres or tori.

Convergent Modeling

8

8.1 Introduction

CAD,CAM and CAE applications are increasingly relying on facet bodies to support workflows in growth areas like additive manufacturing, reverse engineering and virtual simulation. A facet body is a body with Parasolid topology that references geometric data composed of mesh and polyline data rather that classic Parasolid surface and curve geometry.

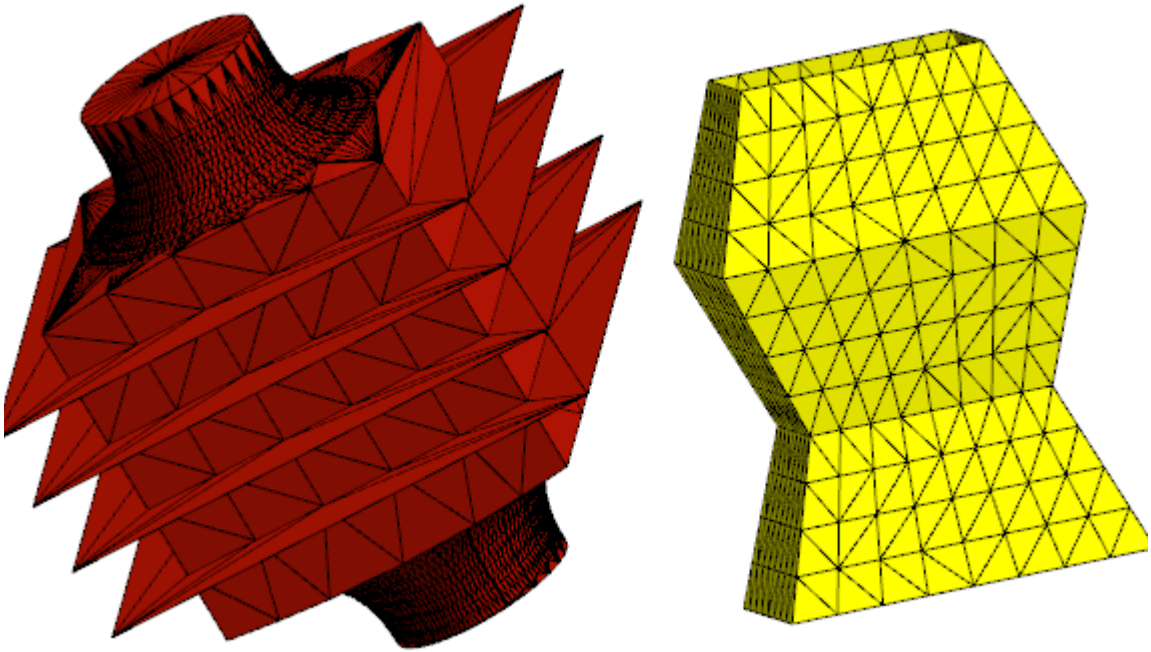


Figure 8–1 *Examples of facet bodies*

In order to perform classic modelling operations on a body, traditional CAD software needs to convert facet bodies to and from accurate CAD geometry. This conversion can result in additional complexity and errors in the result body. Therefore being able to perform classic modelling operations directly on facet bodies can dramatically improve your workflow.Parasolid gives you this ability thereby letting your application perform topological operations, such as booleans, sectioning, and offsetting on facet bodies.

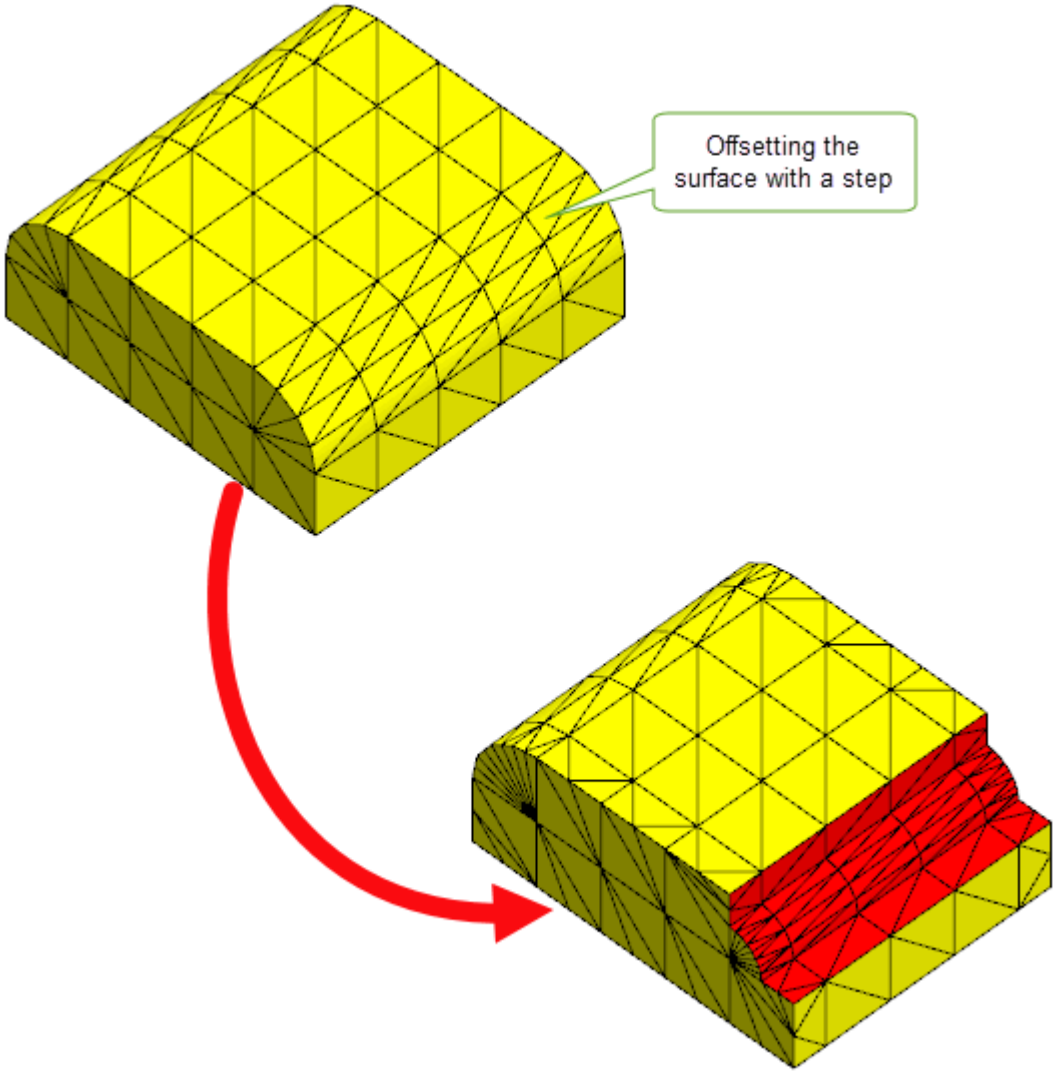


Figure 8–2 *Offsetting a surface of a mesh with a step*

Parasolid Convergent Modeling greatly extends scope beyond pure facet bodies by providing full support for models containing an arbitrary mix of facet and classic geometries. Convergent Modeling support for mixed bodies enables more efficient workflows in a wide range of applications. For example, in reverse engineering processes it can be used to add precise analytic surfaces (e.g. machined holes) to scanned data of parts for re-manufacture. It can also be used in restoring design intent by adding precise, classic mating surfaces to an imported model or one that has undergone facet-based topology optimisations.

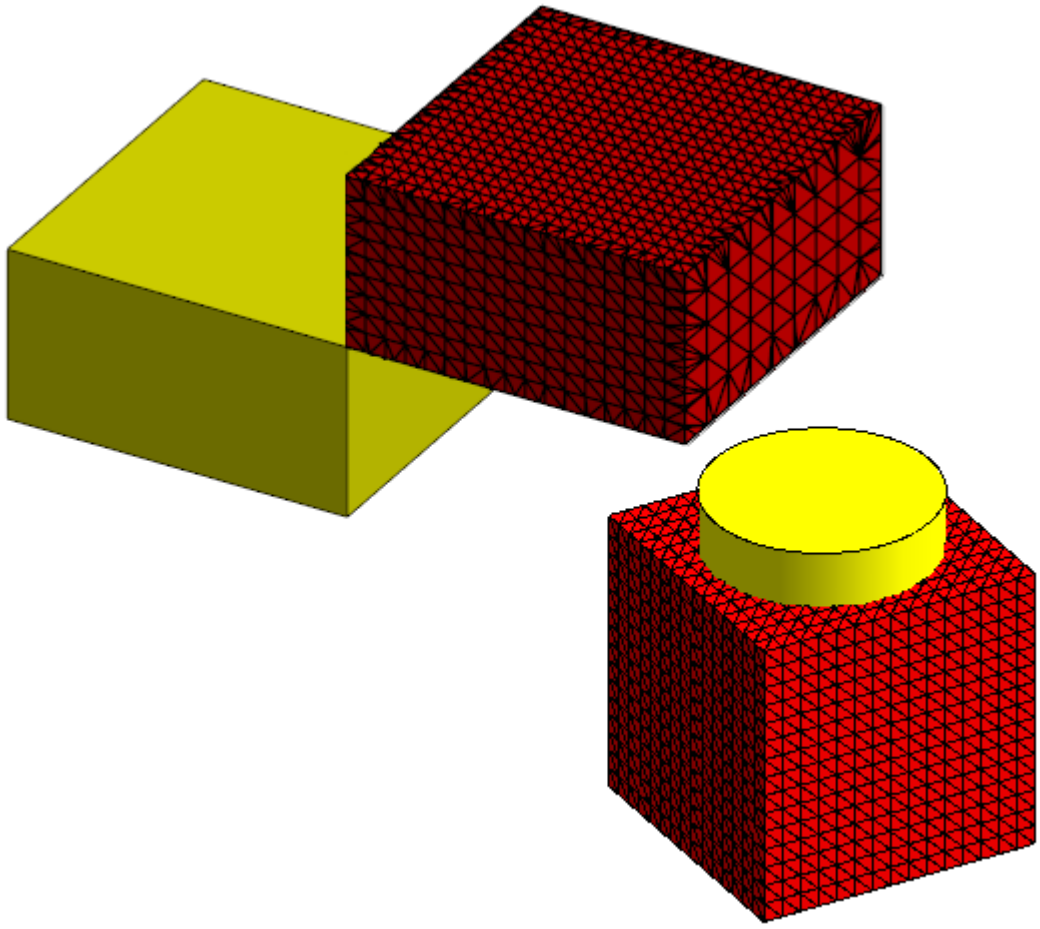


Figure 8–3 *Example of a mixed bodies*

8.2 Managing facet geometry

Parasolid provides a range of APIs for creating facet geometry and for converting between facet and classic geometry.

- You can convert a body containing classic geometry to a facet body as illustrated in *Figure 8–4*.

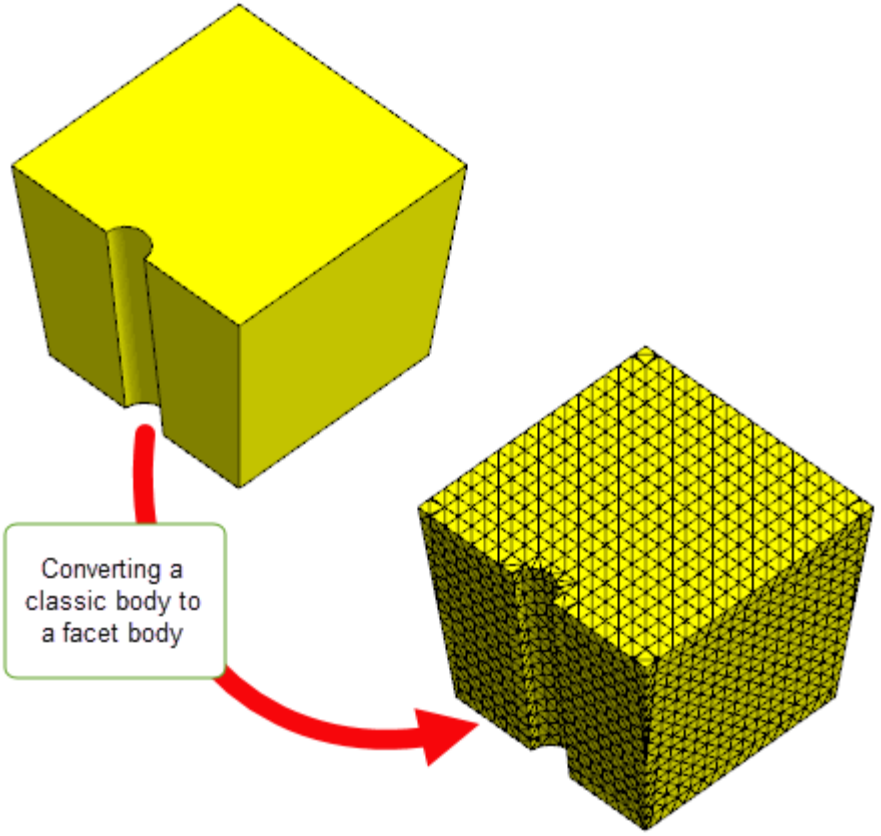


Figure 8–4 *Converting a classic body to a facet body*

- Parasolid can create a mixed body by converting selected classic topology to facet geometry as illustrated in *Figure 8–5*.

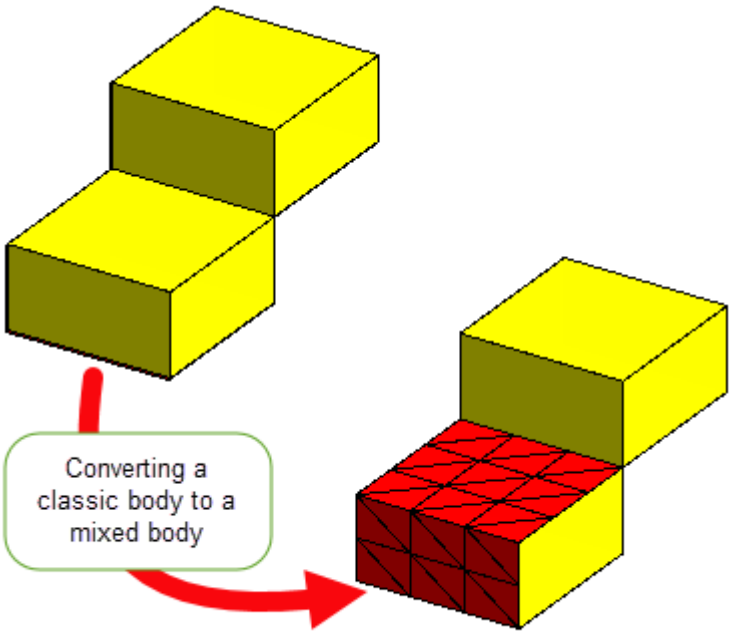


Figure 8–5 *Converting a classic body to a mixed body*

- You can create a mesh from an existing mesh, a subset of meshes, or a selection of mfacets taken from multiple meshes.
- Parasolid can convert mesh data into classic geometry in the form of a single trimmed surface as shown in *Figure 8–6*.

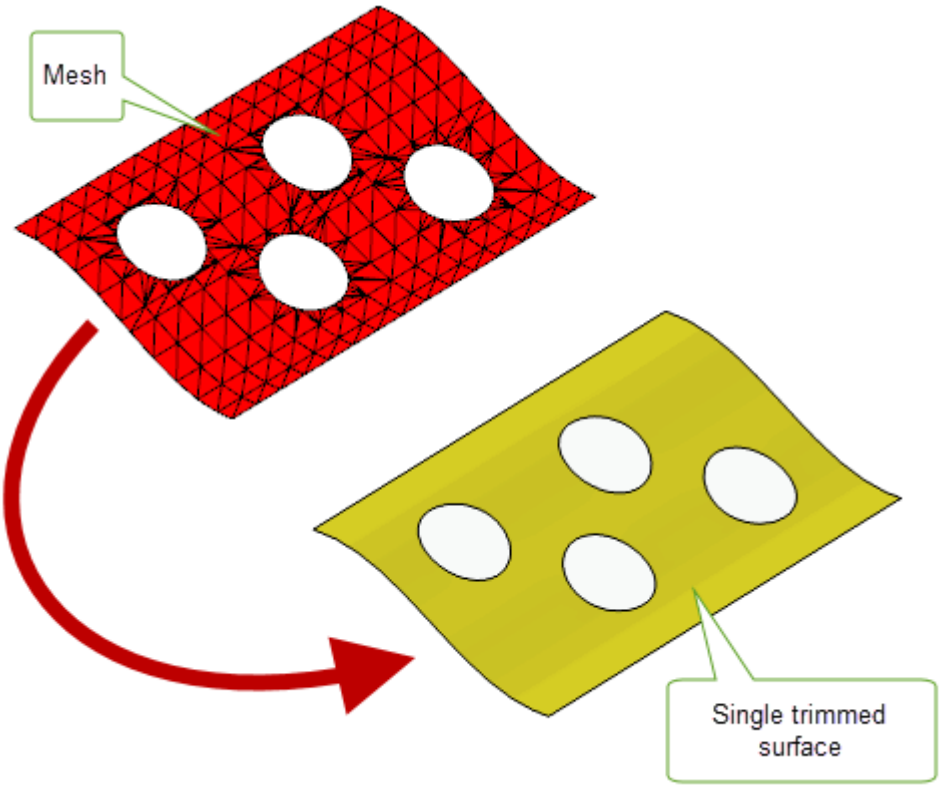


Figure 8–6 *Creating a trimmed surface from a single mesh*

8.3 Mesh-specific operations

There are fundamental differences between classic bodies and meshes which mean that some areas of functionality need to be treated differently in order to produce consistent and accurate results. Parasolid provides a range of mesh-specific operations which enable you to perform the following operations on meshes:

- Evaluating the parameters and normals of a mesh
- Identifying the perimeters of a selected group of mesh facets as shown in *Figure 8–7*

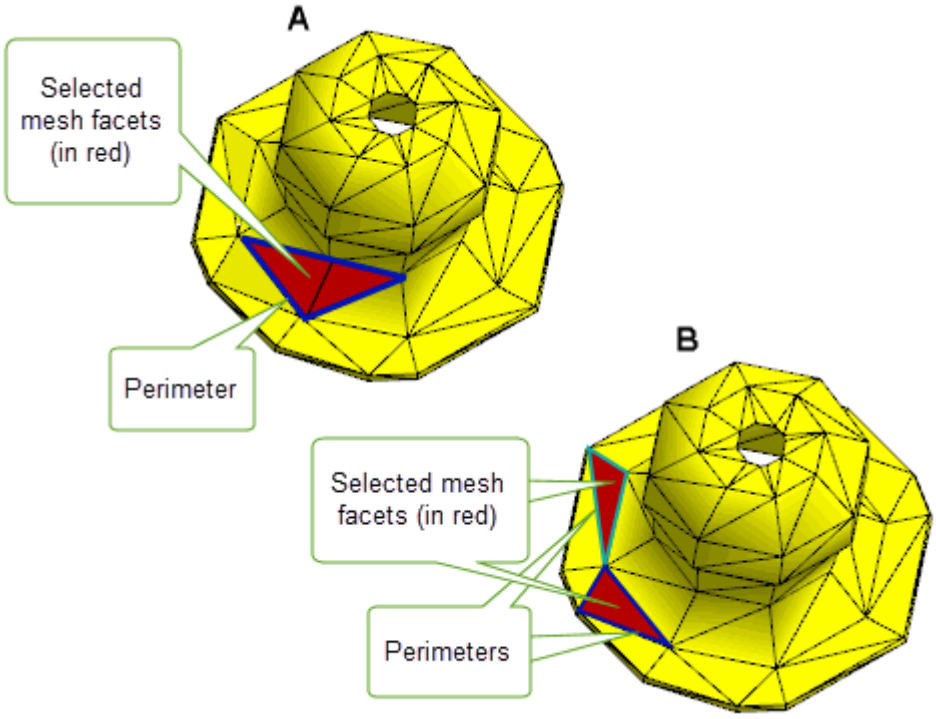


Figure 8–7 *Identifying perimeters of selected groups of facets on a mesh*

Parasolid also provides functionality for:

- Enquiring the facet topology of a mesh
- Checking the mesh for defects such as invalidities or other sub-optimal mesh properties whose removal may be desirable
- Repairing holes in the mesh as shown in *Figure 8–8*

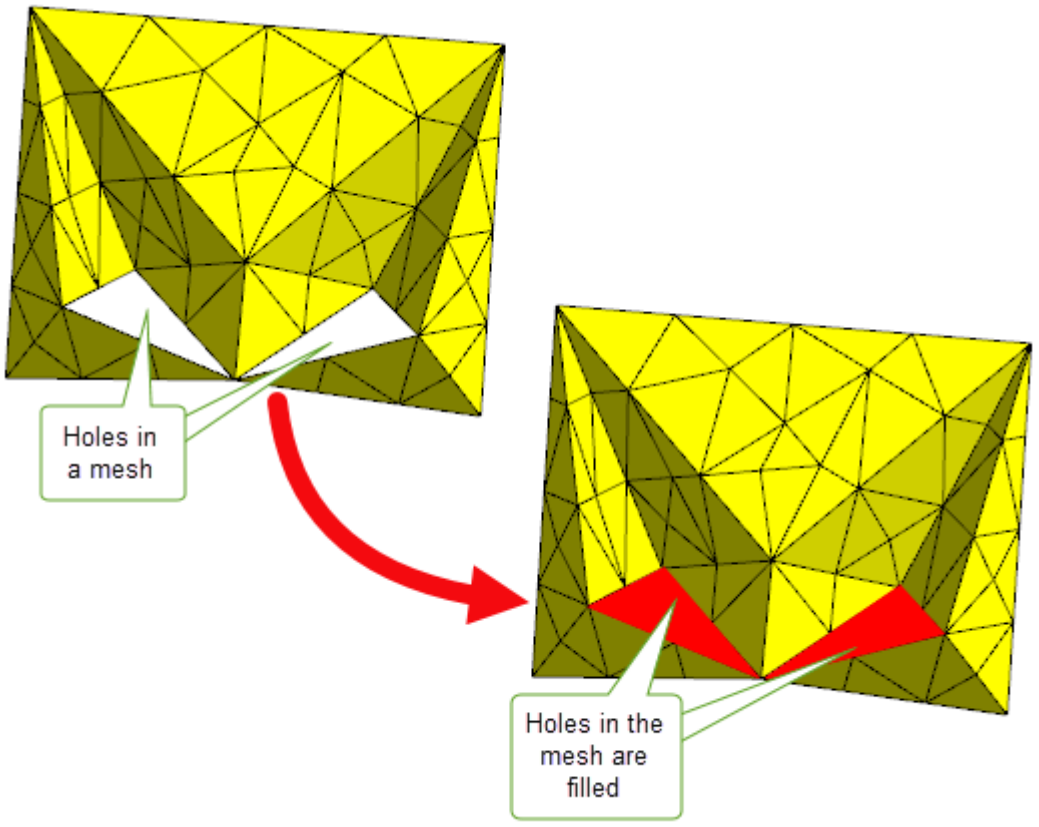


Figure 8–8 *Repairing holes in the mesh*

8.4 Facet geometry support throughout Parasolid

Support for facet geometry is deeply integrated into the Parasolid toolkit, and as a result, most areas of Parasolid functionality fully support bodies that have facet geometry attached.

For a full list of functions along with their level of support for facet geometry, see the *PK Interface Programming Reference Manual*.

Note: There are additional royalty fees for distributing your Parasolid-based products with Convergent Modeling functionality enabled. If you do not license Convergent Modeling, your maintenance fee will cover the use of Convergent Modeling functionality in your development environment for evaluation and prototyping purposes. Your Parasolid Sales Representative can provide full information if you have any questions.

Building Bodies from Profiles

9

9.1 Introduction

Parasolid provides many elementary functions for creating bodies. Some tools, such as those mentioned in Chapter 3, “Model Structure”, can create primitive bodies from scratch. You can use other tools to create more sophisticated bodies using only a minimum of starting information.

This chapter describes the functionality available for creating bodies based on wire or sheet profiles, such as those that can be created using the techniques described in Chapter 6, “Working with Sheets and Wires”. It also contains information on creating swept bodies from solid profiles.

- Section 9.2, “Extruding”, demonstrates bodies created by extruding a profile in a linear direction.
- Section 9.3, “Spinning”, demonstrates bodies created by spinning a profile around an axis.
- Section 9.4, “Sweeping”, demonstrates bodies created by sweeping a profile along a path.
- Section 9.5, “Lofting”, demonstrates bodies created by generating surfaces between a series of profiles.
- Section 9.6, “Embossing”, demonstrates modifying a body by creating emboss features using a specified profile.

Bodies created from profiles may be either sheets or solids, depending on the nature of the operation and the profiles used. Generally, a sheet profile creates a solid body and a wire profile creates a sheet body.

9.2 Extruding

Parasolid can extrude a profile in a linear direction to create a new body. You can define how far the profile is extruded by specifying exactly where the extrusion should start and end. This can be either:

- A specified distance from the profile
- A surface or face that intersects the extrusion, trimming it appropriately
- A solid or sheet body that intersects the extrusion, trimming it appropriately

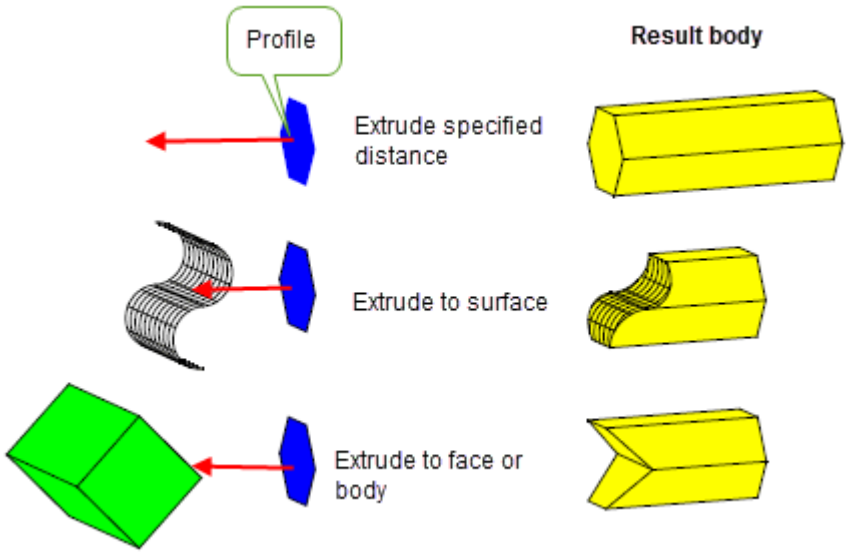


Figure 9–1 *Extruding a profile to create a new body*

9.3 Spinning

You can create a body by spinning a profile about an axis, as shown in *Figure 9–2*. Spinning functionality lets you:

- Form a wire body from a minimum body
- Form a sheet body from a wire profile
- Form a solid body from a sheet profile

You can also spin profiles on an existing body to create new features.

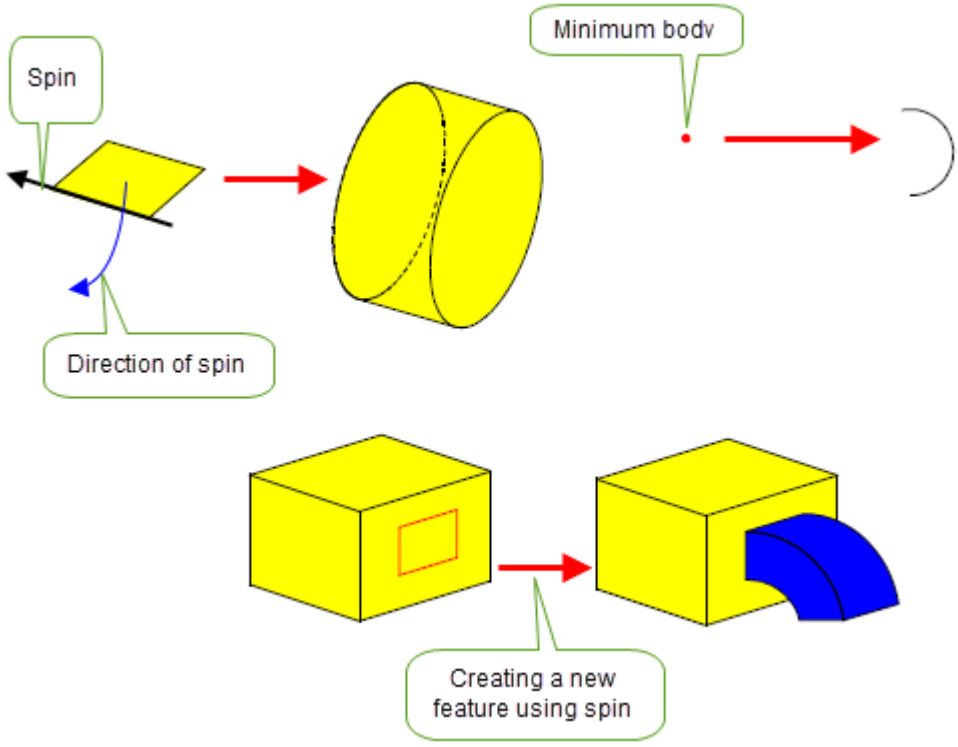


Figure 9–2 *Using spin to create new bodies and features*

9.4 Sweeping

Parasolid's extrusion and spinning functionality is complemented by sweeping, in which one or more profiles are swept along a path using possibly one or more guide wires to produce a new body. The path and guides are specified as wire bodies; unlike extrusion, the path does not need to be linear. *Figure 9–3*, and *Figure 9–4* show how bodies can be created by sweeping.

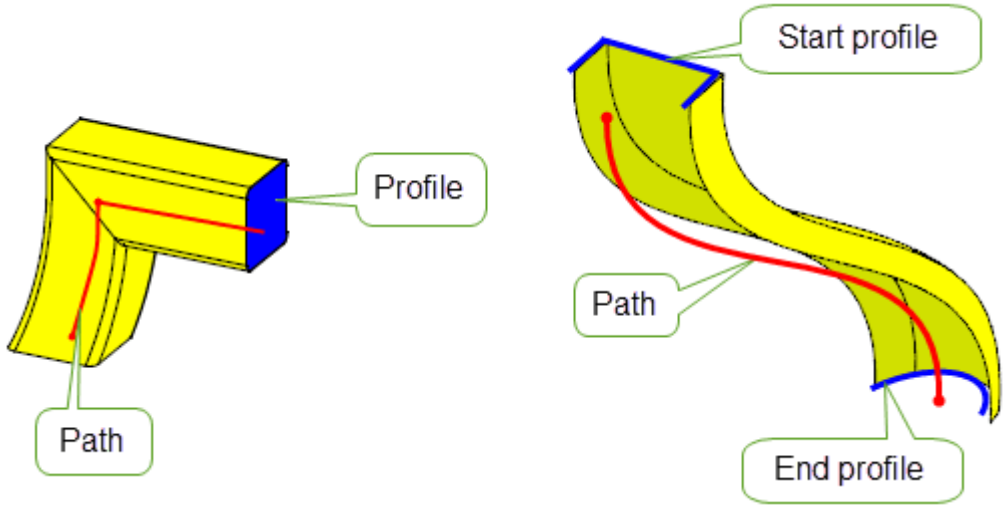


Figure 9–3 *Sweeping profiles along a path to produce a body*

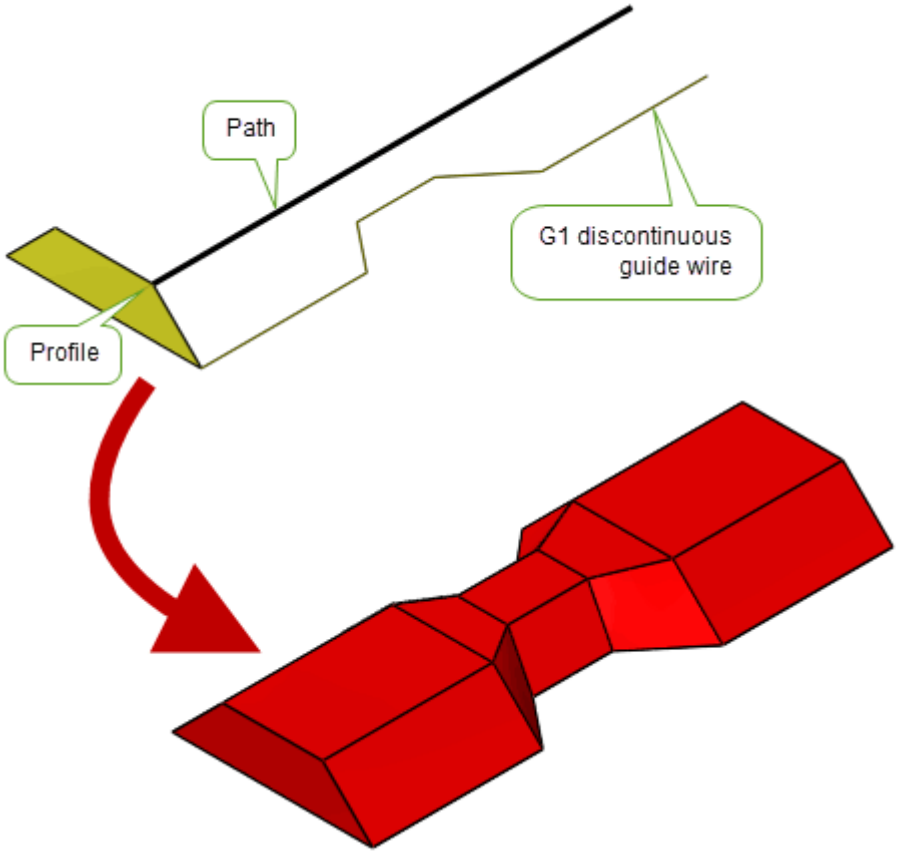


Figure 9–4 *Sweeping a profile along a path with a G1 discontinuous guide wire*

Parasolid lets you control the orientation, size and rotation of profiles in sweep operations, leading to a variety of different effects, as shown in *Figure 9–5*.

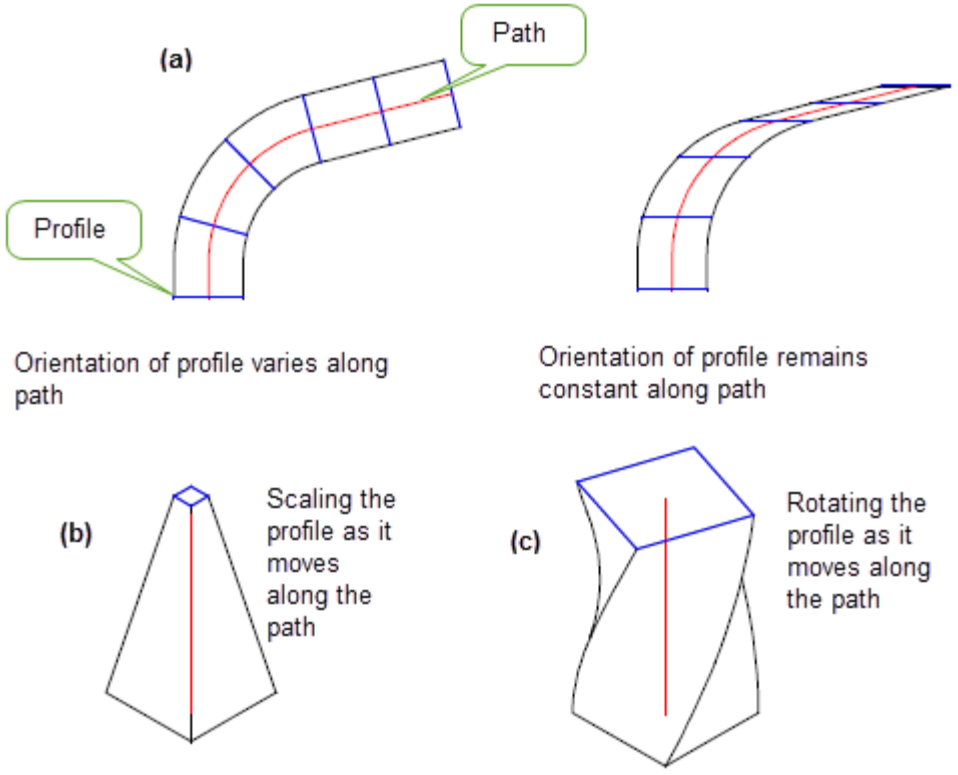


Figure 9–5 *Varying* ***(a)*** *the orientation,* ***(b)*** *the size and* ***(c)*** *the rotation of the profile*

You can also combine these effects, giving you a vast range of possibilities. Parasolid provides particularly strong support for creating helices by sweeping profiles, giving you control over the amount the pitch and taper varies along the extent of the sweep, as shown in *Figure 9–6*.

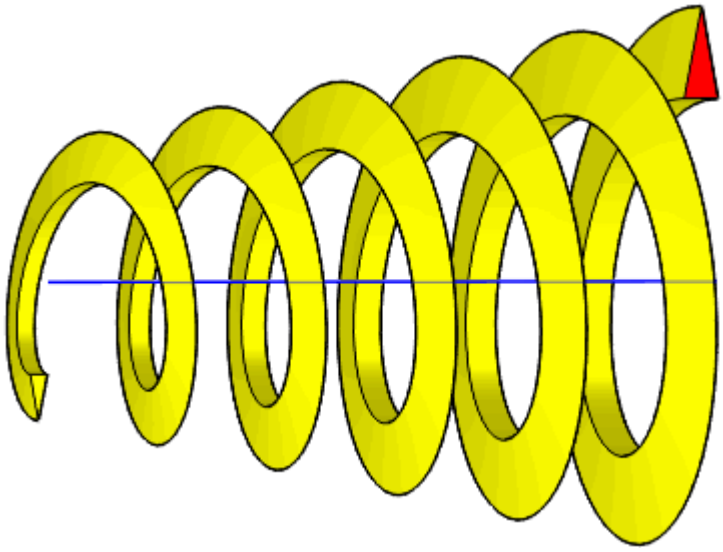


Figure 9–6 *Creating a tapered helix using sweep rotation and size controls*

Parasolid's sweeping functionality also includes many controls to let you, for example:

- Specify the topology of the swept body
- Minimise the amount of twist in a swept body
- Repair certain types of self-intersection caused by the sweep operation
- Simplify the surfaces of the swept body
- Clamp the swept surface at certain profiles during a multi-profile sweep. For example clamping the start and end profiles to faces ensure the swept body meets these faces smoothly.

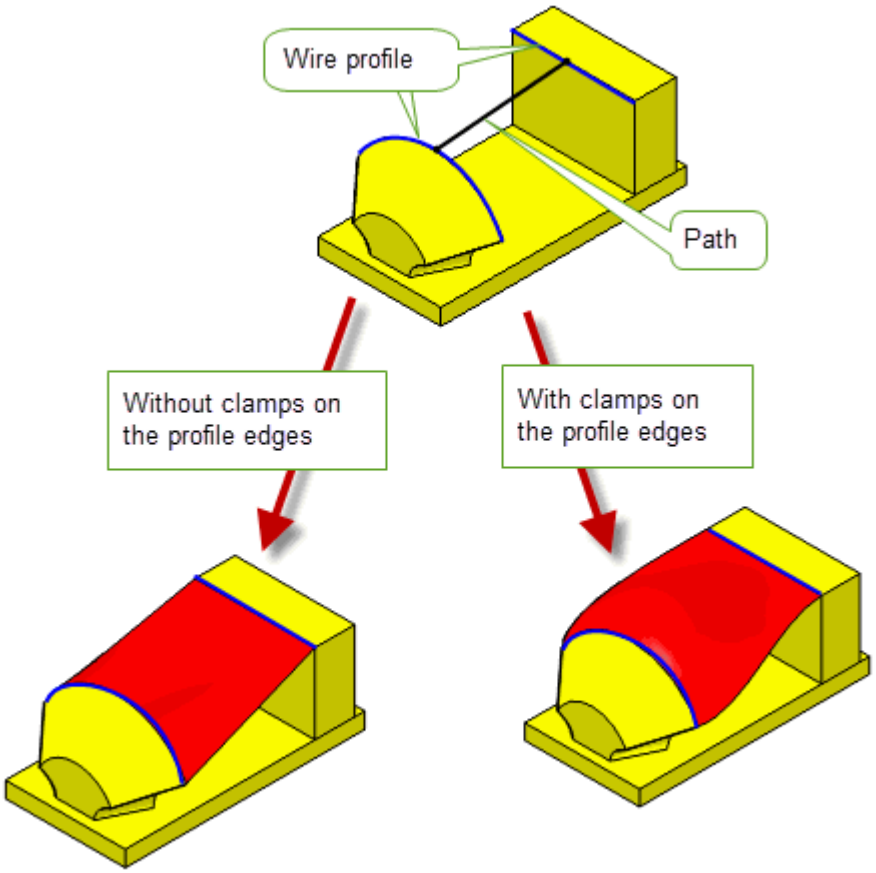


Figure 9–7 *Sweeping profiles with face clamps*

- Lock the orientation of the sweep, either to a set of faces, or in a specified direction

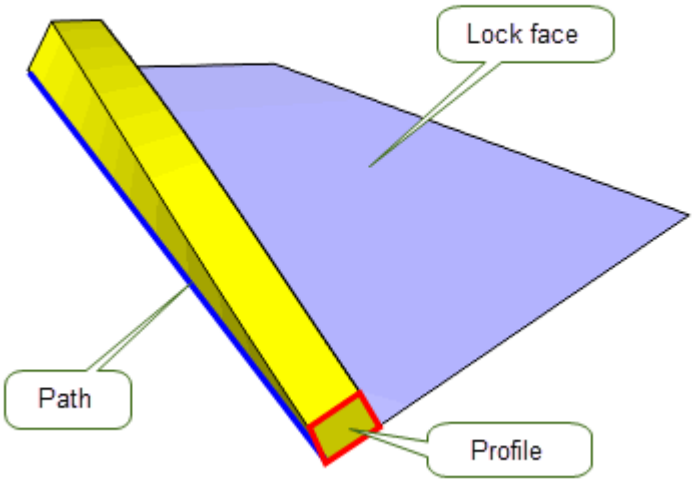


Figure 9–8 *Locking the orientation of the sweep to a set of faces*

- Produce mitred or rounded corners during a sweep operation, as shown in *Figure 9–9*

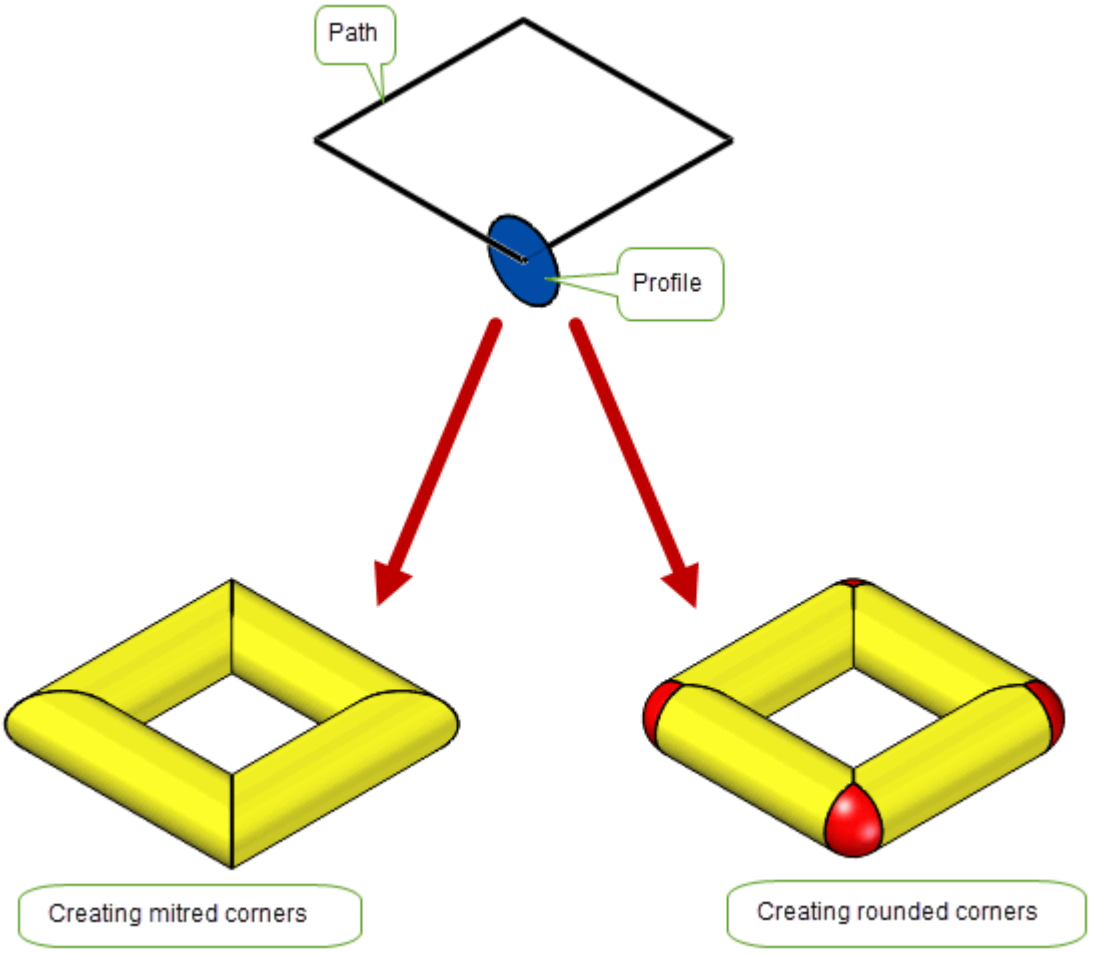


Figure 9–9 *Specifying the type of corner to produce in a sweep operation*

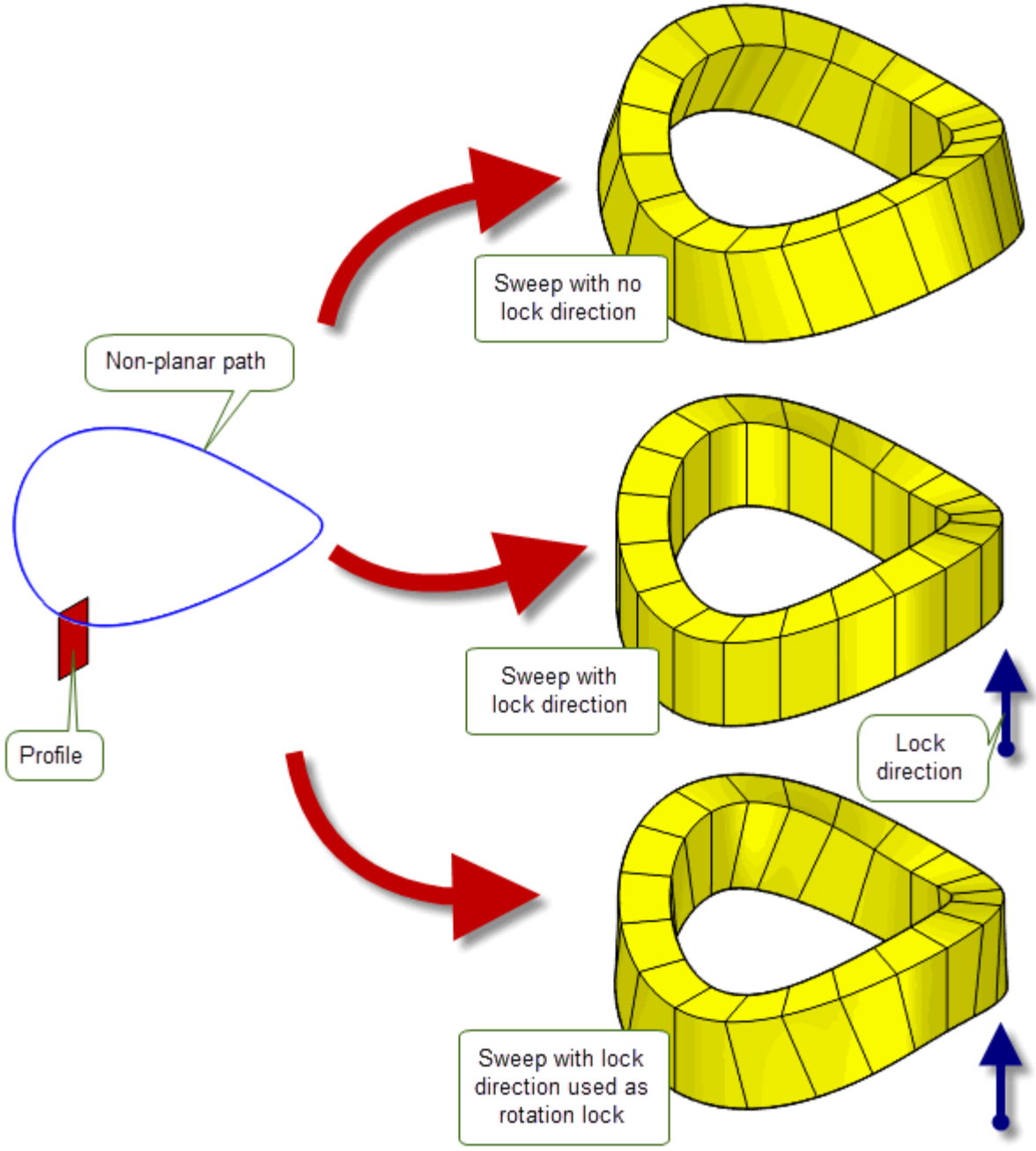

Sweep with no lock direction
Non-planar path
Sweep with lock direction
Lock direction
Profile
Sweep with lock direction used as rotation lock

Figure 9–10 *The effect of locking the direction of a sweep*

- Create a series of cross-section profiles, rather than a swept body, for example for use as a preview tool.

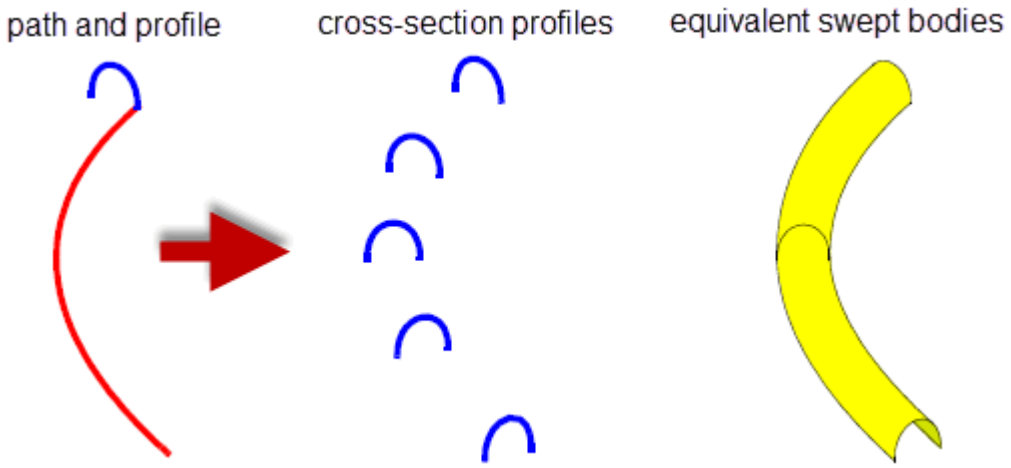


Figure 9–11 *Creating a series of cross-section profiles*

- Trim a swept body so that only the faces between the guide wires are returned

9.4.1 Sweeping solid bodies along a path

In addition to sheet and wire profiles, Parasolid allows you to sweep a solid body along a path, as shown in *Figure 9–12*. This function can be used to construct bodies that represent regions to be cut from another solid body using a machine tool.

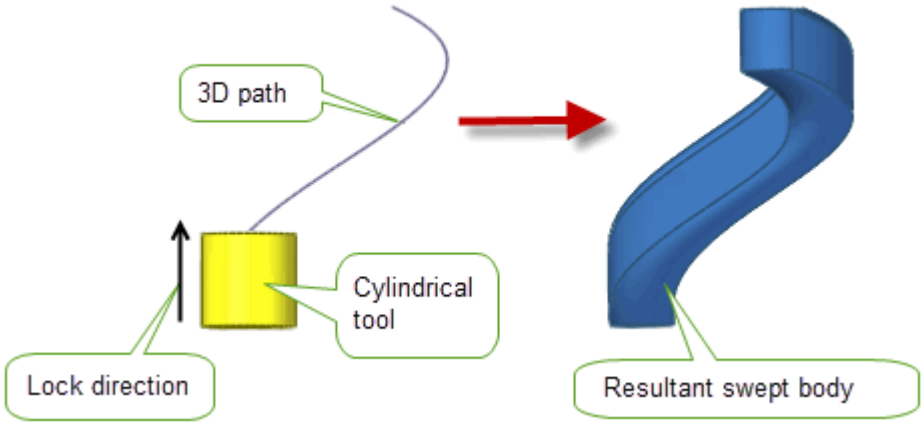


Figure 9–12 *Sweeping a solid tool profile along a path*

You can also use it to specify any faces that will not be involved in a subsequent cutting operation, to generate a simpler set of swept surfaces for the resulting body as shown in *Figure 9–13*.

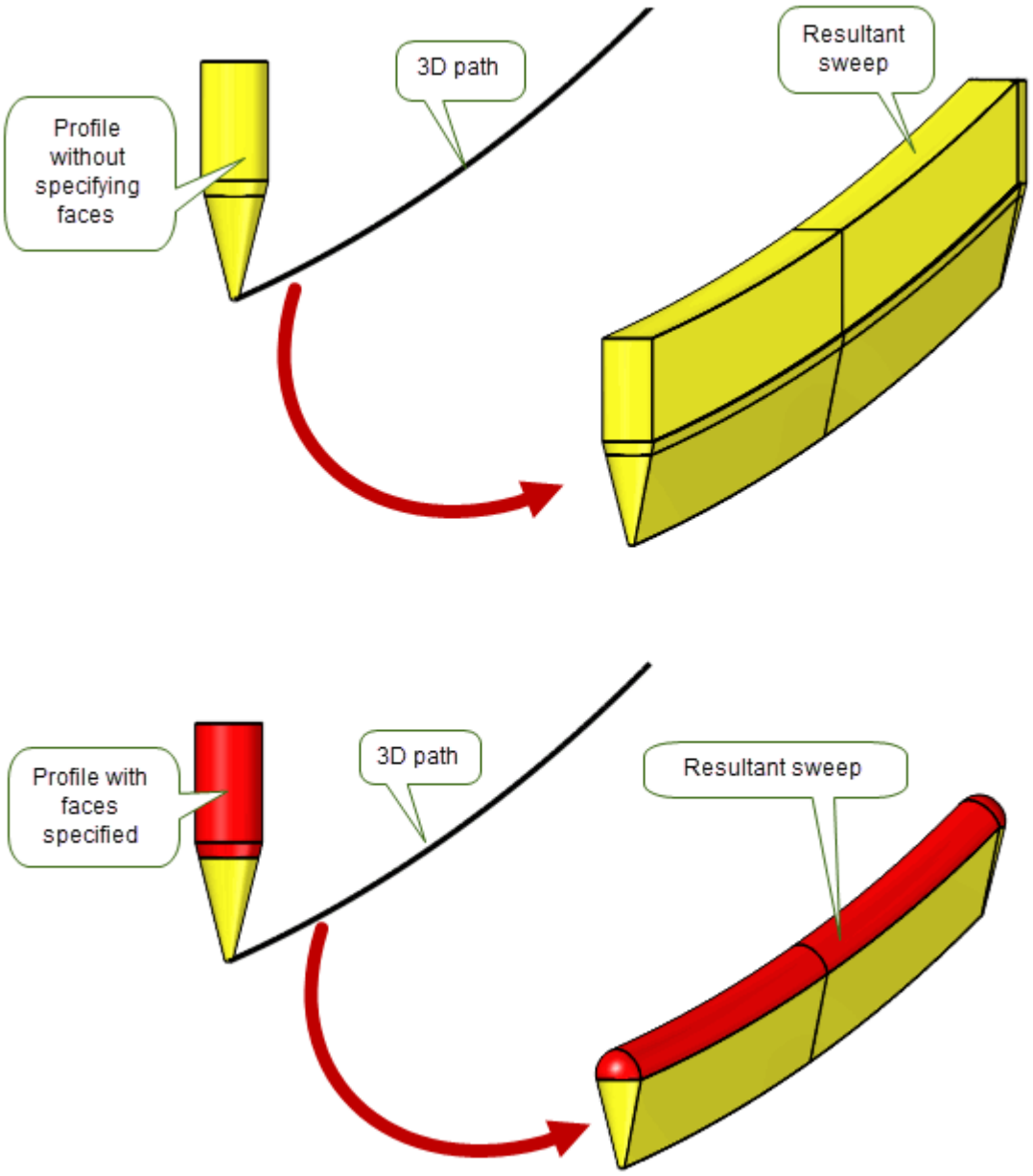


Figure 9–13 *Specifying non-cutting faces on a tool body*

The supplied tool does not have to touch the path: you can choose whether to place the tool on the path prior to the sweeping, or whether to sweep the tool relative to a distant path. This can lead to very different results, as shown in *Figure 9–14* and *Figure 9–15*.

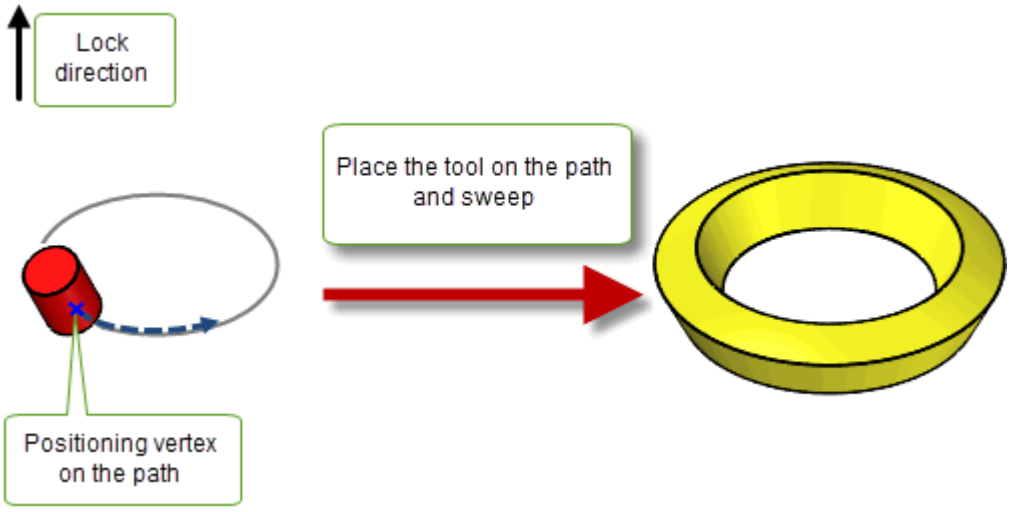


Figure 9–14 *Sweeping a tool that lies on the path*

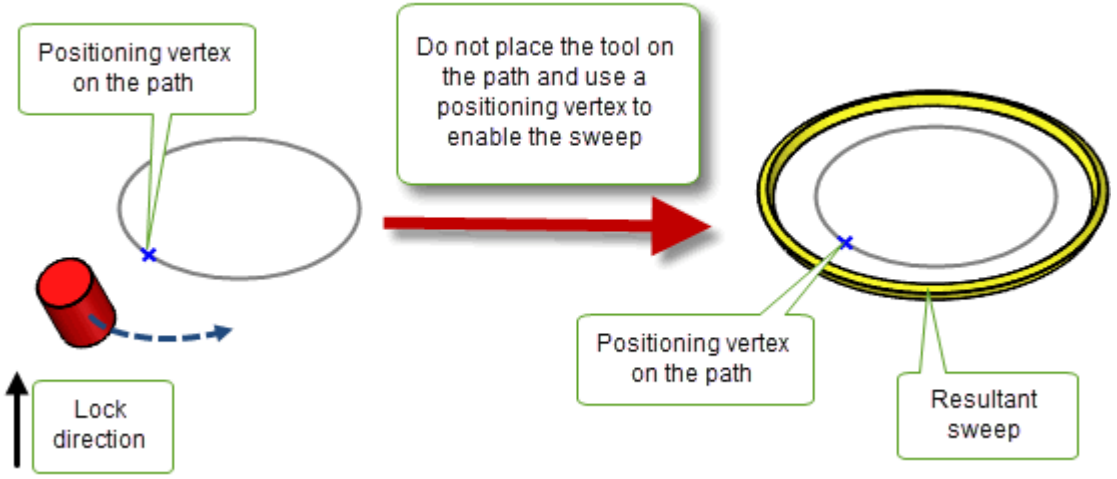


Figure 9–15 *Sweeping a tool that lies off the path*

You can specify the edges of the path that are spun around a supplied axis as part of the sweeping operation as shown in *Figure 9–16*. This is useful in lathing operations where it can help improve the quality of the sweep.

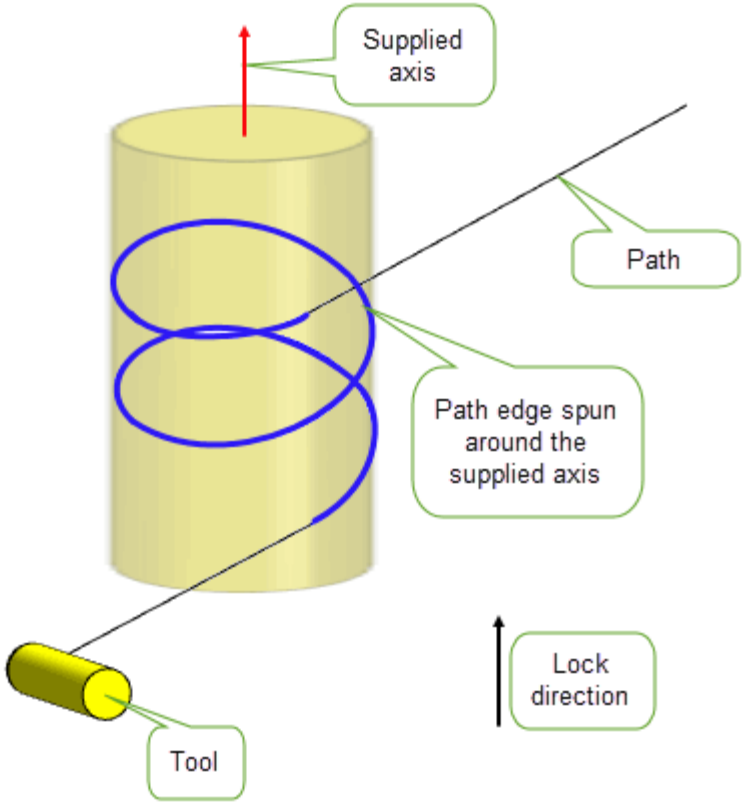


Figure 9–16 *Specifying path edges in the sweeping operation*

You can also choose to perform both the sweep and the boolean operation in one function call. This can reduce the complexity of modelling operations such as simulating lathing operations, where the volume of interest is often the volume remaining after a subtraction operation. *Figure 9–17* shows two examples where you can either subtract or intersect a specified volume to create a result body.

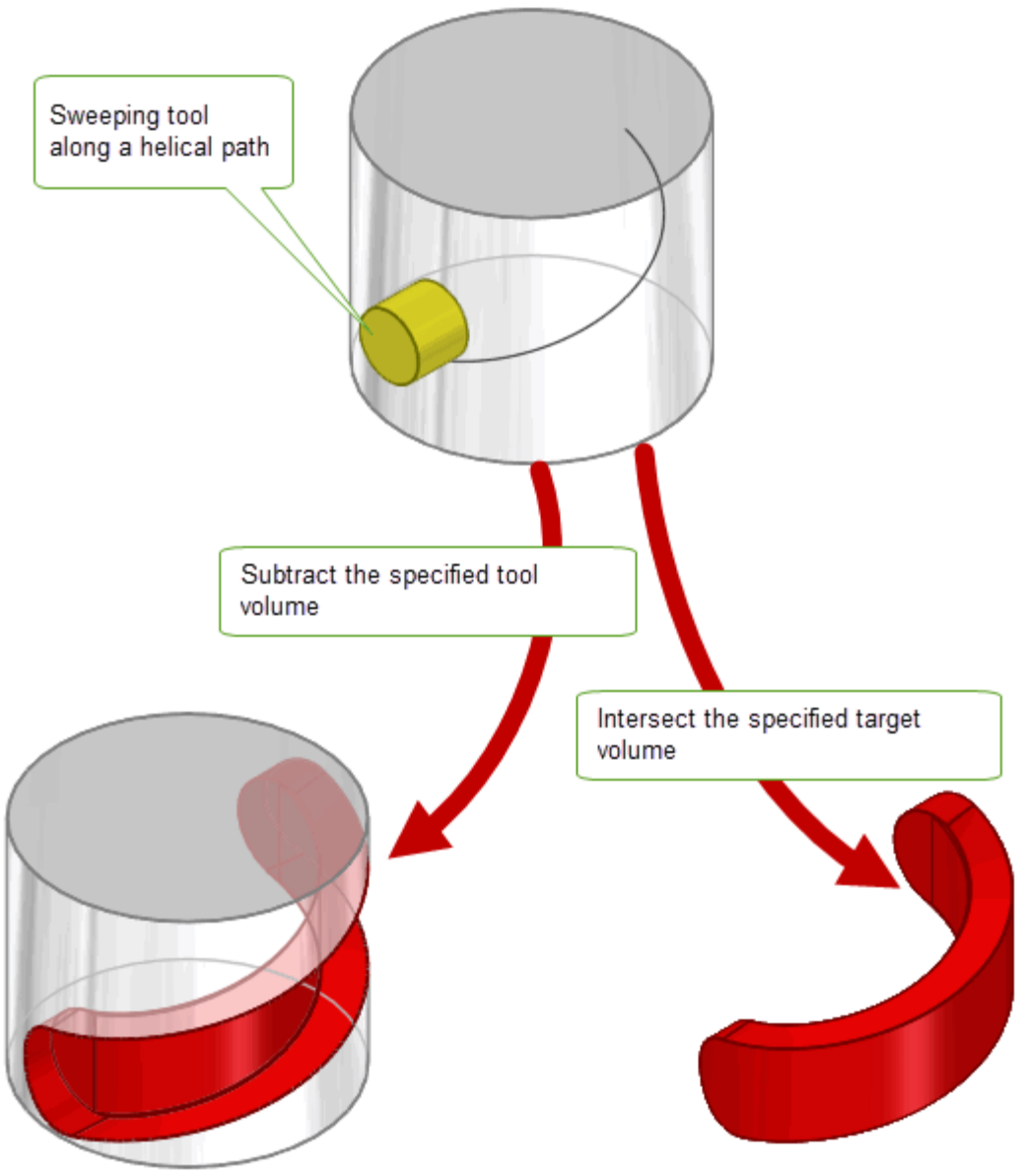


Figure 9–17 *Performing boolean operations with a swept tool body*

9.5 Lofting

Parasolid supports the process of lofting, in which you create a sheet or solid body by fitting surfaces through a series of profiles. Unlike extrusion , lofting involves the use of any number of profiles.

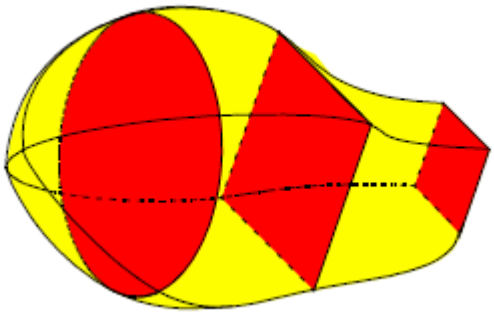


Figure 9–18 *Creating a lofted body using several different profiles*

There are many ways that you can control the final appearance of a lofted body. For example, you can:

- Specify which vertices should match between adjacent profiles. This is shown in *Figure 9–19*, which illustrates two different bodies that can be formed from the same profiles by matching vertices on each adjacent profile in different ways.

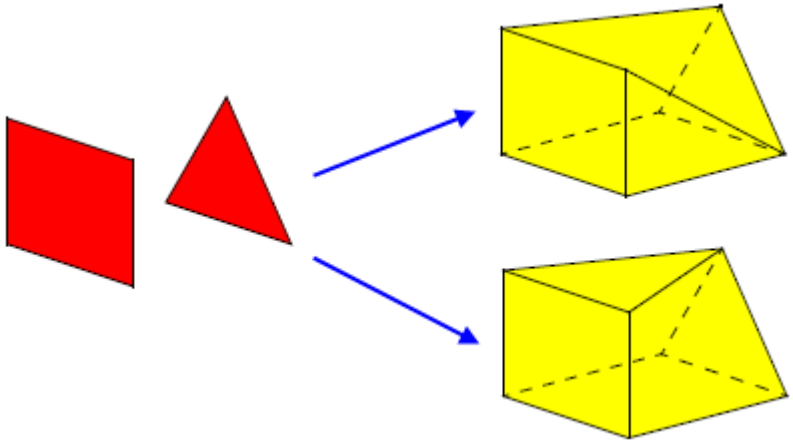


Figure 9–19 *Matching vertices across adjacent profiles to create different bodies*

- Include degenerate profiles at either end of the lofted body, as shown in *Figure 9–20*.

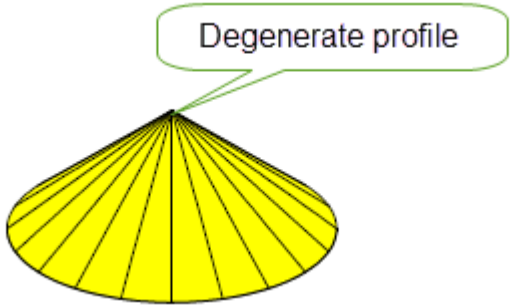


Figure 9–20 *Using degenerate profiles in a loft*

- Control the shape of the lofted surface at each profile, as shown in *Figure 9–21*.

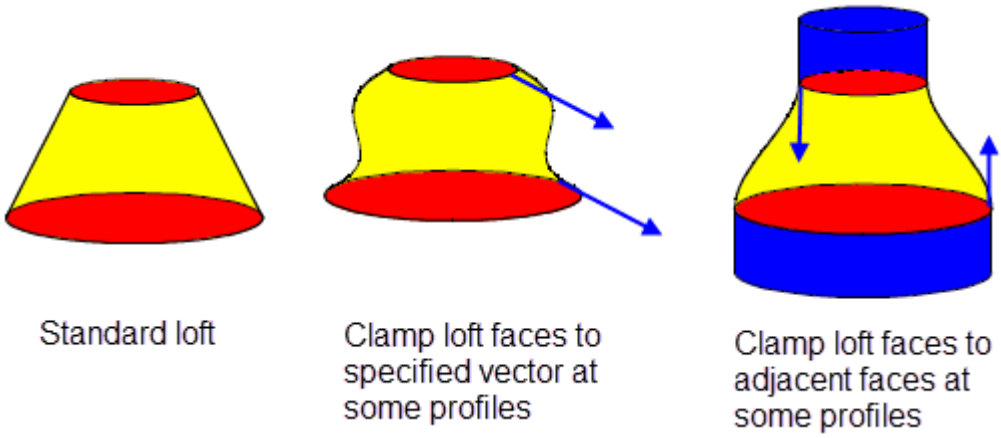


Figure 9–21 *Controlling the shape of a lofted body*

- Constrain the shape of the lofted body by specifying additional guide curves, as shown in *Figure 9–22*.

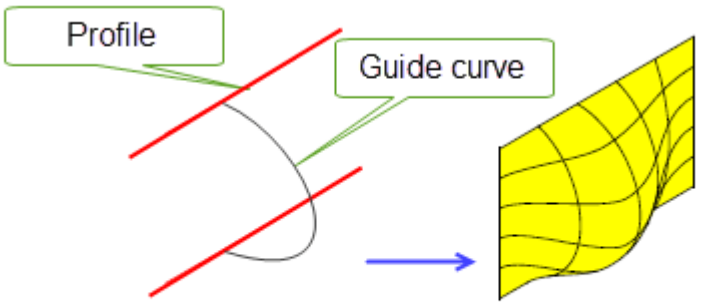


Figure 9–22 *Using a guide curve to control loft shape*

- Create a piecewise lofted body that is non-G1 smooth across the supplied repeated profiles.

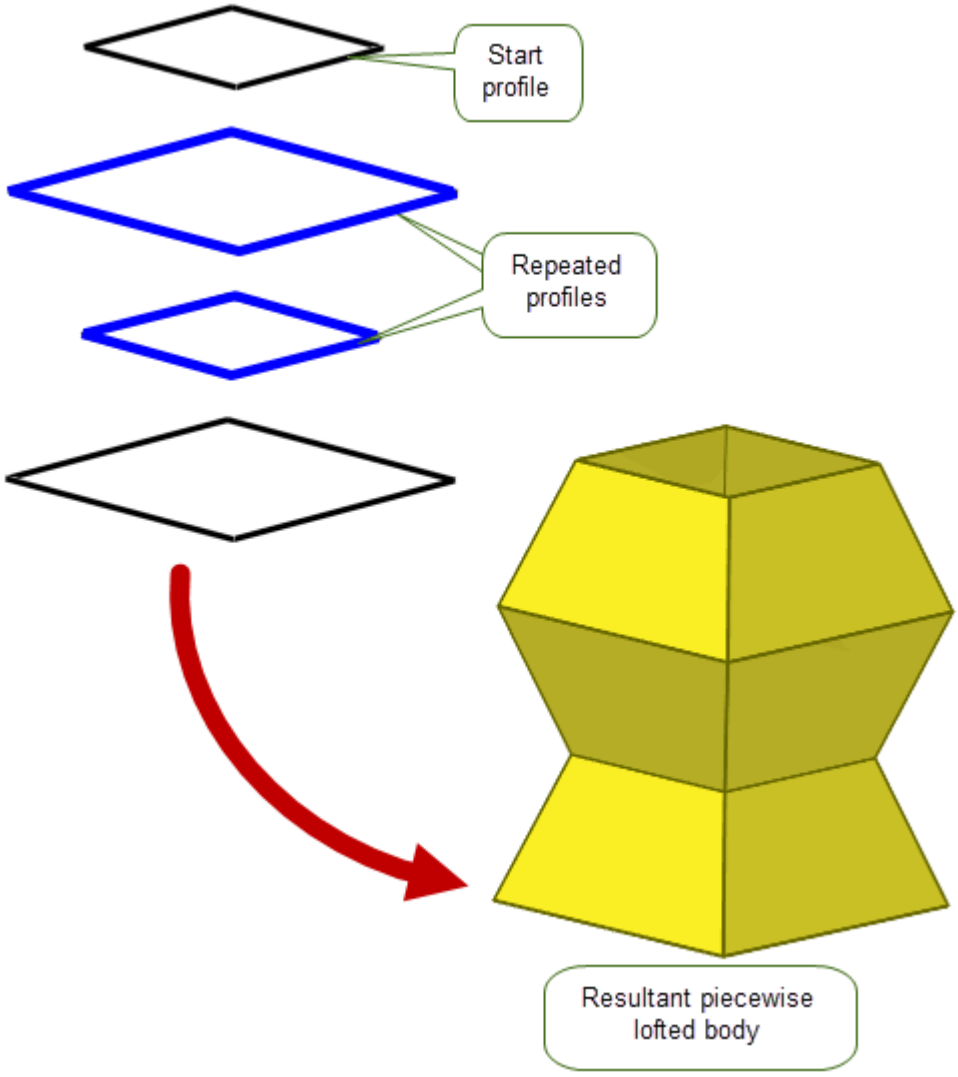


Figure 9–23 *Piecewise lofting with repeated profiles*

Other controls let you:

- Specify the topology of the lofted body
- Minimise the tolerances used in the lofted body
- Simplify the surfaces of the loft body

As with sweeping, you can combine these options, making Parasolid lofting a powerful tool for creating free-form bodies using only the simplest of starting points.

9.6 Embossing

Parasolid provides functionality for adding protruding (pad) or indented (pocket) emboss features to a target body. These operations can be applied globally, or locally to a selection of faces, and the features created can either be attached to the target body or returned as a separate body. The shape of emboss features is described using supplied profiles.

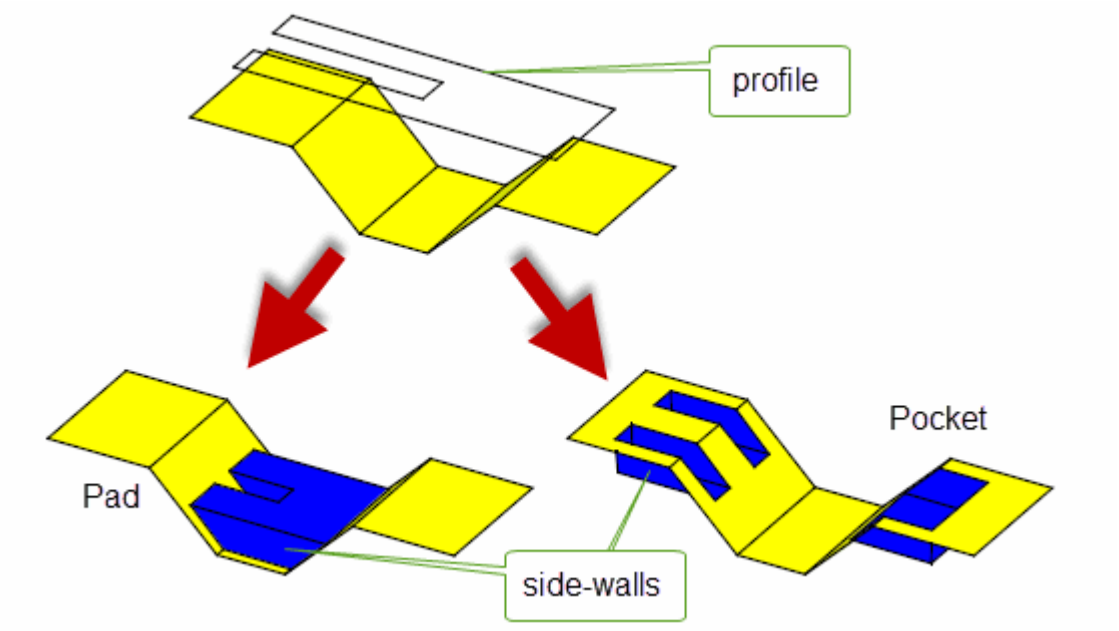


Figure 9–24 *Creating pad and pocket emboss features on a target body*

A fundamental part of Parasolid's emboss functionality is creating sidewalls, as shown in *Figure 9–24*, which is done automatically for emboss features. Parasolid can construct a sidewall using one of following methods:

- Tapered sidewalls can be created based on a specified draw direction and taper angle.
- Ruled sidewalls can be created using the normals of faces in the supplied profile.
- Swept sidewalls can be created based on a specified sweep direction.

In each case, the geometry of the sidewall is formed by a set of ruled surfaces that pass through the boundary of the profile. An example of a pad emboss feature with sidewalls tapered at different angles is shown in *Figure 9–25*.

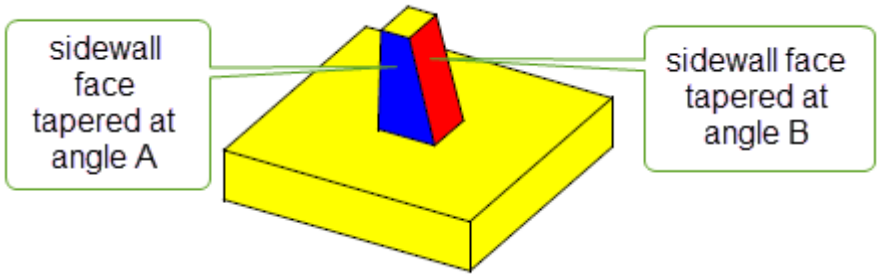


Figure 9–25 *Pad emboss feature with tapered sidewalls using different angles*

As well as these methods of internal construction, you can supply a sidewall yourself.

Blending 10

10.1 Introduction

In Parasolid, **blending** is used as a broad term that encompasses the following operations:

- Smoothing off sharp edges in a body – often known as filleting and chamfering.

  Filleting: replace edges with new blend faces that meet the adjoining faces of the original edges smoothly. You can fillet a sharp edge on a piece of wood by smoothing it off using sandpaper.

  Chamfering: replace edges with new blend faces that have a linear cross-section, but do not meet the adjoining faces of the original edges smoothly. You can chamfer a sharp edge on a piece of wood by flattening it off using a plane.

- Constructing blend faces between non-adjacent faces, either from a single body or from different bodies.

  Here, new surfaces are created between faces that are not necessarily connected, possibly joining bodies together in the process.

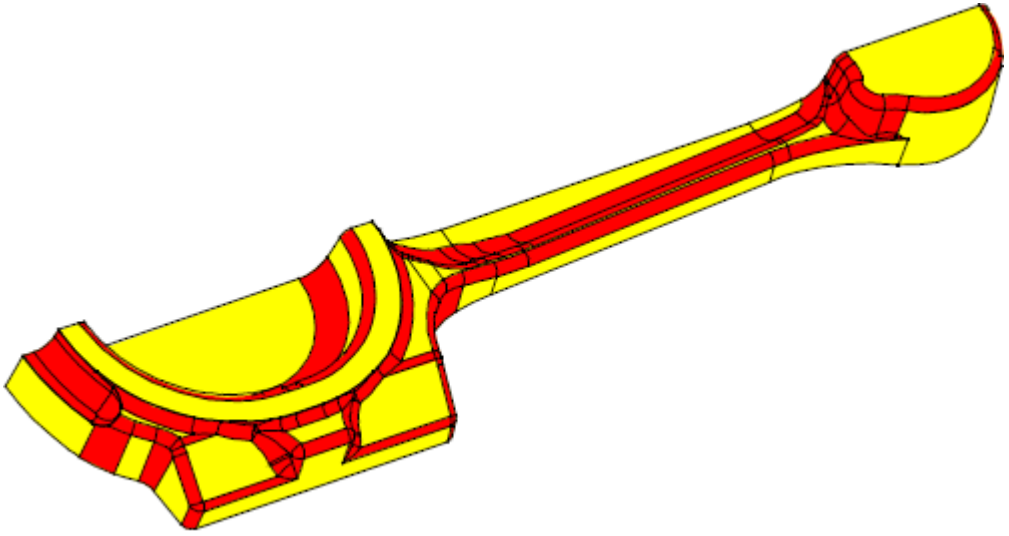


Figure 10–1 *Blending in Parasolid*

Parasolid provides three specific kinds of blending:

Blend	Description
Edge blending	You supply a set of edges and Parasolid replaces them with new blend faces that meet the adjoining faces of the original edges. See Section 10.2.
Face-face blending	You supply two sets of faces (which need not be in the same body) and Parasolid creates new blend faces that lie between them. See Section 10.3.
Three-face blending	A specialized variation of face-face blending: you supply three sets of faces and Parasolid blends them, destroying the middle set in the process. See Section 10.4.

Rather than being alternatives, these different types of blending have complementary strengths so that, when combined, they can work powerfully together.

Strengths of edge blending	Strengths of face blending
■ Can create complicated configurations of blends ■ Simpler to use ■ Can create webs of blends rather than a single chain	■ Good for creating long sweeps of blends ■ One step process ■ Provides more user interaction ■ Provides a huge variety of controls for defining the result ■ Can blend between disjoint components

10.2 Edge blending

During edge blending, Parasolid generates a series of faces to replace a set of specified edges in a body. These new faces meet the faces that were separated by the original edges, usually smoothly.

Edge blends are initially attached as attributes to the edges they will replace. Such blends are referred to as **unfixed** and can be modified or deleted.

Once defined, unfixed blends need to be fixed, incorporating them into the body and replacing the original edges.

This two-stage process allows Parasolid to resolve all the dependencies between multiple blends, such as that shown in *Figure 10–2*, which illustrates several intersecting blends of different radii. If a single-stage process was used, you would have to decide the order in which to apply multiple blends – a far from straightforward task!

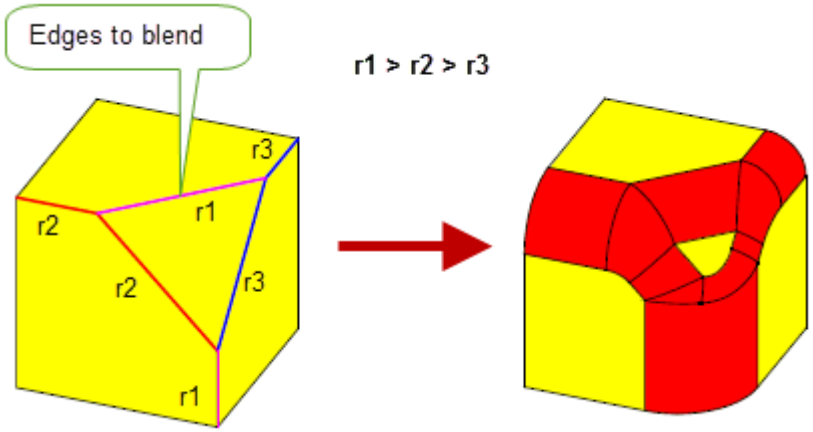
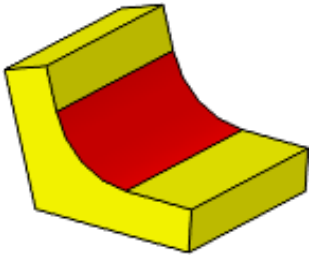
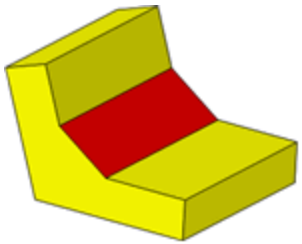
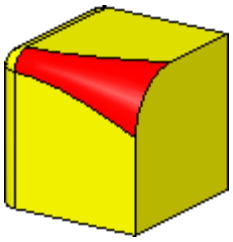


Figure 10–2 *Creating complex intersecting blends using edge blending*

10.2.1 Types of edge blend

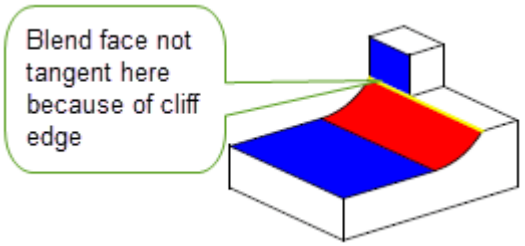
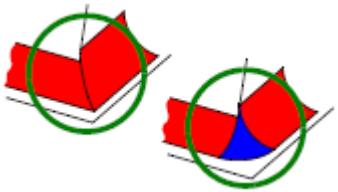
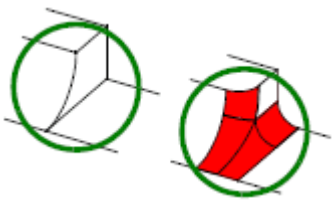
Parasolid can create a variety of edge blends.

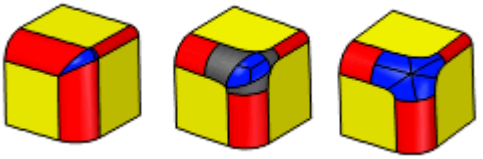
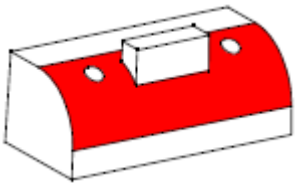
Blend Type	Description	Example
Constant-radius	A blend with a circular cross-section: the result of rolling a ball along the two surfaces adjoining the edge.	

Blend Type	Description	Example
Chamfer	A rolling-ball blend with a linear cross-section. Chamfer blends are not tangent-continuous with the faces adjoining the original edge. There are two types of chamfer blends: ■ face offset chamfer blends ■ apex-range chamfer blends Apex-range chamfer blends provide greater control over the width and angle of the resulting chamfer.	
Variable-radius	A rolling-ball blend in which the radius of the ball varies along the length of the blend. These blends need not even be circular: Parasolid can create blends with elliptical, parabolic or hyperbolic cross-sections.	

10.2.2 Controlling the appearance of edge blends

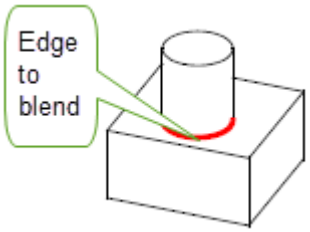
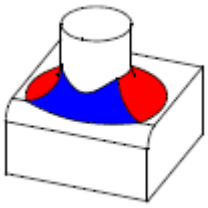
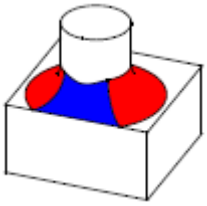
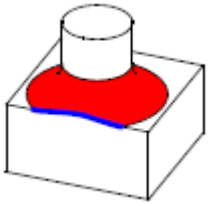
Parasolid provides many options to allow you to control the appearance of edge blends. Some of the most common are illustrated here.

Option	Description	Example
Cliff-edge	A cliff-edge blend is tangent to only one of the faces adjacent to the original blend edge.	
Vertex blending	Vertex blending can be used to roll over adjacent sharp edges at a vertex, further smoothing the blend.	
Y-shaped	Parasolid can create flatter “Y-shaped” blends when two edges of different convexity meet at a vertex with three or more edges.	

Option	Description	Example
Setback	When three or more blends meet at a vertex, a setback blend can be used to create a larger, flatter blend area in the region where the blends meet.. Setback blends can be implemented with or without collar faces.	
Preserving topology	Any topology that is completely overlapped by a blend face can be preserved. Alternatively, such topology can be deleted from the part.	

10.2.3 Edge blend overflows

Many blends cannot fit entirely inside one or both of the faces adjacent to the blended edge. Parasolid has a number of strategies that enable such blends to be fixed successfully. This process is known as creating **blend overflows**. Parasolid can create a number of different types of overflow, depending on the appearance you want and the exact configuration of the body in the region of the blend:

Overflow	Description	Example
No overflow	The body before edge blending	
Smooth	In the overflowing region, the blend surface is changed so that it flows across the smooth edge into the adjacent face. **Note:** In the example, the blend along the front edge of the block is already present.	
Cliff	In the overflowing region, the blend surface is changed so that it blends along the edge using a cliff-edge blend.	
Notch	In the overflowing region, the blend surface is trimmed by extending the neighboring faces. The blend remains the same in all other respects.	

10.2.4 Controlling blend propagation

Parasolid can propagate blends along chains of tangent edges. You create a blend on one edge, and Parasolid will create a blend on the next edge if it meets the previous edge smoothly, continuing the chain of blends until a non-smooth edge is met.

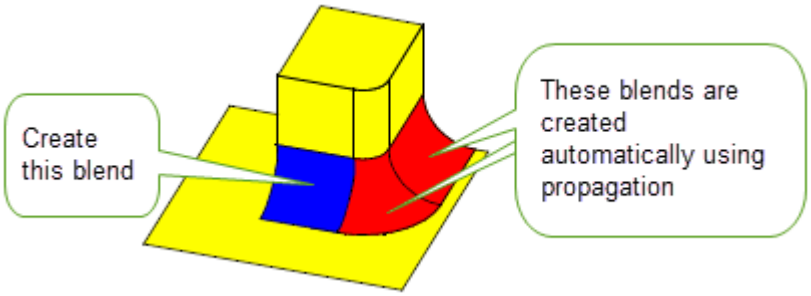


Figure 10–3 *Propagating blends along smooth edges*

10.2.5 Creating blend limits

Parasolid can limit an edge blend to stop it short of its natural finishing point, using either vertex or edge limits, or information about where blends would overlap each other.

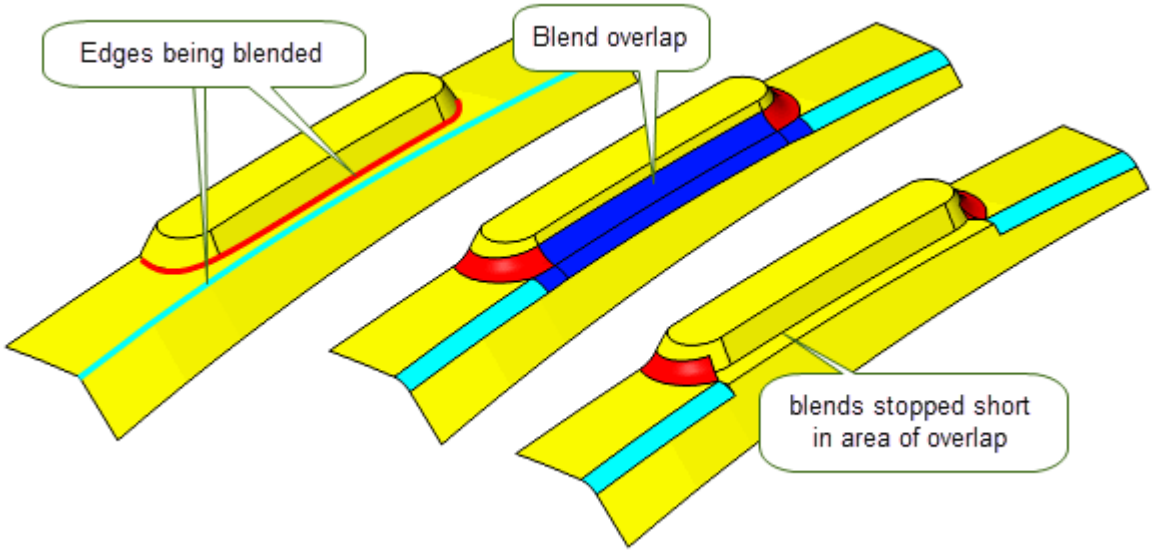


Figure 10–4 *Creating blend limits for overlapping blends*

These methods can be combined, as shown in *Figure 10–5*.

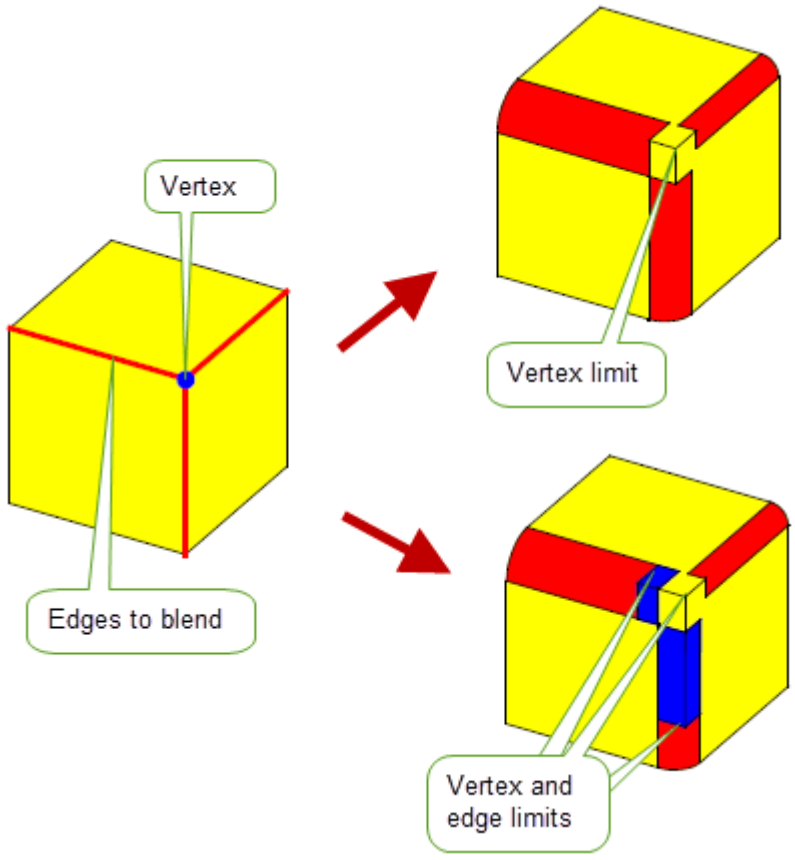


Figure 10–5 *Blends limited at a vertex and along two edges*

In addition, you can stop a blend short by capping it with either faces from the body, a plane, or a sheet body.

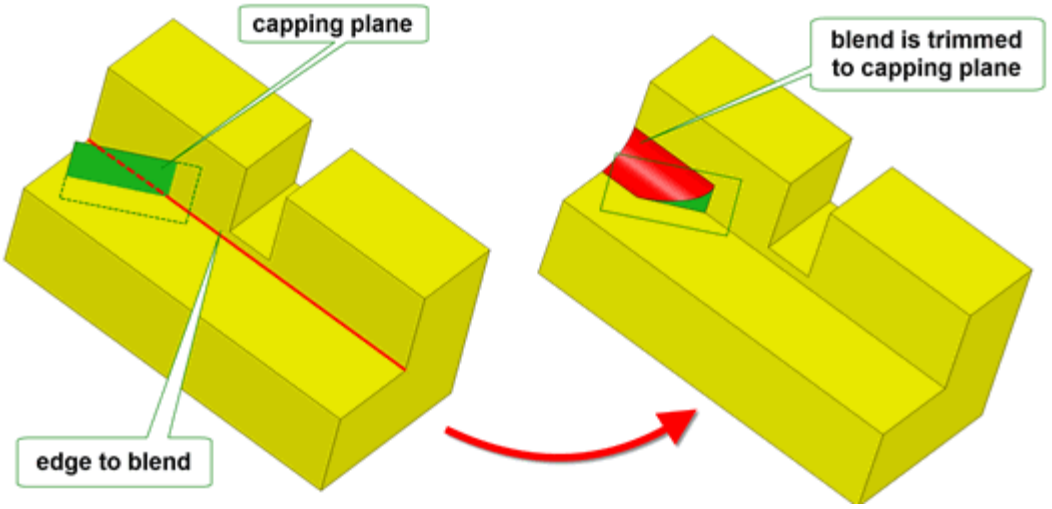


Figure 10–6 *Trimming edge blends to a face in the body*

10.3 Face-face blending

Face-face blending differs from edge blending as follows:

- The sets of faces between which you are blending do not need to be adjacent, or even in the same body.
- Face-face blends are created and fixed in a single operation. There is no such thing as an unfixed face-face blend.

To create a face-face blend, you choose two sets of faces, known as the left and right **walls** of the blend. You then define the blend in terms of the three independent properties shown in *Figure 10–7*.

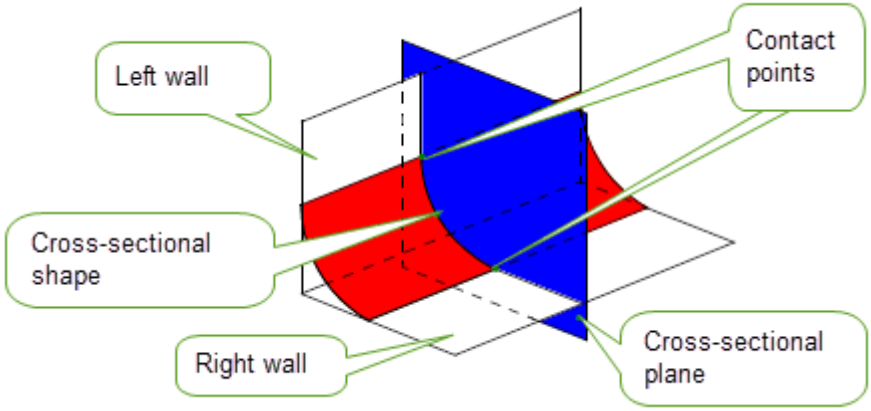


Figure 10–7 *General definition of a face-face blend*

10.3.1 Cross-sectional planes

You can create different face-face blend effects by varying the cross-sectional plane, as shown in *Figure 10–8*.

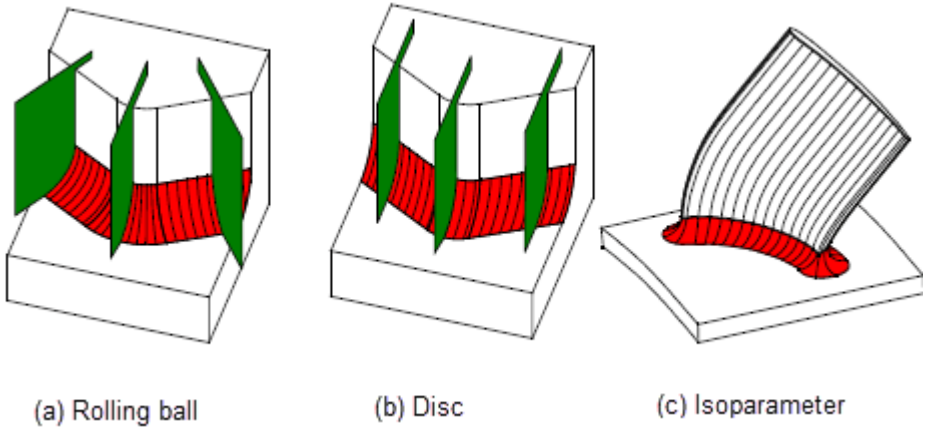


Figure 10–8 *Creating blends with* ***(a)*** *rolling-ball,* ***(b)*** *disc and* ***(c)*** *isoparameter cross-sectional planes*

Rolling-ball and disc blends are the most common form of cross-sectional plane. Isoparameter blends are specialized blends that are useful, for example, in turbine design.

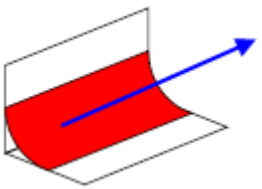
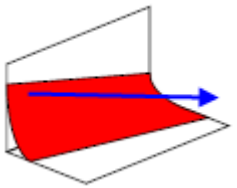
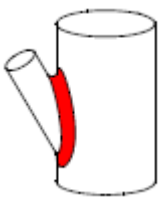
10.3.2 Contact points

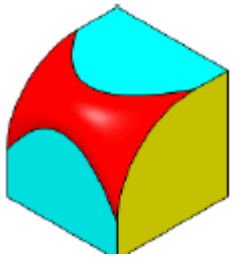
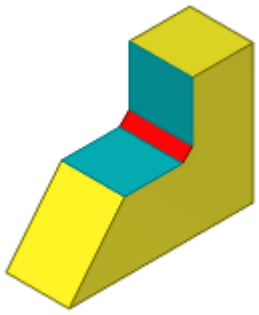
Each cross-sectional plane in a blend contains two contact points, where the blend surface touches one of the blend walls. You can determine the position of these contact points in two ways:

- You specify the size of the blend and let Parasolid calculate the position of the contact points
- You specify the blend boundary that the contact points should lie on, and Parasolid ensures the blend fits inside it.

10.3.2.1 Specifying the size of the blend

The following types of blend are defined in terms of the size of the blend:

Blend type	Description	Example
Constant-size face offset	The blend spine (arrowed line) remains the same distance from each wall along the length of the blend.	
Variable-size face offset	The distance of the blend spine from each wall varies according to ranges you specify.	
Constant-width	The width of the blend remains constant along the length of the blend.	

Blend type	Description	Example
Variable-width	The width of the blend varies according to ranges you specify.	
Apex-range chamfer	The distance of the chamfer apex from each wall and/or the angle of the chamfer to an underlying wall varies according to the data you specify.	

10.3.2.2 Specifying the blend boundary

The following types of blend boundary, or **holdline**, can be defined. Parasolid creates blends that are bounded by these holdlines.

Type of holdline	Description
Tangent	You define the position of one boundary and let Parasolid calculate the position of the other boundary.
Single conic	You define the position of one boundary and use the standard blend size controls to define the position of the other boundary.
Double conic	You define the position of both boundaries and let Parasolid fit the blend within them.

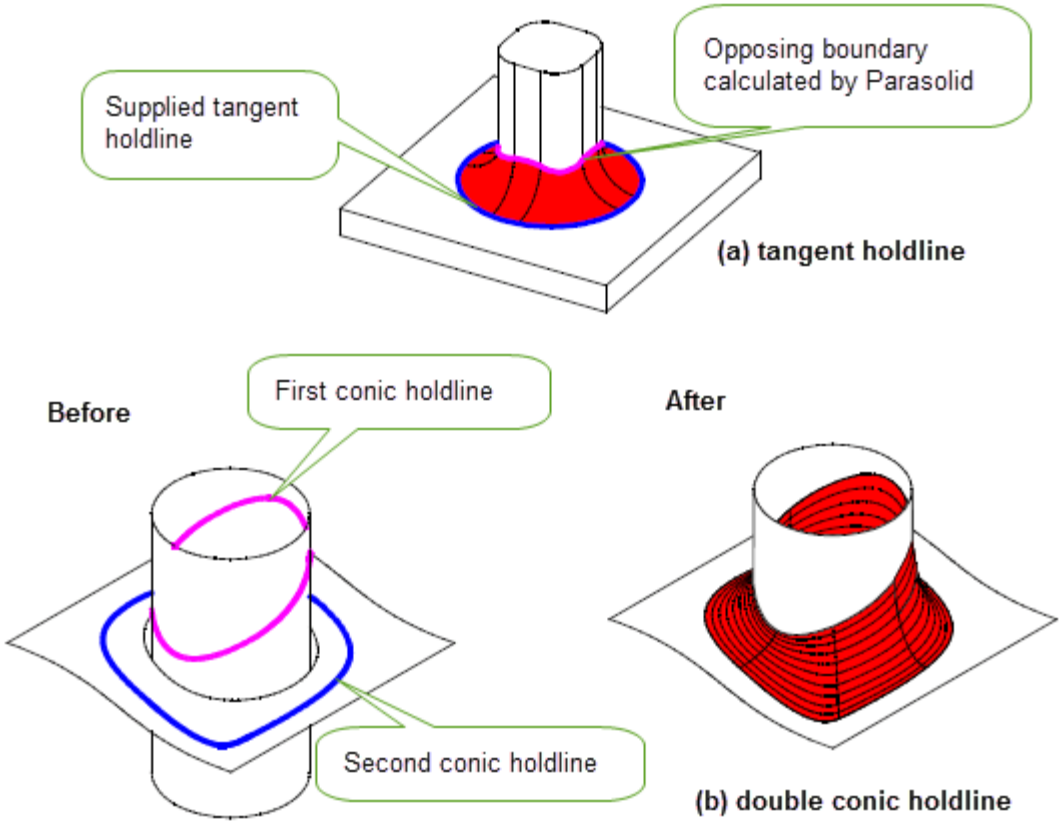


Figure 10–9 *Supplying* ***(a)*** *tangent holdlines and* ***(b)*** *double conic holdlines*

Holdlines can also be inverted, so that the blend is tangent to the face above the holdline rather than the face below, as shown in *Figure 10–10*.

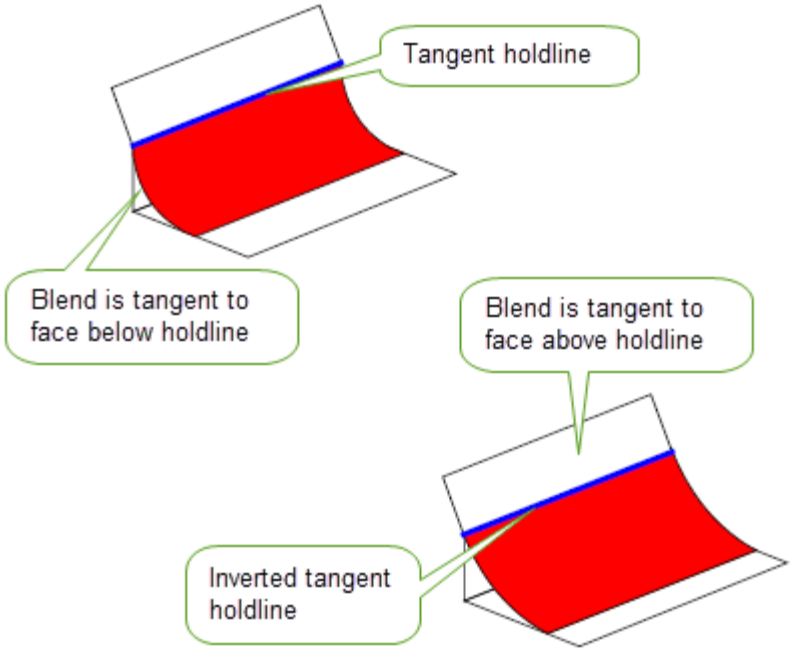


Figure 10–10 *Using inverted holdline blends to control the tangency of a blend*

You can also specify that a boundary should form a cliff-edge blend. As with edge blends, a face-face blend does not have to join the blend wall tangentially along the cliff edge.

10.3.3 Cross-sectional shape

Within a cross-sectional plane, you can control the shape of the blend (that is, the curve that joins the two contact points). Parasolid can create the following types of cross-sectional shape:

Cross-section	Description
Conic	A circular, elliptical, hyperbolic or parabolic cross-section.
Chamfer, or linear	A straight line between the contact points.
Curvature-continuous, G2-continuous	A cross-section with the same curvature as the wall at each contact point.

10.3.4 Trimming

Parasolid provides different ways of trimming a face-face blend to its surrounding faces so as to produce the required final shape. Trimming may involve removing material from the blend surface, the walls, or both. The blend may be attached to the walls or created as a separate sheet body.

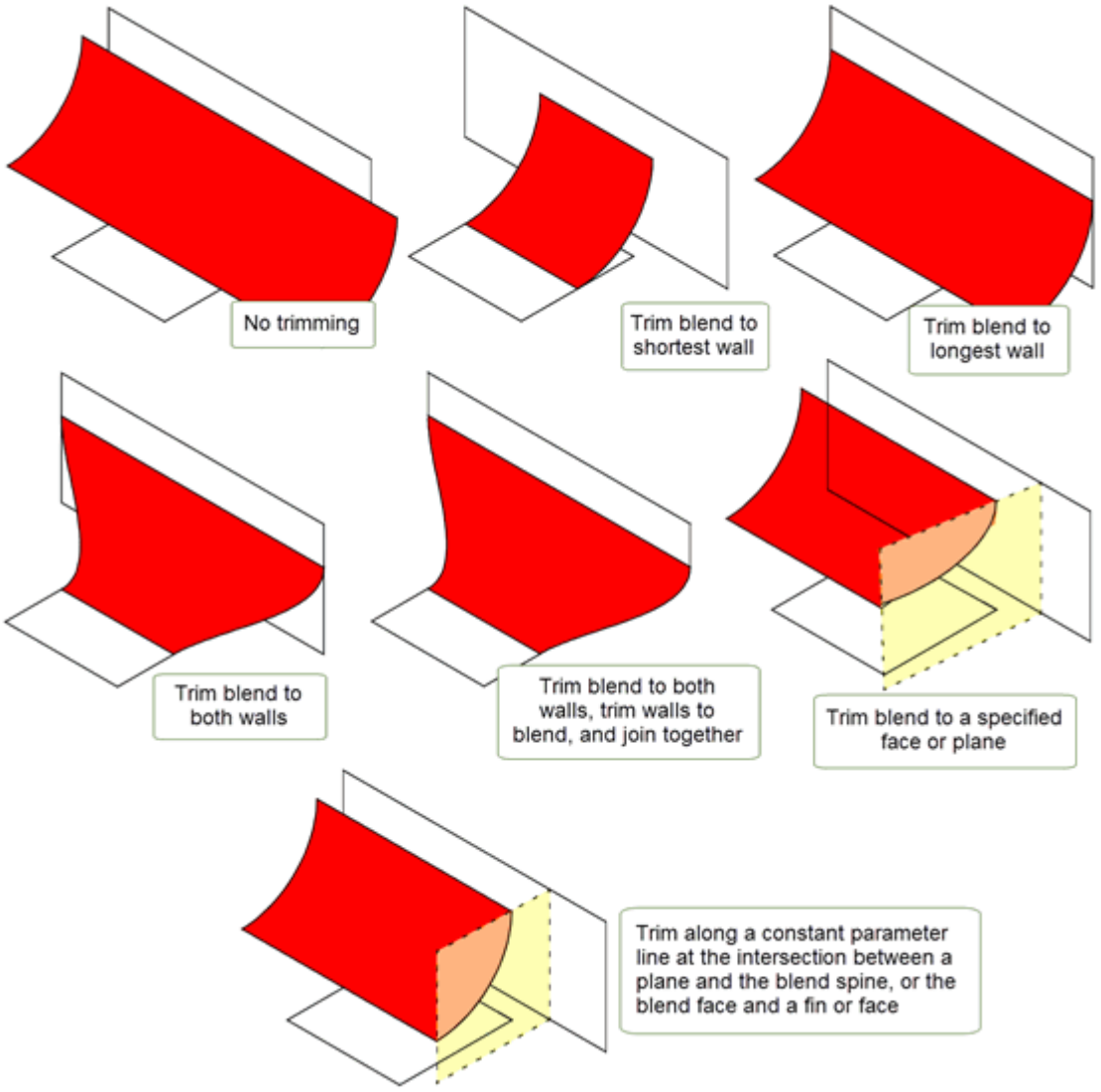


Figure 10–11 *Trimming blend faces and blend walls*

Parasolid can also trim blend walls using its imprint completion functionality as shown in *Figure 10–12*.

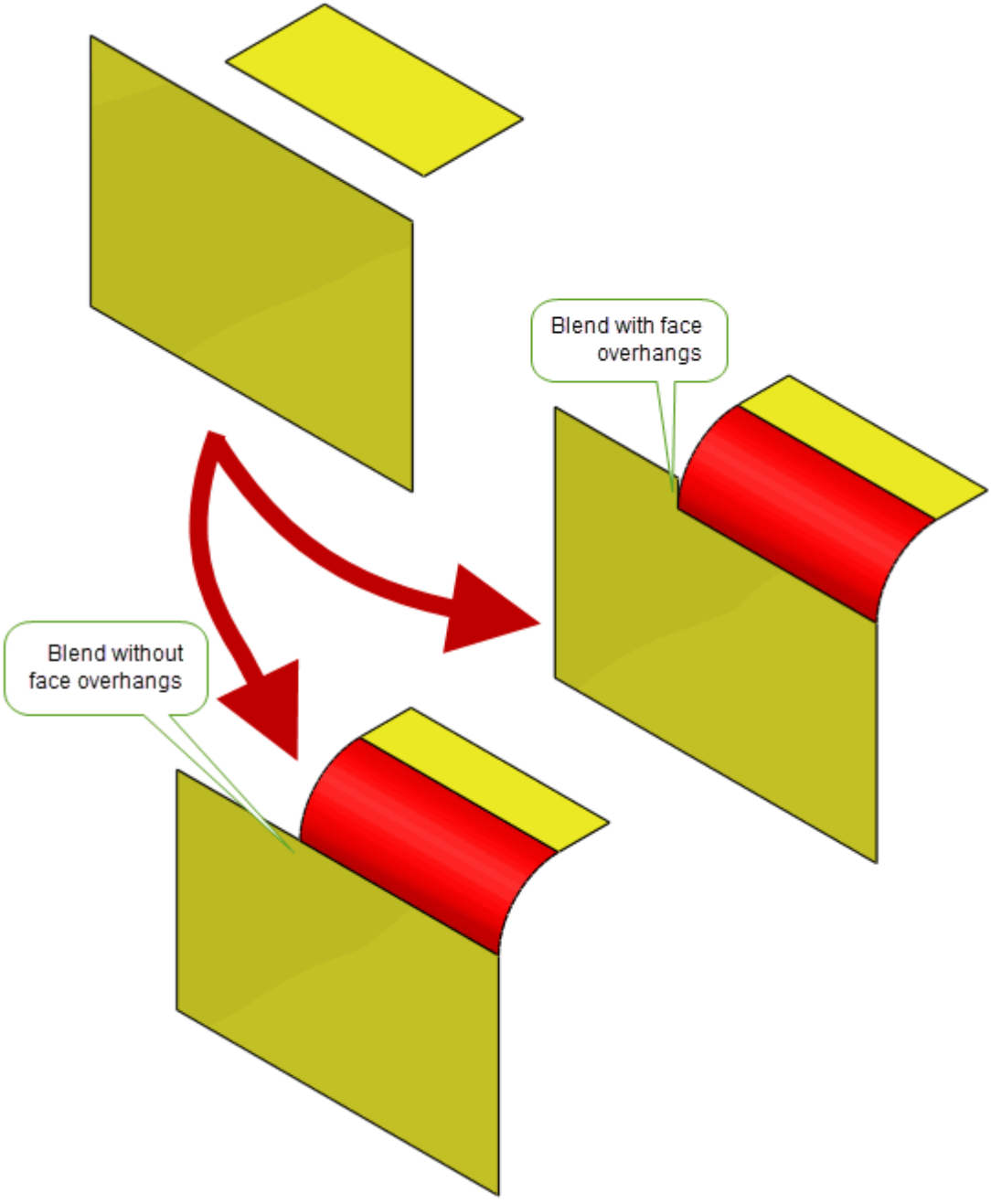


Figure 10–12 *Trimming blend walls with imprint complete*

10.3.5 Propagation and notches

As with edge blends, you can control a wide range of properties, such as propagation, behaviour at notches and so on.

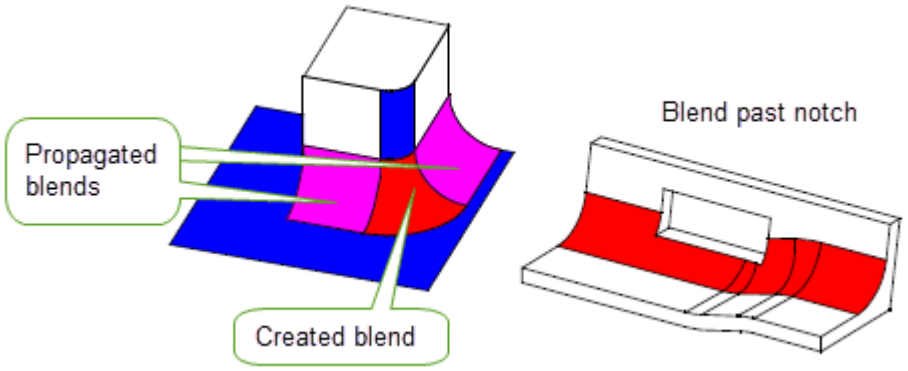


Figure 10–13 *Propagation and notch behaviour in face-face blends*

10.3.6 User-supplied surfaces

Rather than have Parasolid generate a blend surface, you can supply a surface to use in a blend operation. This is essential if your application makes use of specialized surface types that you want to use in conjunction with Parasolid's face-face blending functionality.

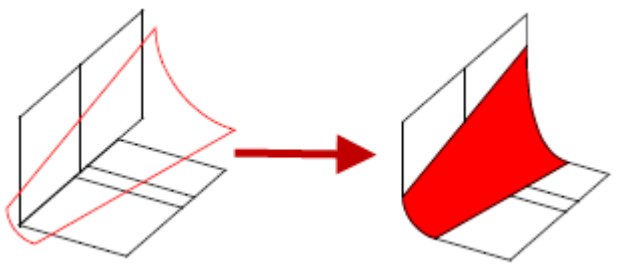


Figure 10–14 *Blending faces with a user-supplied surface*

10.3.7 Wire-face blends

It is possible to make a cliff-edge blend between a wall of faces and the edges of a wire body, known as a *wire-face* blend. There are a number of restrictions on the use of this functionality.

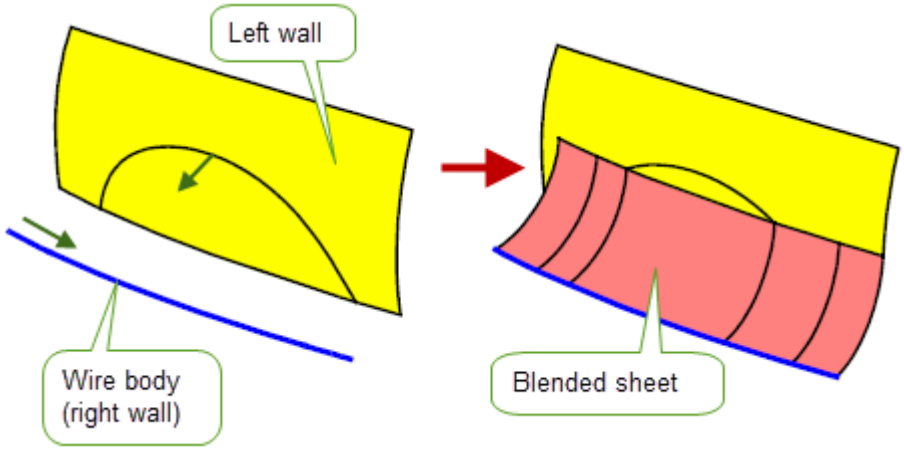


Figure 10–15 *Example of a wire-face blend*

10.4 Three-face blending

In addition to standard face-face blending, you can create three-face blends (sometimes known as *full round fillets*), which are blends between three sets of faces.

Figure 10–16 shows a part containing a number of ribs, each of which has a three-face blend along the top.

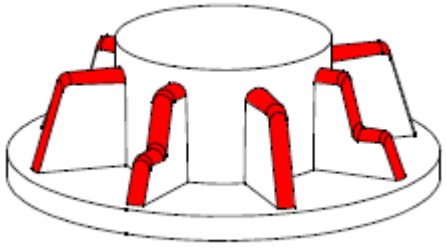


Figure 10–16 *Blending three sets of faces*

When creating three-face blends, three sets of walls are provided, rather than two. The resulting blend is trimmed and attached to the body automatically.

Three-face blending shares many options with face-face blending, as illustrated in *Figure 10–17*.

- Blends can propagate to adjoining faces in the body (even across sharp edges in the centre wall of the three walls provided).
- Blend faces can be trimmed to the boundaries of the blend walls.
- Blend walls can be trimmed to the boundaries of the blend faces.
- Blend faces can be capped, either from the body being blended or by using a plane.
- Blend faces can be stopped short using a limit plane.

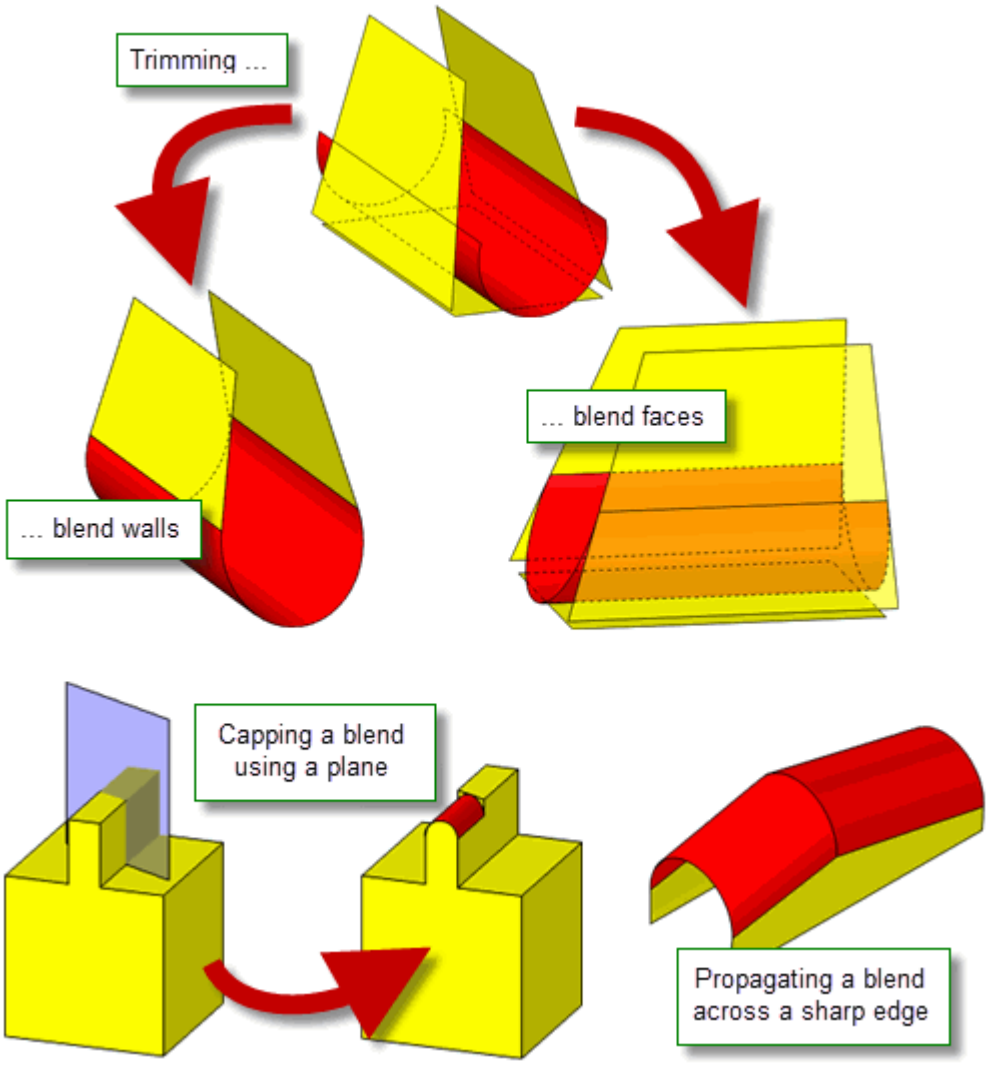


Figure 10–17 *Options when creating three-face blends*

Importing Foreign Data 11

11.1 Introduction

Sharing digital model data as part of the collaborative process of design and manufacture is becoming an increasingly important requirement in product life-cycle management. Organisations require reliable associative interoperability, regular exchange of data, and the confidence to be able to re-use their design information in all possible situations.

In an ideal world, companies sharing data would all use the same system, but this is usually not the case. Different organisations standardize on different systems, specific parts of the product life-cycle require applications suited to their particular needs (for example, FEA), and the formats these applications use to store model data may not be compatible.

Mindful of the requirements and realities of the PLM world, Parasolid supports several ways of reading in parts that were created using applications not based on Parasolid. You can use this functionality either to read data directly into your application, or to develop dedicated translators for importing data into Parasolid-powered applications.

In order to support the widest possible range of application types, Parasolid provides two general methods for importing external model data:

Method	To be used when ...
Trimmed surface	the originating system does not work with topological data. To use this method, the data must be supplied in the form of **trimmed surfaces** (surfaces bounded by loops of surface parameter curves). See Section 11.2, “Trimmed surface import”.
B-rep	the originating system knows the B-rep topology of the model. See Section 11.3, “B-rep import”

These methods are described in more detail in the rest of this chapter.

Note: You can also purchase dedicated two-way translator technology from Siemens Digital Industries Software. These solutions can be plugged into your application directly with no additional development beyond basic integration requirements. See Section 1.4, “The Parasolid product portfolio”, for more information.

11.2 Trimmed surface import

You should use the trimmed surface import method when the data you want to read into Parasolid was generated in a surface modeler that does not attempt to represent a solid model at all.

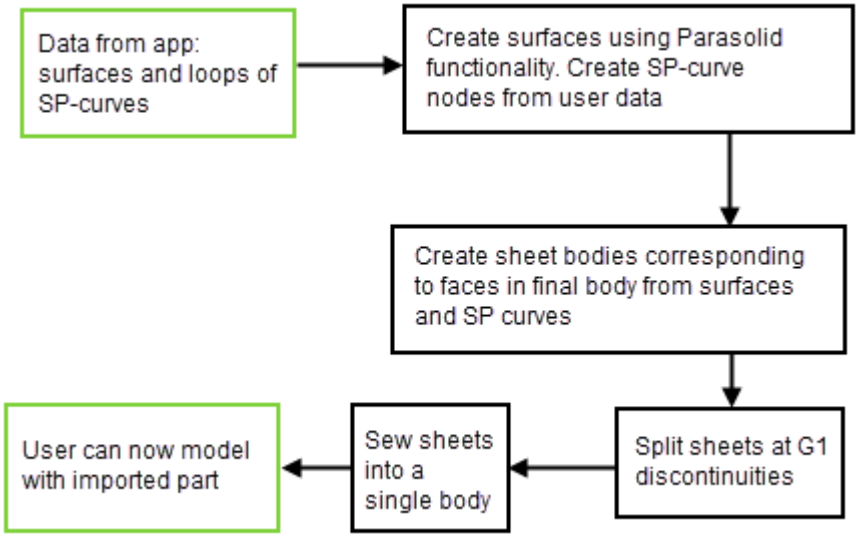


Figure 11–1 *Trimmed surface import method*

When your application creates sheet bodies, you can set a wide variety of options (such as tolerance) to reflect your understanding about the validity of the data being imported.

11.3 B-rep import

Even with data created using a true solid modelling application, there is a good chance that Parasolid may be unable to model reliably with the data as originally created in the other system, because different modelers have their own criteria about what represents a valid model.

Typically, data imported into Parasolid from another solid modeler will have a lower resolution than data created directly in Parasolid. Instead of a set of well defined touching faces, Parasolid perceives gaps and overlaps along every edge, and these must be dealt with before any downstream modelling operations can occur.

Parasolid uses Tolerant Modelling (see Section 3.6, “Accuracy of Parasolid models”) to modify the low resolution data by applying local tolerance information. This approach allows the intentions of the original designer to be preserved in the faces and surfaces of the model, while treating edges and vertices as entities derived from this data that inherit the resolution used by the original modeler. After applying tolerance information, downstream modelling operations may proceed without causing problems due to the nature of the imported data.

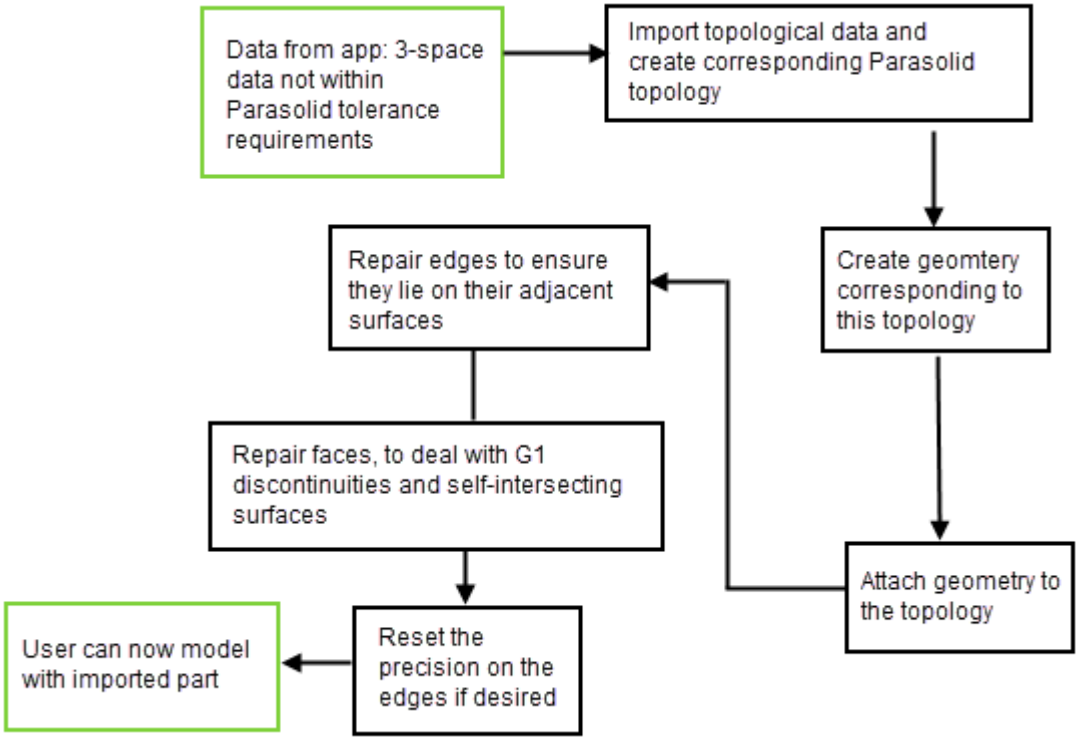


Figure 11–2 *B-rep import method*

If you need to import a geometry type present in the originating application that is not available in Parasolid, you can approximate it using a B-surface or B-curve.

Storing Data 12

12.1 Introduction

In Parasolid, the method of storing model data is broken down into two separate processes:

- Saving a part in a Parasolid session is referred to as **transmitting** a part
- Loading a part into a session is referred to as **receiving** a part.

These processes are sometimes collectively referred to as **archiving**. The functionality that Parasolid offers for managing archives is flexible enough to let your applications save model data to any relational or object-oriented database manager, or simply to files stored on disk.

When you store model data, the following items are saved:

- All topological and geometric entities contained in the part.
- The user field of each entity
- Identifiers of all entities that have them.
- Any features belonging to the part
- Any attributes belonging to entities in the part.
- Any construction geometry attached to the part

12.2 XT format and the Parasolid XT Pipeline

Parasolid saves all model data using the Parasolid **XT** file format. This is an open file format, the details of which have been published by Siemens Digital Industries Software, and are available either as a component in the standard Parasolid documentation suite, or on request.

XT format data from a model can be saved in either binary or text format, directly to disk, or to a database management system, and your applications can combine this information with their own application-specific data (for instance, to save feature information that is not encapsulated directly by Parasolid). Exchange of Parasolid XT data is 100% reliable: if your applications also provide users with the option to save pure XT data to disk – and most Parasolid-powered applications do – then you have an easy way of exchanging data with other users, not just of your applications, but of any Parasolid-powered application, without risking any loss.

Siemens Digital Industries Software uses the XT format as the basis for a concept known as the **Parasolid XT Pipeline**. This is the seamless exchange of digital models across enterprises, between different disciplines, using different MCAD applications from different vendors.

The value of the Parasolid Pipeline to end-users is increasing rapidly as more and more leading MCAD vendors market Parasolid-powered applications. Users of your applications can design and develop products more quickly, more cheaply, and with no risk of corrupting the digital data during product development. The Parasolid XT Pipeline also makes it possible for them to utilize several different applications for product design, allowing them to choose the best tool for each stage of the process so that they can optimise the design without fear of data corruption.

As enterprises continue to build their MCAD strategies around the Parasolid XT Pipeline, the value to application developers of adopting the XT format, with the concomitant prospect of hooking into those strategies in a seamless fashion, is compelling.

12.3 Data compatibility

XT data is **forward compatible**: any XT data can be loaded into any version of Parasolid that is newer than the one it was created in. For example, a part created in Parasolid V10 can be loaded into Parasolid V12. In the most extreme case, data created in Parasolid V1 can still be loaded into the current version of Parasolid.

In addition, because XT data is transparent across all supported computer platforms and operating systems, a part created on one platform can be loaded into a Parasolid session on any other supported platform.

From Parasolid V14.0 onwards, XT data is also **backward compatible** to at least the next major version. Thus, XT data created in Parasolid V18.0 can be received by Parasolid V17.0. This powerful capability means that data can be exchanged between different Parasolid-powered applications without those applications being required to use the same version of Parasolid.

Enquiring Model Data and Identifying Details

13

13.1 Enquiring model data

Parasolid provides a vast array of functionality to let you enquire about the status of any entity. This functionality includes the ability to:

- Enquire about model structure.
- Enquire about topological or geometric information.
- Analyse a model, to find information such as clashing entities, mass properties and model validity.

13.1.1 Information about model structure

You can find out about (and modify) the connections that exist between entities, such as the geometry that is attached to a piece of topology.

These functions access the Parasolid model data structure directly and are therefore very fast. You can use them to develop automatic feature recognition and decomposition applications.

13.1.2 Topological and geometric enquiries

Parasolid provides functionality to enquire about the topological information in a model, such as the properties of a loop, or whether a region is solid or void. In addition, a wide range of functionality is available to enquire about model geometry: for example, you can retrieve parameterisation information for a curve or surface, coincidence information for two pieces of geometry that are coincident, or sample points on a curve.

13.1.3 Model analysis

As well as getting information directly from a model, Parasolid provides sophisticated functionality to let you analyse any given model.

Analysis	Description
Clashing	Detect clashes between any entities in a model. Categorize any clashes in terms of whether one entity is contained completely inside another, whether the entities interfere with each other (that is, whether they cross over), or whether they are touching.
Containment	Detect whether a point is contained inside or outside an entity, or on its boundary.
Intersection	Detect intersections between pairs of any combination of curves, surfaces or faces.
Maximum/minimum distance	Find the minimum or maximum distance between any combination of geometric or topological entities.

Analysis	Description
Mass properties	Find mass property information for an entity, such as its length, mass, center of gravity, surface area or moments of inertia.
Model validity	Check whether a part is valid at any time: for example, during a sequence of modelling operations, or following a failed operation. Parasolid guarantees that operations on valid models will succeed or produce an intelligible failure code.
Redundant topology	Identify redundant topology such as wire edges and vertices on split rings on a body. Delete this redundant topology and merge together any remaining topologies that becomes mergeable as a result.

13.2 Identifying model details

Parasolid includes support for identifying model details, namely:

- Finding bounded face sets
- Identifying and classifying specific details in a body
- Identifying blends in a body
- Identifying the faces that underlie a blend

Face-sets identified by any of these methods might subsequently be deleted from the model, or used as input to other face-based functionality; *Figure 13–1* shows an example of identifying and deleting blends and holes from a body.

13.2.1 Finding bounded face sets

You can find distinct sets of faces in a body that are separated by a given set of edges.

13.2.2 Identifying and classifying specific details

You can identify distinct sets of faces in a body that represent a specific detail, such as faces that comprise a cylindrical hole. Parasolid returns face sets corresponding to individual details, which you might then decide to delete from the model.

13.2.3 Identifying blends

Parasolid allows you to identify blend faces that are within (or connected to) a set of input faces. These must be constant-radius rolling-ball blends. However, these need not necessarily be *defined* as constant-radius rolling-ball blend faces, so long as the actual blended face is of near constant radius (to within a specified tolerance). Once identified, blends can also be deleted, and the body repaired using dedicated blend deletion functionality.

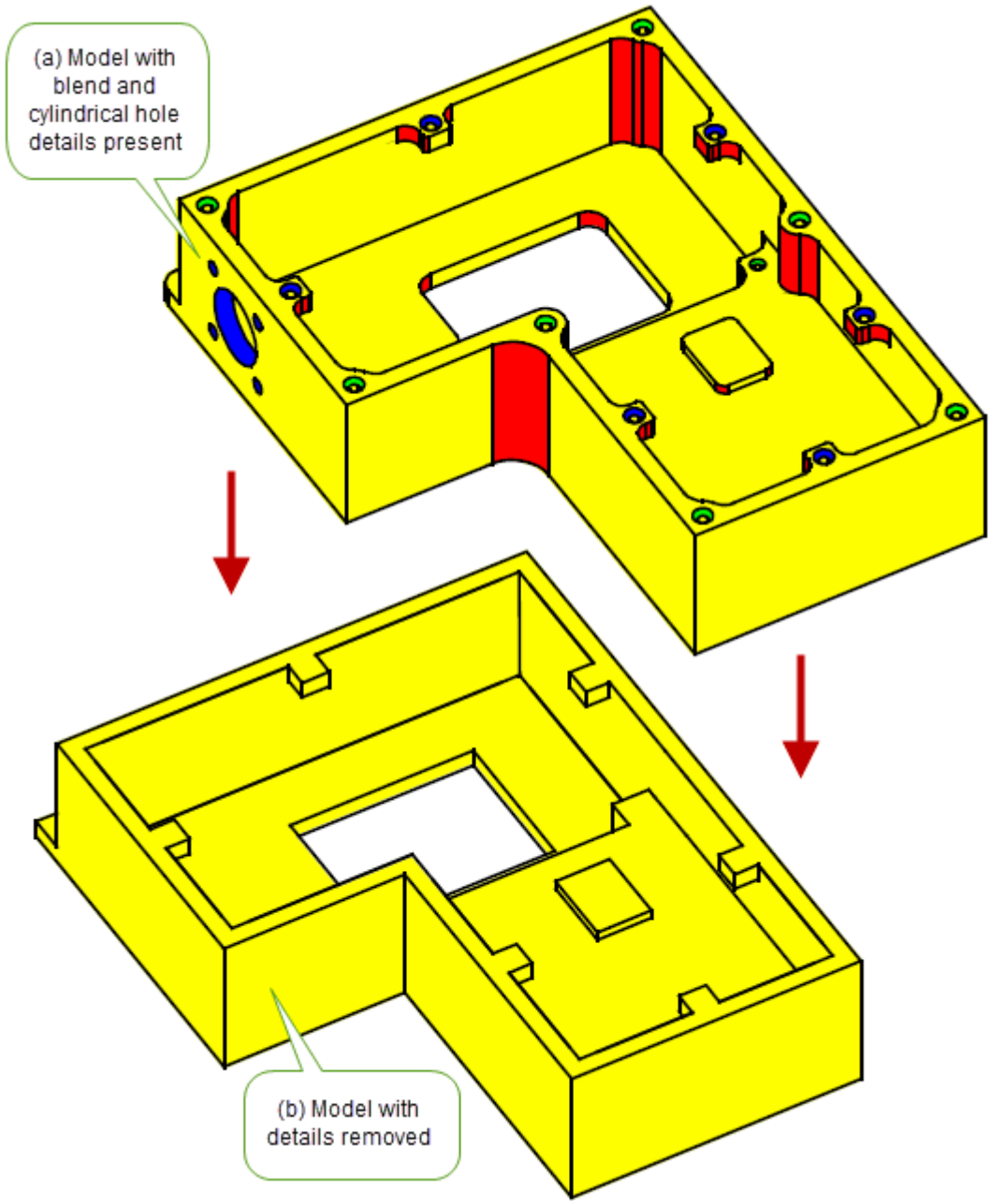


Figure 13–1 *Identifying and deleting blends and holes from a body*

Displaying Data 14

14.1 Introduction

Parasolid generates rendering, faceting and graphical data that you can use to produce pictures of the models you create. There are several steps to this process:

- First, the model is **rendered**: Parasolid generates graphical data for a particular entity or set of entities, optionally generating **facet data** for those entities if your application requests them.
- Next, Parasolid passes the graphical data to your application via the **Graphical Output**, or **GO** functions, that you supply as part of your application.
- Then, your application parses the data that was passed to it as required, finally passing the resulting data to a graphics library for display on the screen. Parasolid is compatible with all graphics libraries and user interface standards. This means that your application can use any graphics library you wish.

Parasolid provides a number of rendering and faceting functions: the one you choose depends on the type of display your application is generating and the method you are using to produce it. This chapter provides an overview of the functionality offered by these functions:

- Section 14.2, “Processing graphical data”, describes the steps required in order to generate graphical data and render it on an output device.
- Section 14.3, “Rendering pictures”, describes the different ways that model data can be rendered.
- Section 14.4, “Faceting pictures”, describes the different ways that facet data can be rendered.

14.2 Processing graphical data

Figure 14–1 gives an overview of the steps involved in creating and processing graphical data for use on an output device.

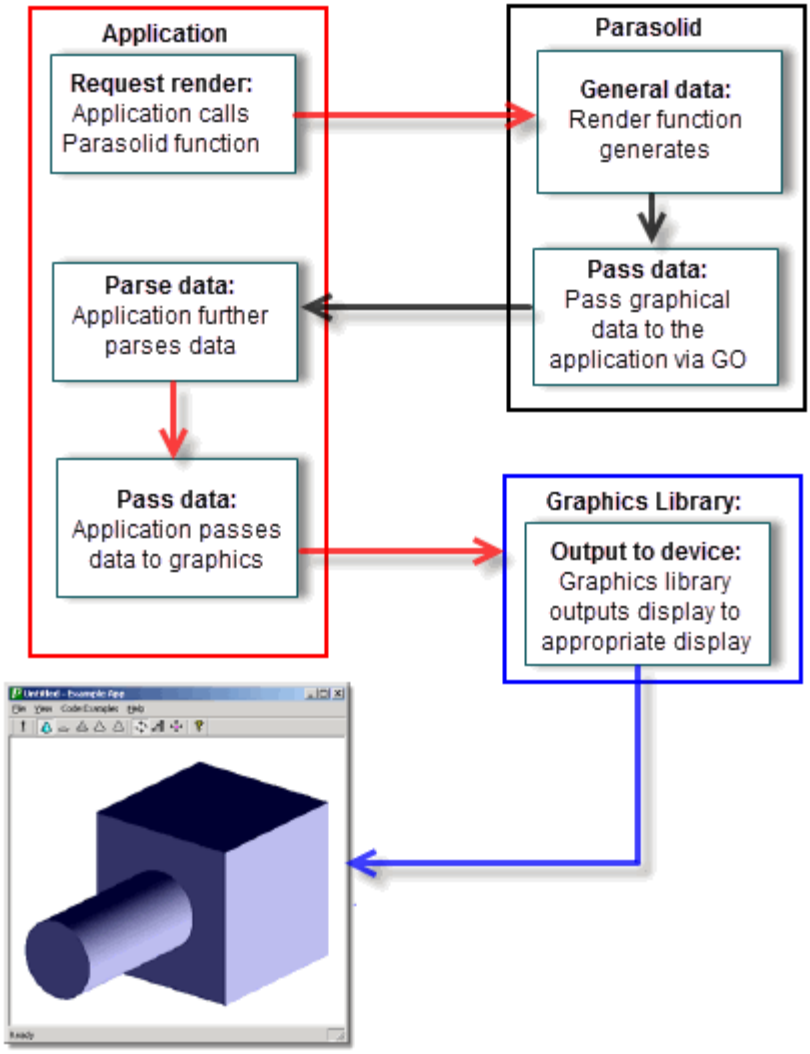


Figure 14–1 *Processing graphical data for display on an output device*

In order to produce a picture of a model, graphical data is sent from Parasolid to your application using the GO functions that you to supply. These functions encapsulate the graphical data created by Parasolid, and Parasolid calls them whenever your application calls a Parasolid faceting or rendering function.

Once your application has the graphical data, it can further parse it if required, depending on the nature of the data that was requested. For example, if hidden-line data was requested, then Parasolid outputs graphical data with visibility information attached to each data segment. Your

application can parse this information so that data segments that are flagged as hidden are displayed using a specific line style, such as that shown in *Figure 14–4*.

Depending on the functionality required by your application, pictures can be displayed on output devices (typically the user's computer monitor), or hard copy can be produced on a printer connected to the computer. In order to do this, you must pass the graphical data to a suitable graphics library in order to render the image. You need to provide the graphics library with your application; no graphics library is supplied with Parasolid itself. You can either write one yourself or, more commonly, use a third party library such as OpenGL or DirectX.

14.3 Rendering pictures

Parasolid can generate data that your application can use to create three kinds of picture: wire-frame, hidden-line and faceted. These are illustrated in *Figure 14–2*.

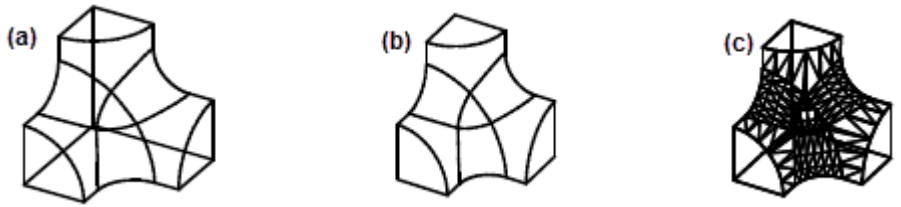


Figure 14–2 *Wire-frame (a), hidden-line (b) and faceted (c) pictures*

14.3.1 Wire-frame pictures

Wire-frame pictures depict models by drawing their edges. They can optionally include additional information to give the impression of the surfaces bounding the model, such as silhouette lines and hatching lines. However, they do not include information about which parts of the model are visible and which are hidden. *Figure 14–3* shows a wire-frame picture of a cylinder that could be produced by an application. It includes both silhouette and hatching information.

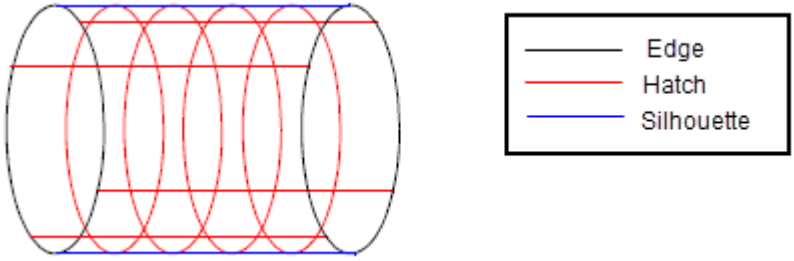


Figure 14–3 *Wire-frame rendering with silhouette and hatching information*

14.3.2 Hidden-line pictures

Hidden-line pictures distinguish lines that are visible in the current view from those that are not. Your application can produce pictures in which hidden lines are suppressed entirely (as in *Figure 14–2* (b)), or they may be output in a different style, for example, as dashes, or using a different colour, as shown in *Figure 14–4*. Hidden-line pictures always contain edges and silhouettes, and hatch-lines can also be included if required.

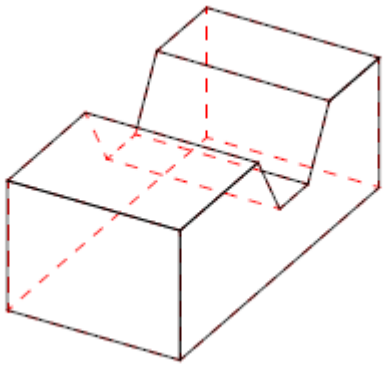


Figure 14–4 *Displaying hidden lines in a different style*

Parasolid's hidden-line routines use regional data, so that automatic form-feature recognition and decomposition applications can be implemented. The hidden-line-removal functionality supported by Parasolid also provides data essential for developing advanced "drafting-from-solid" applications.

14.3.3 Rendering options

Parasolid provides many options to control the way rendering data is generated. These include support for:

- Ignoring small features in order to simplify the result.
- Rendering a body as if it was transparent.
- Detecting the presence of separate overlapping bodies and rendering hidden-line views appropriately.
- Rendering planar and radial hatching information.
- Drawing unfixed blends as they would appear after they are fixed.
- Hierarchical output of edge, silhouette and hatch lines in a hidden-line drawing for more efficient processing.
- Supplying viewports to allow rendering of only a small portion of a larger body.

Figure 14–5 illustrates several of these options.

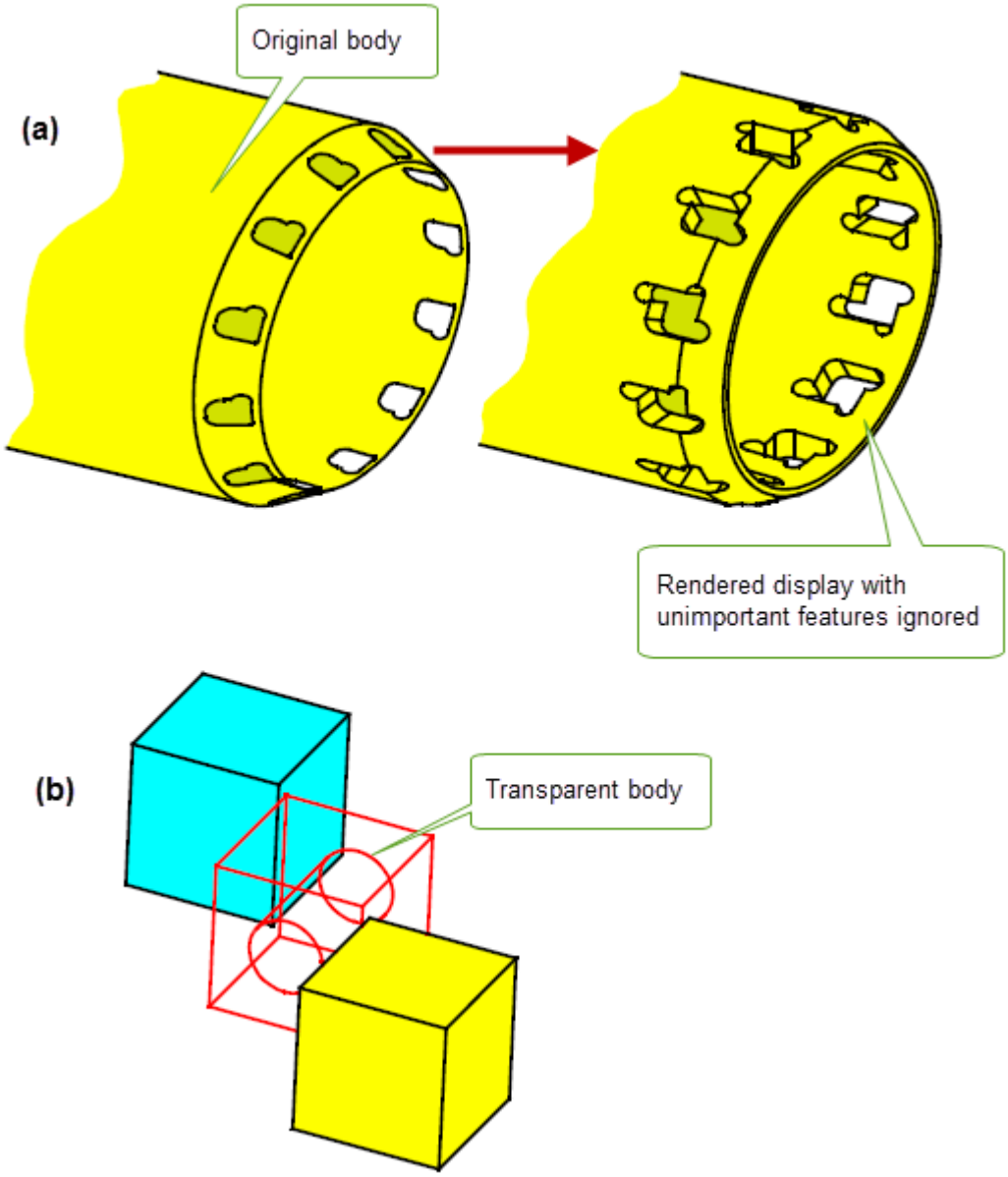


Figure 14–5 *Rendering options: (a) ignoring small features and (b) transparent bodies*

14.4 Faceting pictures

Faceting is a means of representing faces as a combination of vertices and straight edges that form planar or near planar facets. Facets generally coincide along adjoining edges, and each facet lies close to its “parent” face (that is, the face that it represents all or a part of), so that (unless you request otherwise) the edge of a face in the model corresponds closely to the edge of one or more facets.

Faceting does not change the entities in a model, it merely provides a different representation of the entities.

The most common use of facet data is to produce shaded 3D images of models: most graphics libraries require facet data in order to produce a shaded image.

Parasolid can generate graphical data in facet form so that your application can support the creation of faceted images if required. Facets can be output either individually, or as strips of connected facets: facet strips are more economical to represent, and many graphics drivers are optimised for this method of hardware shading.

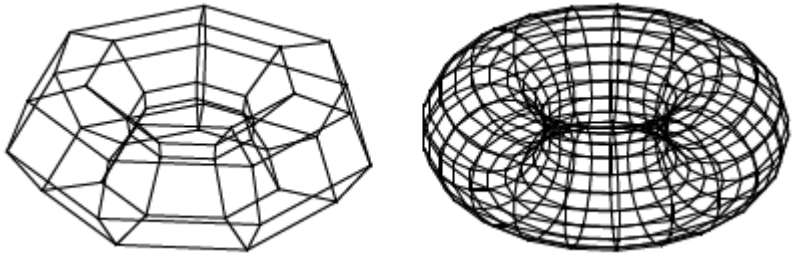


Figure 14–6 *Coarse and fine facet representations of a torus*

Note: Once calculated, facet data can be passed to your application in two ways:

- Via the GO, as is the case with rendering.
- Via an array table. This option does not use the GO.

Parasolid’s powerful faceting functionality can typically be useful in any of the following situations:

- To produce approximate shaded displays quickly where maximum resolution is not essential.
- To implement incremental graphics functionality that only updates the display for those parts of a model that have changed since the last update.
- For Stereo Lithography (SLA) applications.
- For “workcell simulation” applications.

14.4.1 Faceting options

Parasolid provides many options to provide control over the style and number of facets created when generating facet data. These include:

- Control over the size, shape and tolerances within which facets are created.
- Control over whether facets join together along geometry or topology.
- Control over the creation of facets in particular view directions, to increase the density of facets in an area of interest, or to prevent creating facets that will be hidden from view.
- Control over the generation of connectivity data for a facet mesh, so that your application has information on how facets are joined.

The precise combination of faceting options available depends on whether you return facet data to your application using the GO, or using an array table.

Application Support 15

15.1 Introduction

Any Parasolid-powered application needs to provide functionality to the user that extends beyond core modelling operations. Although you could develop much of this functionality from scratch yourself, having built-in support from Parasolid makes the task easier, and provides better and more reliable performance to the user. In recognition of this, Parasolid provides tools to help you implement functionality in a range of key non-modelling areas. This chapter describes the most important of these.

- Section 15.2, “Attaching attribute information”, explains Parasolid’s support for attaching arbitrary information to modelling entities that is either provided by the user or by your application.
- Section 15.3, “Partitions and roll-back”, describes features that can be used to let your users undo and redo operations, and recover elegantly from errors.
- Section 15.4, “Groups”, explains how you can create collections of associated entities for further processing by your application.
- Section 15.5, “Tracking entities”, describes the information recorded by Parasolid during a session that you can use to provide feedback in your application.

15.2 Attaching attribute information

Attributes are data structures that can be attached to Parasolid entities to supplement the standard data structure defined by Parasolid. Your application can use attributes to store arbitrary information that is associated with the entity to which it is attached. This is a convenient way of representing information such as constraints, manufacturing data, analysis data, references to entities in application data structures, and so on. Any of the following types of information can be stored in an attribute:

- real
- integer
- string
- Unicode string
- vector
- coordinate
- direction
- axis
- pointer

Attributes can be attached to any of Parasolid’s geometry and topology types. Once attached to a Parasolid entity, an attribute remains attached until either the entity is deleted or the attribute is deleted. When a Parasolid model is archived, any attributes attached to it are also archived.

Attributes are useful in almost any Parasolid-powered application. At the simplest level, they can be used to store information such as colour, or notes written by a user. They are a particularly

powerful feature when used in concurrent engineering systems where the Parasolid model is the basic building block of a complex product definition model.

15.2.1 Attribute definitions

An application can have many different types of attributes at its disposal. The structure of each type (in terms of the type of data it can hold, where it can be used and so on) must be clearly defined using an *attribute definition*. Each attribute definition records information such as:

- The attribute *class*, which defines the attribute's behaviour when the entity to which it is attached is involved in modelling operations
- The types of entity to which the attribute may be attached
- The order and types of attribute *fields* in which data is held

Your application can create as many different attribute definitions as required. Once an attribute definition has been created, attributes of that type can be attached to entities.

15.2.1.1 System-defined attributes

As well as specifying your own attribute definitions for use in your application. Parasolid comes pre-defined with a number of attribute types, known as *system-defined attributes*. These attribute types are used by Parasolid itself (for example, mass properties calculations use attributes that hold density information), and so should not be redefined in any way, but you are free to use them in your application, if you wish.

The system defined attributes provided by Parasolid include:

- Colour
- Density of various topological entities
- Transparency, translucency and reflectivity
- Planar, radial and parametric hatching

15.2.2 Using attributes

Parasolid provides a comprehensive range of operations for creating and managing attributes and attribute definitions. You can create attributes, attach them to entities, add information to them, retrieve information from them and delete them.

Parasolid also offers higher level attribute management functions, allowing you, for example, to filter attribute information so as to retrieve all attributes on a part based on some property of the attribute, such as its class or a specific value in one of its fields.

15.2.3 Attribute behaviour

The class of an attribute definition indicates what happens to an attribute of that type when its owning entity is involved in a modelling operation. For example, if a face has an attached attribute, and that face is split into two as a result of imprinting an edge, what happens to the attribute? Under different circumstances, the attribute may be deleted, remain on the original face, or (as shown in *Figure 15–1*) be copied so that it exists on both faces, depending on the class of its attribute definition and the nature of the event.

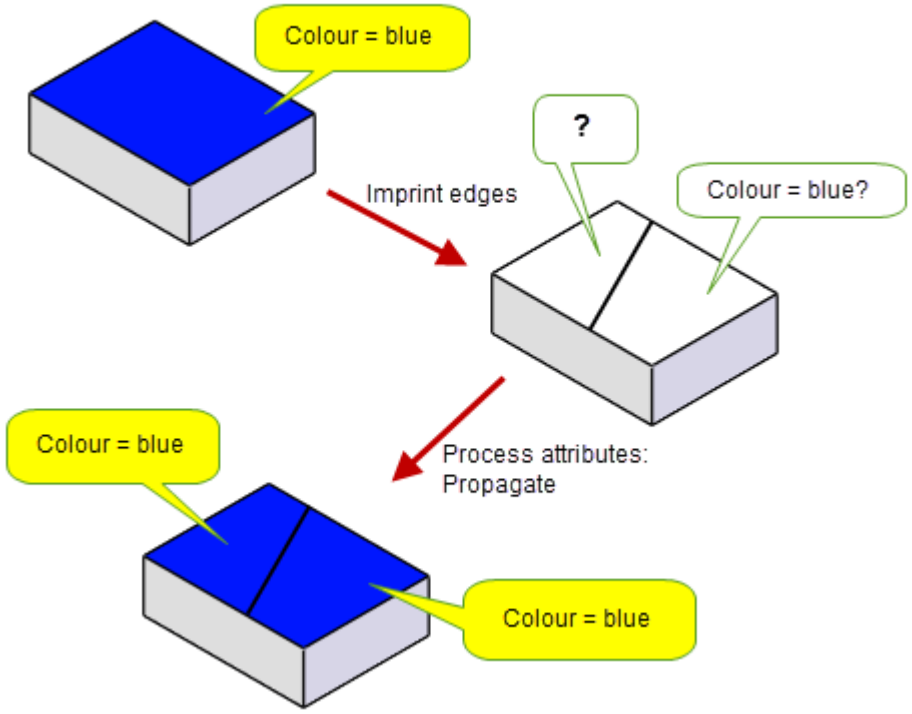


Figure 15–1 *Attribute behaviour when entities are modeled*

Parasolid uses an attribute definition's class to specify how to process attributes when their owning entities are involved in a modelling operation that cause any of the events shown below:

Event	Behaviour
Transform	An entity may be transformed via any of reflection, rotation, translation or scaling. Each type of transformation may independently be applied to, have no effect on, or delete an attribute attached to a transformed entity, depending on the class.
Split	A topological entity may be split into two or more entities of the same type. For example, a face may be split by a boolean, scribing, or a local operation. When an entity is split, each attribute can be either deleted or propagated to all resulting entities, depending on the class.
Merge	Two topological entities of the same type may be merged to form one. Attributes on those entities may be kept if they are equal and deleted if they are not equal. Alternatively, they may simply be deleted regardless of the data they contain.
Transfer	An entity may be transferred between, into or out of parts. For example, shells, faces and loops may be transferred between bodies during a boolean. Attributes attached to such entities may be deleted or left as they are, depending on the class.

Event	Behaviour
Create/Delete	When creating an entity, either with a direct call to a Parasolid creation function, or as an effect of a modelling operation such as a boolean, the entity has no attributes. When an entity is deleted, so are its attributes.
Change	An entity may be changed in some other way that is not expressible in terms of the above events, though may be induced by one of them. For example, reflecting a body induces a change of handedness (what was on the left is now on the right) on all topological relationships between entities in the body. To allow for this, Parasolid supports a general change event, which may delete associated attributes or alternatively have no effect on them, depending on the class.

The Parasolid documentation suite contains a full list of all the attribute definition classes available, and the effect on attributes of those classes whose owning entities undergo any of these events.

15.2.4 Specifying your own attribute behaviours

Instead of relying on Parasolid's normal attribute processing behaviour, you can specify your own by defining *attribute callback functions*. These let you provide your application with attribute processing behaviour that is more specifically tailored to your requirements than that available using Parasolid's normal attribute handling.

You associate a callback function with a specific attribute definition, for a particular event. When the owning entity of an attribute of that type undergoes the event, the callback function is called. The events for which you can define callback functions are:

- split
- merge
- delete
- copy
- transmit
- receive

There are two types of callback function:

Normal callbacks These functions can modify and enquire about the attributes or other parts of a model as necessary. When a normal callback is invoked, the action it carries out replaces Parasolid's normal attribute processing (as described in Section 15.2.3).

Read-only callbacks These functions cannot modify the attributes or other parts of a model in any way. After a call to a read-only callback returns, Parasolid processes the attributes according to their class, in the usual way (as described in Section 15.2.3).

15.3 Partitions and roll-back

All but the most trivial applications should be able to undo user operations and return the application to the state it was in at some previous point in time. This is especially true of design environments, where users are likely to want to try out different ideas and experiment with tools and techniques. In addition, users expect sophisticated error recovery, and this requires an application to be able to return an entire session to a previously known stable state.

Parasolid addresses both these needs using a facility that is known generically as *roll-back*. There are different forms of roll-back, which between them provide comprehensive support for each of the situations mentioned above.

If you take advantage of Parasolid's roll-back functionality, there are many ways that you might choose to make this available to the user. For example:

- To automatically revert to a stable state after any failed operation.
- To recover and save data in the event of an emergency (such as a system crash).
- As a simple, one-step undo facility to be called by the user.
- To mark the major milestones in the design history of a part within a feature modelling environment (to enable sophisticated user-controlled undo and redo facilities, and the implementation of history trees).

15.3.1 Concepts

A *partition* is a self-contained collection of bodies that can be manipulated independently of other bodies. At any one time, many different partitions may exist. Parasolid provides two different types of partition:

- Standard partitions contain the information required for sophisticated undo and redo facilities, together with history management.
- Light partitions contain just the information required to implement undo functionality, thereby requiring significantly less memory overheads.

A *session* is the collection of all partitions. There can only ever be a single Parasolid session open at any one time; each time a new session is started, a single partition is created automatically.

Parasolid supports two types of roll-back, both of which are available through the same general mechanism.

- *Partition-level roll-back* allows your application to roll an individual partition back or forward independently of other partitions.
- *Session-level roll-back* allows your application to roll the contents of the entire session back or forward, including all partitions in the session.

Note: For its roll-back mechanisms, Parasolid uses terminology familiar to database users: *rolling back* refers to returning to an earlier state in the session or partition (i.e. undo), and *rolling forward* refers to moving to a later state in the session or partition (i.e. redo).

Partition-level and session-level roll-back both work by employing a system of marks. Rather like tying knots in a piece of string, your application can save marks at significant points in either the modelling session or the current partition. At any point, you can move to one of these marks, thereby restoring the session or partition status, as appropriate.

- Marks in a session are referred to as *session marks* (or sometimes just marks).
- Marks in a partition are referred to as *partition marks* (or p-marks).

15.3.2 Partition-level roll-back

Partition-level roll-back can be used primarily as a means of providing your application with undo and redo functionality, allowing your users to move back and forth in a history tree of modelling operations. Depending on the amount of p-mark data your application maintains, this functionality can be as sophisticated at the user level as you require it to be: anything from a simple one-step undo facility to a complete implementation of a comprehensive history tree.

Partition-level roll-back is also crucial for enabling feature modelling applications to perform model updates after model parameters are changed: a feature modeler needs to roll back a model to the point where a feature was added, reapply the modified feature, and then reapply subsequent features to generate the updated model.

Individual bodies or collections of bodies can be rolled back independently, so updating one model in a session does not affect other models. Branching roll-back of bodies – several alternative updates to a given body – can also be implemented.

In the context of partition-level roll-back, a *partition* is a simply a self-contained collection of bodies which can be rolled back and forward independently of other partitions. When a Parasolid session is started, a single partition is created automatically. Your application can create additional partitions when required, so that a given Parasolid session may eventually contain many partitions.

You can use partitions to collect together bodies in any way you wish. For example, you may decide that each partition in a session should contain only one part. Adding partition-level roll-back to your application would then mean that your users could make changes to individual parts and roll those changes back or forward independently of other parts in the session.

Only one partition can be *current* (that is, in use) at any time. As the Parasolid session progresses, your application should create p-marks in the current partition, thereby creating a series of checkpoints in the history tree that can potentially be rolled back to later in the session.

If new entities are created that do not reference existing entities, they are automatically placed in the current partition.

Parasolid provides a full range of functionality to let you manage the partitions in a session, including the ability to:

- Create, delete, copy, and merge together partitions.
- Roll between any two partition marks.
- Receive and transmit partitions.
- Return information about the different entities in a partition.
- Perform “what-if” calculations, to find which entities would be created, deleted, or modified when rolling between different points in a session.

Figure 15–2 shows an example of how you might implement partition-level roll-back and present it to the user. The figure illustrates a number of stages in the creation of a part, showing the

different design decisions that the user has gone through, and how your application can use partition-level roll-back to support those design decisions:

- First, the user creates a block. After the operation has completed, your application creates a p-mark: pm0
- Next, the user adds a boss to this block. Your application then creates pm1.
- Then the user adds a cliff-edge blend around the boss. Your application creates pm2.
- At this point, the user decides that one edge of the block should have been blended before the boss was applied, and uses the history facility that is available in your application to go back to the initial stage. To achieve, this, your application rolls the partition back to pm0.
- The user adds an edge blend to the block. Your application creates pm3.
- The user adds the same boss to the block that was added earlier. Your application creates pm4.
- The user then adds a blend around the boss, this time opting to let the blend overflow into the blend created earlier. Your application creates pm5.
- The user decides that this blend is not what was desired and goes back a stage. Your application rolls back to pm4.
- Finally, the user adds a cliff-edge blend around the boss. Your application creates pm6.

At this stage, Parasolid can potentially roll between any of the partition marks already created (unless your application has explicitly tidied up some of the p-mark data along the way). Depending on the history facility you have implemented in your application, this can give the user the potential to jump to any stage in the design process at will.

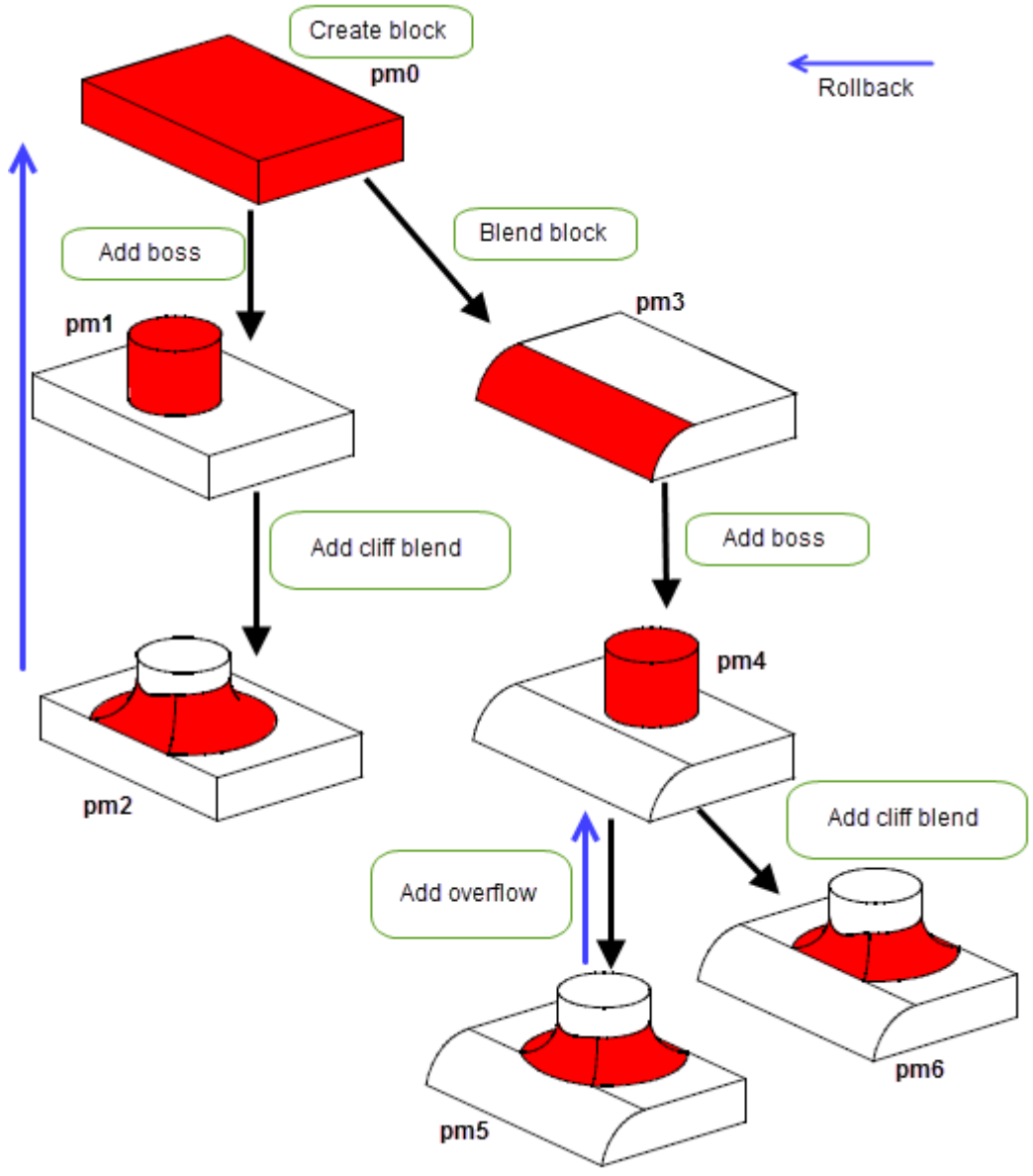


Figure 15–2 *An example of partition-level roll-back in use*

15.3.3 Session-level roll-back

In addition to partition-level roll-back, session-level roll-back lets you roll back the contents of an entire session, including partitions, session parameters and modelling parameters. Session-level roll-back can be used as part of an error handling strategy for your application, giving you the ability to return the session to a stable state whenever an operation fails.

The session-level equivalent of a p-mark is a *session mark*. These record the state of the entire session and, as with p-marks, you can return the session to any recorded session mark.

15.4 Groups

Groups are used to collect entities together so that they can be modelled as if they were a single entity. For example, you can put all the faces of a particular feature, such as a boss, into a group. You can use groups to supplement the Parasolid data structure to suit your application. Parasolid groups enable form feature applications and constraints managed applications to be easily developed. The way you use this functionality depends upon your specific requirements.

15.4.1 Characteristics of groups

Each group is owned by a specific part and can only contain entities that belong to that part. In addition, each group has a specified class that determines the classes of entities that can be added to the group. You can use the group class to prevent inappropriate entities being added to a group.

A given body or assembly can have any number of groups, which can be of the same or different classes. Similarly, a given entity can be added to any number of valid groups, as determined by the group class.

Parasolid tracks and updates entities in groups during modelling operations. A documented set of rules determines how group membership is affected if any entity within a group is merged with other entities, split or deleted.In addition, Parasolid lets you define rules based on certain criteria to enable you to intelligently manage the number of groups defined on a part, for example by automatically deleting groups when they are empty.

15.5 Tracking entities

Keeping track of changes to entities during a modelling session is an important factor for most 3D modelling applications. When new entities are created as a result of a modelling operation, your application may need to keep a record of how that entity came into being. Tracking entities is particularly important in any feature-based system. Likewise, an application may need to track entities in order to update graphical display information.

Parasolid offers a number of different ways that you can track activity during a modelling session. Which of these you use (or which combination of them you use) depends almost entirely on what information you need to track.

- Parasolid functions return generic *tracking structures* that match topological entities in the original body with the corresponding topological entities in the result body, together with the type of operation that was performed on the original entities.
- If you have very specific requirements, you can also record tracking information yourself by attaching attributes to entities as you progress through a session. See Section 15.2, “Attaching attribute information”, for more information.
- If preferred, you can automate information tracking via attributes by defining attribute callbacks that are called automatically when particular modelling operations occur. See Section 15.2.4, “Specifying your own attribute behaviours”, for more information.
- If necessary, you can use groups in conjunction with entity tracking. See Section 15.4, “Groups”.
- When working with partitions, you can track entities created, deleted, or modified prior to creating a p-mark.
- Finally, the Parasolid *bulletin board* is also available as an alternative method for tracking the changes that occur to entities during a modelling session.

Writing Parasolid Applications

16

16.1 Introduction

Parasolid is supplied as a single library of C-callable functions, together with associated header files, and in order to perform solid modelling operations, you call these functions from within your application code. Before this can be done, Parasolid needs to be integrated with whatever development environment you are using. This is simply a case of specifying the location of the required library, headers and so on. In addition, your application needs to include code that ensures that it can interact with Parasolid correctly.

This chapter gives you an overview of how you approach the development of a Parasolid application, and discusses:

- What code you need to supply before you can make calls to Parasolid operations.
- The overall structure of the Parasolid API, and how it can help you to make the development process easier.
- What design decisions you need to take before you progress too far with your application development.

You can deliver Parasolid to your customers with your application just by including a few files (including the library file itself) with the other components of your product. Any commonly available installer package will ensure that it gets installed on the customer's computer in the correct location for the operating system they are running.

16.2 Downward interfaces

Your application must control all interaction between Parasolid and the operating system. In order to do this, you need to provide a number of modules, referred to generically as *downward interfaces*. There are usually two such modules, each of which defines a small suite of functions:

Module	Description
Frustrum	Contains functions that deal with: ■ **File handling:** Saving and retrieving Parasolid part files and other data. ■ **Memory management:** Allocating memory for internal calculations and data structure storage.
Graphical Output (GO)	Functions that encapsulate the graphical information output by Parasolid's rendering functions in a form suitable for passing to a graphics library, such as OpenGL or DirectX. See Chapter 14, "Displaying Data", for more information.

If you also wish to support Parasolid's roll-back functionality (see Section 16.4.2), you also need to provide a **delta frustrum**, containing functions specific to that functionality.

You need to provide the required downward interfaces before you can successfully call Parasolid operations from your own application code. Parasolid ships with example code to help you write your own versions of them.

16.3 Calling Parasolid functions

Calling Parasolid functionality from your application code is a simple process. Parasolid has a long history of strong design principles and offers its functionality through a consistent interface that is accompanied by complete and thorough reference documentation. This consistency gives you the freedom to structure your own application code in the way that suits you best, without having to make compromises because of any requirements imposed due to integration with Parasolid. This section introduces you to the structure of the Parasolid API.

16.3.1 Design of functions and related data

Parasolid entities are organised in an object-oriented class hierarchy such that a function that is called on a given class can also be called on all its subclasses. For example, any function that is defined on curves can also be called on circles, lines, B-curves and so on.

Each function has a fixed set of arguments: some of these are used to supply data, and others are used to return information. Arguments are expressed as simple values, arrays, and structures of the types described in Section 16.3.2. The overall relationship between these different elements is illustrated in *Figure 16–1*.

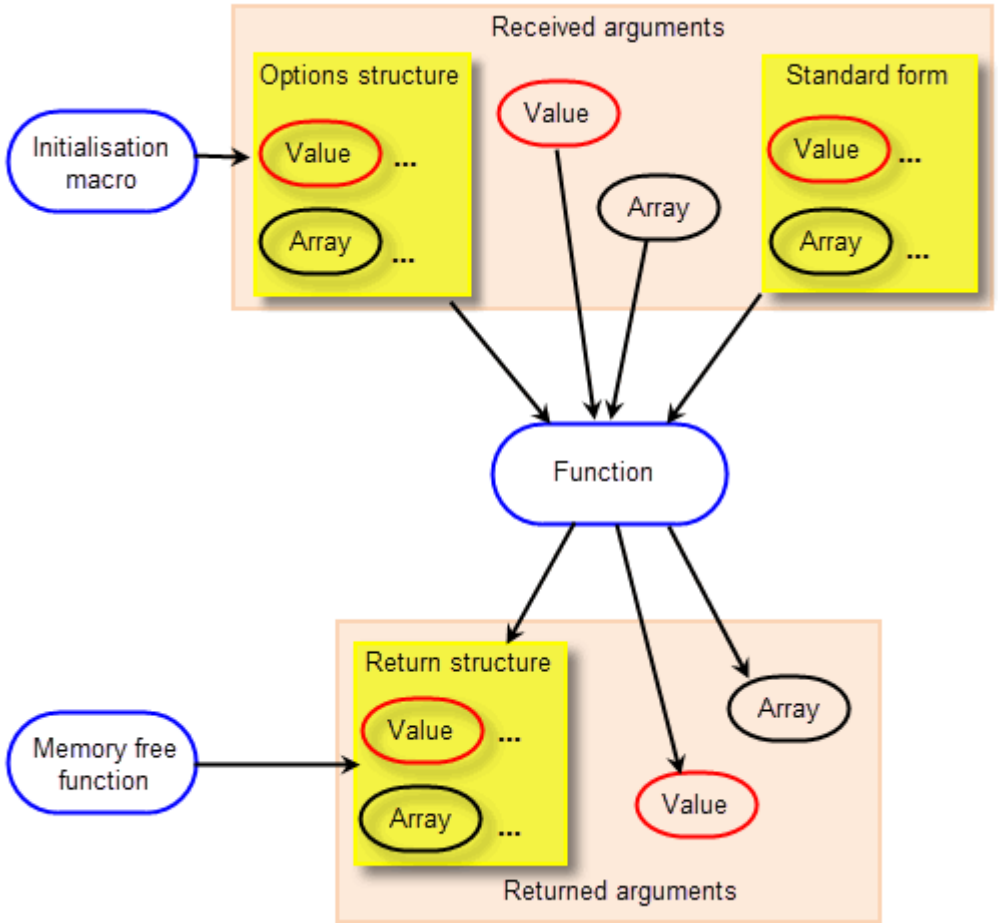


Figure 16–1 *Structure of a typical function group*

You must set each argument explicitly when calling Parasolid functions, rather than omitting them completely. This is made easier by the use of option structures and initialisation macros, as described in Section 16.3.2.

For many functions, helper functions are provided that free the memory used for return arguments when it is no longer required: see Section 16.3.3.

16.3.2 Structures

Much of the information passed in Parasolid function code is collected together in related groups, or structures. There are three basic structures that you use in your code.

Structure	Description
Options structure	These structures are used to contain optional arguments and option switches that are passed to Parasolid functions. Every options structure has an accompanying macro that you should use to initialise all the fields in the structure to their default values, prior to setting any specific values that you require.
Return structure	These structures are used to pass related information back to your application. For example, tracking information is passed back to your application using a return structure.
Standard form	These structures are used as templates for specifying information when you create entities. For example, when creating a circle, you need to pass a standard form into the relevant creation function that specifies the radius, center point and reference direction for the circle.

16.3.3 Memory management

Return arguments in Parasolid functions consume variable amounts of data that are returned from Parasolid as C arrays. While Parasolid allocates most of the memory required automatically, your application must ensure that memory is freed correctly when it is no longer required. Parasolid provides functionality to let you allocate and free memory.

16.3.4 Naming conventions

All Parasolid interfaces follow strict naming conventions that make them easy to use and identify. For example, function names always include the class of entity on which they can be called (BODY, FACE, etc.), as well as words that describe the operation to be performed. Different types of structures can easily be identified according to the suffix used on the structure name, and options structures and their macros are named according to the functions to which they apply.

16.3.5 Calling Parasolid from .NET applications

A Microsoft .NET binding DLL for Parasolid is available. This allows Parasolid-based applications to be written in the C# programming language.

16.4 Implementation decisions

Like any complex application, when you start building a Parasolid-based application you need to plan what functionality you want to provide to your users. Many of these considerations affect the whole of your application, rather than just parts of it, and so you need to make decisions about them early on in the development process. This is equally true whether you are developing an application from scratch (in which case you need to decide the strategy you want to follow), or whether you are integrating Parasolid into an existing application (in which case you need to decide how Parasolid can best support your existing strategy, or consider improvements in your current design that Parasolid makes possible).

This section discusses some of the more important implementation decisions you will need to make early on in the development process, and what options Parasolid provides.

16.4.1 Managing errors

All Parasolid functions return an error code as the function value to the calling routine. This value broadly indicates whether the intended operation has succeeded (if zero) or failed (non-zero); in the case of failure, the error code describes the probable cause. In other situations, more details concerning the nature of the failure can be returned via the output arguments of the function. In these cases, a value of zero is returned as the error code, and the corresponding output arguments must also be examined.

To ease the handling of situations where non-zero error codes are returned, your application can register an error handling function with Parasolid. This function is called automatically whenever an error with a non-zero code occurs, and can be used to perform any clean-up operations needed as a result of the error.

If you wish, your application can also register signal handlers with the operating system in order to recover from run-time errors and enable user interrupts.

16.4.2 Roll-back

As discussed in Section 15.3, Parasolid provides two types of roll-back.

- Session-level roll-back allows your application to roll back the contents of the entire session.
- Partition-level roll-back allows your application to roll back an individual partition independently of any other partitions.

You need to decide which, if either, of these mechanisms you want to implement, and whether you want your users to be able to roll sessions and partitions forward, as well as backward.

For example, you could implement partition-level roll-back so that a new partition is created for each individual part. Users could then make changes to individual parts and roll them back or forward without affecting the other parts in the session.

16.4.3 Tracking entities

Your application may need to track various entities within any Parasolid session. For example, tracking is required in order to:

- Update graphical display information
- Track entities in a feature-based system

Parasolid provides several methods for tracking entities, discussed in Section 15.5. You need to decide which of these methods is best suited to you.

16.4.4 Session parameters

There are a number of session-level options you can control that alter the behaviour for the whole Parasolid session. You might decide to give your users control over these options (via a scheme of application preferences), or you might decide to hard code them into your application. Examples of these options include:

Parameter	Description
Continuity and self-intersection checking	Whether checks are performed by various operations, and whether to attach discontinuous or self-intersecting geometry to topology. See Section 3.7.3.
General topology	Whether to create general bodies from boolean operations. See Section 3.7.2.
SMP	Whether to use SMP when more than one processor is available. See Chapter 17, “Multi-Processing Support”.

Multi-Processing Support

17

17.1 Support for multi-processing in Parasolid

Parasolid provides several separate mechanisms to help applications benefit from multiple processors on the host machine.

- Symmetric multi-processing (SMP): See Section 17.2, “Symmetric multi-processing in Parasolid”, for more information.
- Multiple application threads: See Section 17.3, “Calling Parasolid from multiple threads”, for more information.
- Partition locking: See Section 17.4, “Locking partitions to threads”, for more information.

17.2 Symmetric multi-processing in Parasolid

Parasolid provides multi-processor support via its Symmetric Multi-Processing (SMP) mechanism.SMP allows key algorithms beneath the Parasolid API to make use of multiple processors if they are present on a host machine.

A number of functional areas within Parasolid have been adapted for SMP, making it possible to reduce the execution time of certain operations, thereby improving the performance of Parasolid when running on multi-processor systems. For example, if your application needs to calculate the mass of a model. Parasolid will perform this calculation face-by-face and if it uses SMP then calculations on the different faces may be calculated on different processors.

For a list of areas that are SMP-enabled, see Section 17.2.3, “Functional areas that are SMP-enabled”.

Parasolid SMP is enabled on the most widely used Parasolid platforms - Windows, Linux and MacOS.

Currently, Parasolid can create a maximum of 8 threads for SMP work.

17.2.1 Expectations from Parasolid SMP

The SMP mechanism facilitates a shorter execution time for key Parasolid algorithms. Significant improvements in execution time can be achieved, though since the improvement depends on the nature of the operations being performed, it is not possible to quote a figure for the general performance improvement achieved by using SMP.

SMP is implemented beneath the API level, typically in low-level internal algorithms. These algorithms typically only form part of the code path called by a given API function and may be invoked to different degrees depending on the nature of the received arguments passed by the application to the function. The reduction in execution time is affected accordingly. For these reasons, you should not expect an automatic 50% speed up at the function level when enabling SMP on a two processor machine.

Similarly, due to the nature of SMP, the scaling of performance with an increased number of processors is not expected to be linear. For example, if you see a 20% performance improvement for a particular function call on a 2 processor machine, this does not automatically mean that you would see a 40% performance improvement on a 4 processor machine.

17.2.2 Thread locking

The overhead of thread locking can mean that isolated function calls on some platforms will actually run slightly slower with SMP enabled than with SMP disabled. We strive to minimise this in our implementation and work in collaboration with hardware vendors to do the same. If you do find functions that run excessively slowly, the Parasolid team is happy to investigate these on a case by case basis.

17.2.3 Functional areas that are SMP-enabled

The following table shows the main functional areas and/or functions that can benefit from Parasolid SMP being enabled.

Functional area	Notes
Validity checking	Implemented at the face level.
Boolean operations	Implemented for face-face clashing portion of algorithm.
Wireframe rendering	Implemented at the body occurrence level.
Hidden line rendering	Implemented at the body level during initial processing and at an intersection level during line/line intersection.
Closest approach	Multiple points must be supplied.
Faceting	Implemented at the body level.
Mass properties	
Spline creation	
Isocline generation	

17.3 Calling Parasolid from multiple threads

Parasolid is **thread-safe**, in that it can be called simultaneously from multiple threads within an application safely. Threads that an application sends into Parasolid are called **application threads**. They are not the same thing as the internal **worker threads** that are created by Parasolid SMP. If Parasolid SMP is enabled, calling an SMP-enabled function in an application thread causes Parasolid to create and use worker threads.

The Parasolid APIs manage multiple application threads to ensure safety. Parasolid functions can behave as one or more of the following types:

- **Concurrent** functions can be executed simultaneously by different application threads.
- **Exclusive** functions prevent other application threads from entering Parasolid until it has completed.
- **Locally exclusive** functions act as concurrent functions when called from a locked partition, but behave exclusively if no partitions are locked. For more information, see Section 17.4, “Locking partitions to threads”.

Parasolid can be called from multiple application threads on all available platforms.

Enabling internal algorithms to benefit from SMP or allowing Parasolid APIs to be called concurrently is not a substitute for general performance optimisation of our algorithms. Our philosophy is to ensure that all algorithms, are written in a performance optimal manner with minimum redundancy

This is an area of active development; Parasolid support for multiprocessors is continually evolving to meet the exacting requirements of a sophisticated and diverse customer-base.

17.4 Locking partitions to threads

When calling Parasolid from multiple threads, it is possible to lock one or more partitions to a given thread. With this mechanism, the thread can only operate on the data in the partition(s) that it controls and no other thread can operate on the data in the locked partitions. The advantage of this is that it allows an application to execute exclusive API calls in parallel so long as each one is called on data from locked partitions.

Index *In*

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y