Enhancements in Blending Algorithms

This article describes a rounding operation for a 3D CAD boundary representation (B-Rep) solid model. Complex combinations of convex and concave edges are handled predictably and reliably. At vertices the surfaces are smoothly connected by one or more surface patches. An algorithm for the creation of blending surfaces and their integration into the model is outlined. The sequence of topological modifications applied to the solid model is illustrated by examples including some special case handling.

by Stefan Freitag and Karsten Opitz

Apart from the basic Boolean operations, a modern solid modeling CAD system needs to provide easy-to-use facilities for local modifications of the primary model. One of the most important examples is the *blending* or rounding of edges, in which a sharp edge of the model is replaced by a surface that smoothly joins the two adjacent faces (see Fig. 1).

Blending surfaces serve several purposes in mechanical designs, including dissipating stress concentrations and enhancing fluid flow properties. In addition, some machining processes do not permit the manufacture of sharp edges. Smooth transitions between surfaces are also often required for aesthetic reasons. Besides functional requirements, edge blending is conceptually quite a simple operation, which makes it very popular among designers using CAD systems.

A common characteristic of almost all applications is that the smoothness of the blend is more important than its exact shape. For the user this means that it should be possible to create a blend by specifying only a few parameters. It is then the system's task to fill the remaining degrees of freedom in a meaningful manner.

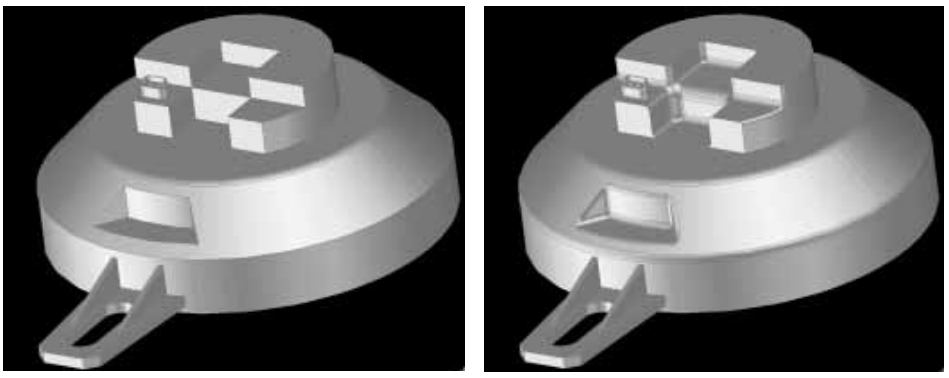
From an algorithmic point of view, blending one or more edges of a solid model simultaneously falls into two subtasks. The first is to create a surface that provides the transition between the adjacent surfaces defining the edge. Secondly, the surfaces need to be trimmed properly and integrated into the body such that a valid solid model is maintained. While the first step is a purely geometric problem, the second one involves both geometric and topological operations.

The blending module in HP PE/SolidDesigner was designed with the goal of allowing blending of a wide variety of complex edge combinations in a robust manner. This is accomplished through the use of freeform geometry as blending surfaces, along with quite involved geometric and topological considerations in several phases of the algorithm.

The lack of freeform surfaces was the primary reason for most of the restrictions concerning edge blending in HP PE/SolidDesigner's predecessor, the HP PE/ME30 3D modeling system. HP PE/ME30's kernel, the Romulus geometric modeler, does in fact provide more complex surfaces,[1] but these enhanced blends were never implemented in the product.

The current capabilities of HP PE/SolidDesigner's blending algorithm go far beyond HP PE/ME30 with respect to the topological situations that can be handled reliably. Moreover, the architecture of the algorithm allows the inclusion of future enhancements in a consistent manner.

It is the aim of this paper to illustrate the basic blending algorithm and to provide the reader with examples that demonstrate the complexity of the geometric and topological problems that must be solved to integrate one or more blend surfaces into a solid model. More information on this subject



(a) (b)

Fig. 1. (a) A solid model with sharp edges. (b) Edges rounded by blending.

can also be found in Woodwark[2] and the excellent survey of Vida.[3]

HP PE/SolidDesigner's underlying philosophy allows flexible modifications of the solid model in every stage of the modeling process. In the context of edge blending this means that it should always be possible to remove or modify an existing blend surface without regard to how it was created.

In the next section, the second section of this article, we introduce some terminology commonly used in solid modeling, in particular in the blending context. The third section describes the use model of edge blending in HP PE/SolidDesigner. An overview of the algorithm is given in the fourth section, followed by a more detailed discussion of its major steps. Finally, in the last section, we discuss some performance and stability issues.

Blending Module of the HP PE/SolidDesigner Kernel

Currently, the blending operation in the HP PE/SolidDesigner kernel implements what is commonly known as the *rolling ball blend*. This type of blend can easily be visualized as a ball moving along the edge and touching the adjacent surfaces (the *primary* surfaces) simultaneously. The touching loci are curves that define the boundaries of the blend surface. Depending on whether the radius of the ball is constant or varies while it is moving, we speak of *constant-radius* or *variable-radius* blends.

The geometry module of HP PE/SolidDesigner's kernel supports a number of different surface types (Fig. 2). These include the natural quadrics (planes, spheres, cylinders, and cones), toruses, and NURBS (nonuniform rational B-spline) freeform surfaces. All of the surface types are represented parametrically. The object-oriented design of the kernel allows the use of generic algorithms for general surfaces as well as special-case solutions for particular surface types.

Most algorithms such as surface/surface intersections or silhouette calculations behave considerably more stably and perform more efficiently when dealing with nonfreeform or *analytic* surfaces. Consequently, the blend algorithm tries to employ analytic surfaces whenever possible. This necessitates several case distinctions during the process of blend creation, which will be pointed out later.

Depending on the local geometry, that is, the convexity of the edge, blending an edge may involve adding or removing material. These operations are sometimes distinguished as *filleting* or *rounding*, respectively. In this article we will refer to both cases as blends.

If several edges to be blended meet at a common vertex, the blending surfaces should be joined in a smooth manner. We call these transitions *vertex regions* because they replace a vertex by a set of surfaces. In some special cases, a vertex region can be defined by a single analytic surface like a sphere or a torus. In general, however, they are defined by up to six tangentially connected B-spline freeform surfaces.

HP PE/SolidDesigner belongs to the class of B-Rep (boundary representation) modelers, in which the solid model is represented internally as a set of vertices, edges, and faces. In addition, the representation contains information about how these entities are related to each other—that is, the *topology* of the model. B-Rep modelers usually employ a restricted set of operations to perform topological manipulations of the model. The application of these *Euler operators* ensures the topological integrity of the model.

Integrating one or more blend faces into a solid involves quite a number of topological modifications and different Euler operators. We will not discuss the underlying concepts in detail here, but refer the reader to the standard sources.[4,5,6] For our purposes, it suffices to know that the blend algorithm employs these basic operators (for example, ADEV, ADED, KEV, KE) to create the new topological representation of the blended body.

The blend module also takes advantage of basic functionality provided by the geometry module of HP PE/SolidDesigner's kernel. Examples are closest-point calculations with respect to a curve or a surface. We call these operations *relaxing* a point on a curve or surface. This applies to curves or surfaces of any type. For instance, it is often necessary to relax an arbitrary point on the intersection curve of two surfaces. Since these operations are part of the kernel's generic functionality, we will not go into the details of their implementation.

Using the Blend Command

Like all of HP PE/SolidDesigner's commands, the user interface for the blend command is designed to be easy to use and require as little input as possible from the user. This is greatly facilitated by some general mechanisms used throughout HP PE/SolidDesigner's user interface such as the selection methods and the labeling feedback.

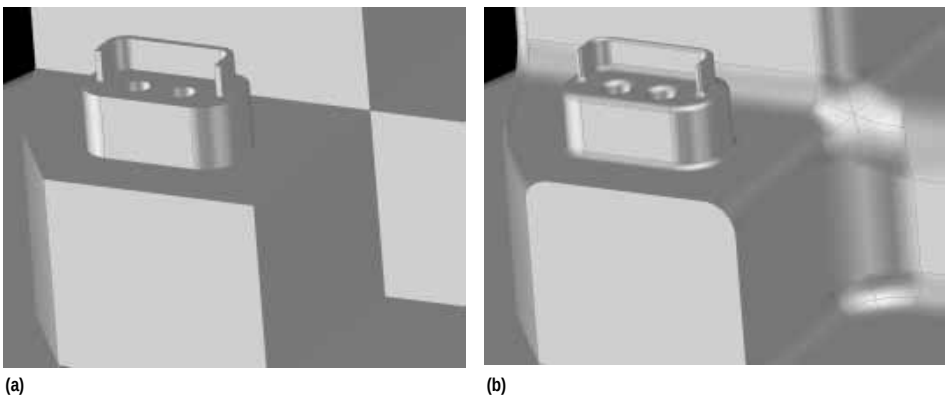


Fig. 2. Detail from Fig.1 showing different types of surfaces employed by the blending algorithm.

The blend command distinguishes two modes: the *definition mode* and the *preview mode*. In definition mode, single or multiple edges can be selected and assigned a radius (of the rolling ball). Variable-radius blends are specified by start and end radii to be assigned to the end vertices of the edge. Since the choice of the start and end vertices is arbitrary, the vertices of the currently selected edge are marked with labels. The radius of the rolling ball varies linearly between the two end vertices of the edge.

An important feature of the blend command is its ability to handle both types of blends simultaneously. This gives the user the ability to specify an arbitrary combination of constant and variable radius blends, each with possibly different radii, in a single blend session.

The blend command uses straightforward radius defaulting. For example, the constant radius of the active edge carries over to all subsequently selected edges unless the user chooses a new radius explicitly.

While processing the selected edges, the algorithm decides about the inclusion of a vertex region to provide a smooth transition between the blend surfaces. A vertex region will be created if all edges adjacent to a particular vertex are to be blended in the same session. In other words, a vertex region can easily be suppressed by blending adjacent edges one after another.

In preview mode, the blend faces are shown using a preview color. Modification of the radius or the edge information is not possible in this mode. However, upon returning to the definition mode, the user can specify further edges to be blended, modify the blend radius assigned to an edge, or remove an edge from the list.

There are two ways to terminate every command in HP PE/SolidDesigner. Canceling the blend command causes the blends to be discarded, while completing it makes the blends "real."

For convenience, the blend menu contains a small number of options:

- The part checker usually run on the blended part can be switched off to provide a faster, although possibly invalid result.
- The labels attached to edges and faces, which might be annoying if a large number of edges are selected, can be turned off.
- A *chain* option allows the user to select all edges connected tangentially to a given edge by a single pick.

Because of the complexity of the operation, blending one or multiple edges sometimes fails. While some problems are easily detected, others are caused by topological or geometrical restrictions rooted at a relatively low level. A typical example for the first kind of problem is the case where the blend radius is chosen too large. In any case, a failure is reported to the user by displaying an error message and highlighting the edge that is causing the problem.

How the Blending Algorithm Works

As noted above, the rolling ball blend provides us with a very intuitive way to define a blend surface. While moving along the edge, the ball sweeps out a certain volume. The blend surface is simply a part of the surface bounding this volume. In mathematics, surfaces that are swept out by families of moving spheres are called *canal surfaces*.[7] The cylinder and the torus are the most obvious examples.

A number of blending problems can be handled by inserting surfaces of these types. We will refer to these cases as *analytic blends*. In other than the simple cases, however, the explicit representation of a canal surface takes on quite a complicated form. Therefore, an approximation of the ideal blending surfaces by freeform blends is constructed. In particular, we use C^1-continuous B-spline surfaces.

The general algorithm is divided into a number of smaller modules. Each of these modules typically scans over all edges to be blended and performs a certain task. However, care is taken that the result is symmetric, that is, it does not depend on the order in which the edges are operated on.

The task of the first module is to filter out all cases where an analytic solution exists and flag the corresponding edges accordingly. In the second step, the touching curves of the ball with the primary surfaces are calculated. While this is straightforward for analytic blends, the boundaries of freeform blends must be computed numerically. This is accomplished by a *marching* algorithm.

Having calculated the boundaries of the blend surface, we determine their intersection points with other edges. It is often necessary to remove edges from the model to find useful intersection points. This is the first step that possibly involves topological modifications of the original body.

Other major changes to the model are done in the next two modules, which represent the blend face topologically. The first module performs the *zipping* of the original edge, that is, it replaces this edge by two new ones connected to the same end vertices. Secondly, the appropriate topology at the end vertices is inserted.

From a topological point of view, the model containing the primary blends is now complete. However, several topological entities are still without geometry. The surfaces corresponding to the blend faces, for instance, are not yet defined. These are computed in the next module based on the already available boundary data.

Furthermore, the surfaces need to be trimmed at the end vertices of the original edges. The trimming curves of the surface are, in general, computed by intersecting them with adjacent surfaces. However, it might also be necessary to intersect two adjacent blend surfaces created in the same session. The intersection curves are then "hung" under the corresponding edges.

Finally, the last major module performs the inclusion of vertex regions, both topologically and geometrically. These steps will be described in more detail later.

Analytic or Freeform Blends

It is not difficult to list all cases where a cylindrical or toroidal surface fits as a blend between the two primary surfaces. The simplest case is the one in which two intersecting planes blended by a cylindrical surface. A torus can be used when blending the edge between a conical and a planar surface as shown in Fig. 3. In a first pass over all involved edges, the algorithm tries to match one of the cases where

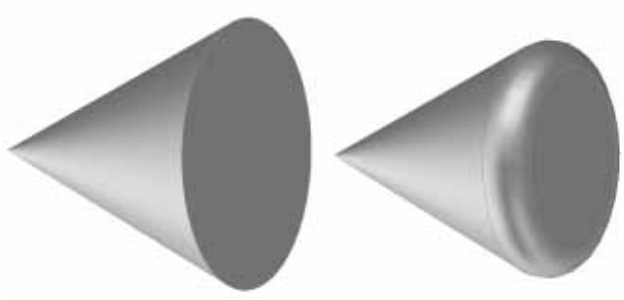


Fig. 3. A torus provides a smooth blending of the edge between a conical surface and a planar surface.

such a solution exists. The corresponding edges are then flagged as analytic.

The decision about when to employ analytic or freeform blends, however, is also dependent on other, more global factors. For example, suppose that three cylindrical blends with different radii meet at a common vertex (Fig. 4). This necessitates the inclusion of a freeform vertex region. Depending on the numerical tolerance used in the system, this might lead to very expensive B-spline surfaces in terms of data generated (the B-spline boundaries of the vertex region must lie—within some tolerance—on the adjacent cylinders). Therefore, it is often necessary to use freeform blends rather than analytic ones at a subset of the edges for the benefit of reducing overall data size. The corresponding checks are done in a second pass over the edges.

As a side-effect of switching from an analytic to a freeform blend for a particular edge, other edges adjacent to this one might be affected. This is also taken care of in the second pass.

The results of these operations are flags attached to all involved edges and their end vertices which provide information to all following modules about the types of surfaces to be used.

Blend Boundary Creation

The task of the second module is to compute the blend boundaries and tangency information along these curves. This information will be used later for the construction of the blend surfaces. The calculation of the boundaries for cylindrical and torodial blends is a straightforward exercise in analytic geometry and will not be described here. More involved and computationally more expensive is the general case, which will be the main topic of this section.

A major advantage of the rolling-ball blend is that its definition can be put into mathematical terms quite precisely. Suppose a ball with radius r moves along the edge between the primary surfaces. The curves where the ball touches the surfaces will be the boundaries of the blend surface to be inserted. The center of the ball moves along a third curve, the *spine* of the canal surface. If the radius of the ball changes while rolling, the curves touching the surfaces will define a variable-radius blend surface. In HP PE/SolidDesigner a general B-spline curve is used to define the radius function.

The spine lies entirely on a surface with constant distance r from the original surface. This is called the *offset surface*. This applies to both primary surfaces. Therefore, we can calculate the spine as the intersection curve of the two offset surfaces (Fig. 5).

Computing surface/surface intersections is a ubiquitous problem in solid modeling and many algorithms have been devised for its solution. Very popular are the marching algorithms, which trace out the intersection curve starting from a given point in its interior. In our blending algorithm, we get such a starting point by taking the midpoint of the original edge and relaxing it onto the spine. The entire curve is then computed by marching the intersection of the two surfaces in both directions. The marching stops when the curve leaves a certain 3D box provided by the calling routine. The boxes are chosen such that the resulting blend surfaces are large enough to fit into the model.

The particular strategy we employ for the marching is to reformulate the problem as one of solving a differential equation in several unknowns. The solution is then computed by a modified Euler method.

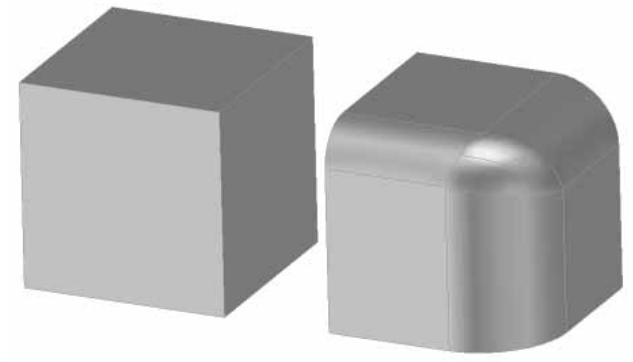


Fig. 4. Three cylindrical blends with different radii connected by a freeform vertex region.

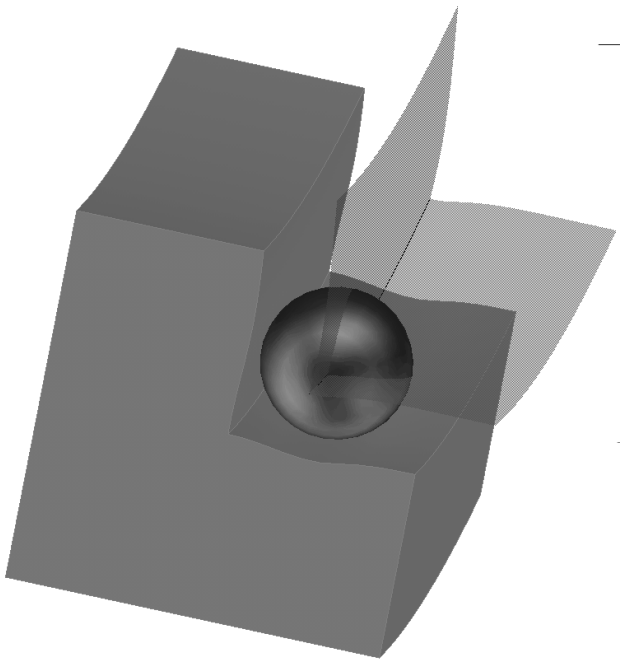


Fig. 5. The center of the rolling ball moves on the intersection curve between the two offset surfaces.

A common problem in marching algorithms is the choice of an appropriate step size. Choosing the step size too big might lead the algorithm astray. On the other hand, very small steps usually guarantee convergence of the method but might generate too much data. Therefore, we use an adaptive technique based upon the curvature of the intersection curve: a small curvature indicates that the intersection curve behaves almost like a straight line. This means that we can proceed with a large step. On the other hand, if the curve bends sharply, that is, its curvature is large, we use very small steps to capture all of its turns.

The result of these computations is a set of isolated points lying exactly on both offset surfaces and thus on the spine. Conceptually, the corresponding points on the blend boundaries can be determined by projecting these points onto the original surfaces (Fig. 6). In fact, for parametric surfaces this operation is trivial because the offset surface inherits its parameterization from the underlying surface. This means that we simply have to evaluate the primary surfaces at the parameter values of the points on the spine.

The blend boundaries are now created by constructing cubic Hermitian segments between the given points. However, we still have to check whether the entire segment lies on the surface, within a given tolerance. In cases where it doesn't, we use a fast bisection method for "pulling" the curve segment onto the surface.

While the intersection curve—and thus the blend boundaries—are traced out, we also collect tangential information along the boundaries. This information is used in the surface creation step to construct smooth transitions between the primary surfaces and the blend surface. The same bisection and representation techniques as for the boundary curves are used for these *cross-tangent curves*.

Before we conclude this section, we still have to address the question of singularities, which are critical for every marching algorithm. In our context, we have to deal with two types of singularities: those of the surfaces to be marched and those of their intersection.

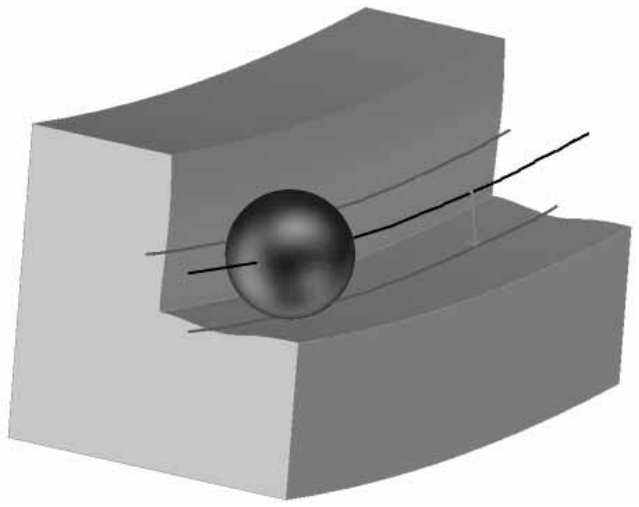


Fig. 6. The blend boundaries (red) are created by mapping the spine (black) onto the primary surfaces.

The first problem is illustrated in Fig. 7. While a small offset leads to well-behaved curves, larger distances result in offset curves with cusps or self-intersections. Analogously, we might have degenerate offsets of the primary surfaces if the distance (radius of the blend) is chosen too large. For too large a radius, a rolling ball blend is not possible. When such a situation is detected the marching stops, the entire blend algorithm stops, and the user is advised to try the operation again with a smaller radius.

The second type of singularity occurs if the primary surfaces and consequently their offsets possess a common tangent plane (Fig. 8). These *tangential intersections* typically create the biggest problems for marching algorithms. Loosely speaking, it is very difficult to find where to go at these points. However, a rolling ball blend is still well-defined. The touching curves of the ball are identical with the original edge, and the blend surface degenerates to one with zero width. HP/PE SolidDesigner's kernel enforces the rule that these extraordinary points may only occur at the endpoints of an edge. This considerably eases the task for the blending algorithm. It is quite simple to check whether the intersection curve degenerates at its endpoints. This information is provided to the routine that performs the marching. Since the algorithm starts at the midpoint of the intersection curve, the occurrence of a singular point of this type indicates that we have reached one of the endpoints of the edge.

In a final step, the segments of the boundaries and the cross-tangent curves are merged into C^1-continuous B-splines. The overall result of this module consists of four C^1-continuous curves with a common parameterization describing the boundary curves and tangency information of the blend surface.

Trimming the Blend Boundaries

After creating the blend boundaries we need to integrate the boundaries into the body. Most important, we have to find the position where the boundaries are to be trimmed. Fig. 9 shows a particularly simple example.

The six points shown in blue can be calculated by intersecting the blend boundaries with the adjacent edges at the end

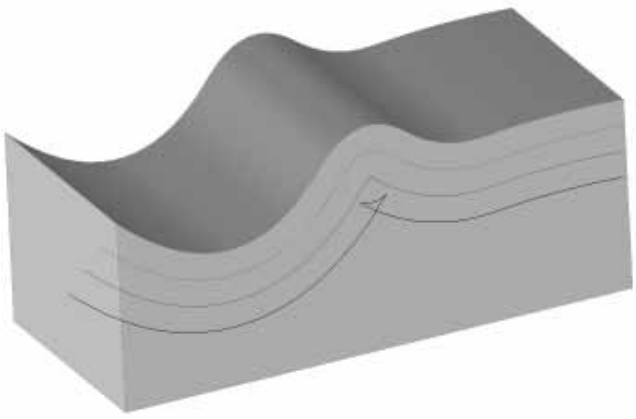


Fig. 7. When the blend radius is chosen too big, the blend boundary will have a cusp (red curve) or even be self-intersecting (black curve).

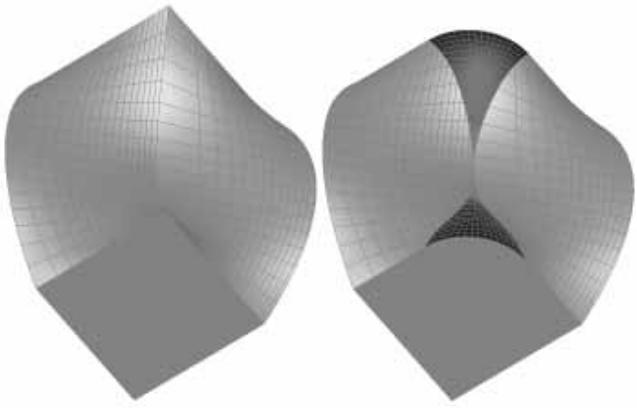


Fig. 8. If the primary surfaces have the same normal along the edge, the blend surface (blue) degenerates.

vertices. However, usually the set of edges to be blended with possibly different radii is not limited to one edge but may contain several edges or even all of them. This means that while the boundaries of a given blend face are being trimmed they must be intersected with other blend boundaries created in the same session (red points).

Intersecting a blend boundary with an existing edge of the solid model may have three results:

- One intersection point found. This is the general case.
- No intersection found. The edge is too short to be intersected by the blend boundary. In this case the edge will be removed from the model. The edge newly attached to the vertex will now be intersected by the blend boundary. Repeating this procedure guarantees the existence of at least one intersection point.
- Multiple intersection points found. Such a situation might occur, for instance, if the adjacent edge is part of a B-spline curve "wiggling" around the blend boundary. In this case, the most valuable intersection point has to be chosen. A valuable point in this context is the one that produces the most predictable and expected result.

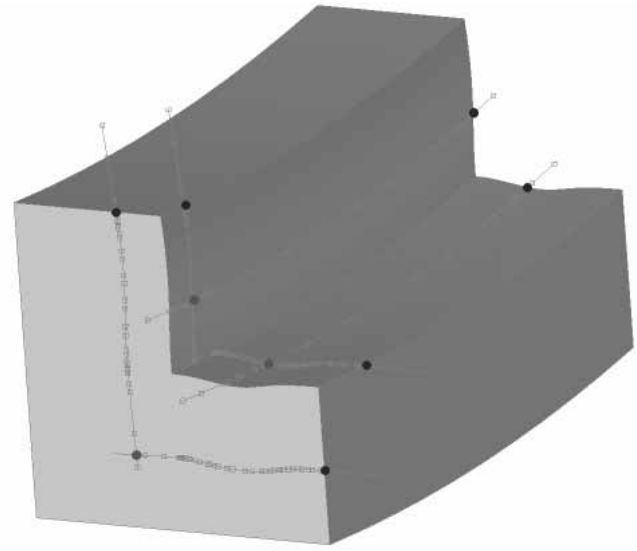


Fig. 9. The blend boundaries are trimmed at points where they intersect adjacent edges (blue) or another blend boundary (red).

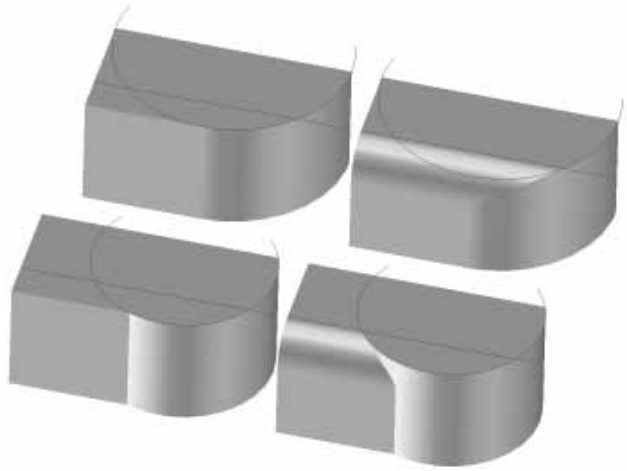


Fig. 10. Selecting the correct intersection point between a blend boundary and an adjacent edge also depends on the local surrounding geometry.

In fact, very often there are several possible solutions and all of them result in a valid solid model. Several different criteria are used to select the best intersection point. Fig. 10 shows two examples. The remaining intersection points are ignored.

Creating the Topology of the Blend Face

Having computed the trimming points of the blend boundaries, we build up the topology of the blend face. The first step is similar to opening a zipper: the original edge of the body is replaced by two new ones connected to the same vertices. The new face is then extended at its end vertices. More precisely, four new edges—two at each end—are added. In addition, the adjacent edges are split at the four trimming points (Fig. 11).

Blend Surface Creation

Now the face is ready for the inclusion of the blend surface. There are two possibilities. In the first case, analytic surfaces are inserted based on the decision made in the first module. Possible surface types are cylinders, cones, and toruses only (Fig. 12). In all other cases a freeform surface is created. We use C^1-continuous B-spline surfaces. This surface is defined by the blend boundaries created by the marching algorithm, the tangency information along these boundaries represented by cross tangent curves, and the fact that the blend surface should have circular cross sections. Using this knowledge the surface can be created very easily. The circular cross section is approximated by a single cubic B-spline segment. Although not precise, this approximation is sufficiently good for practical purposes. In fact, given the input

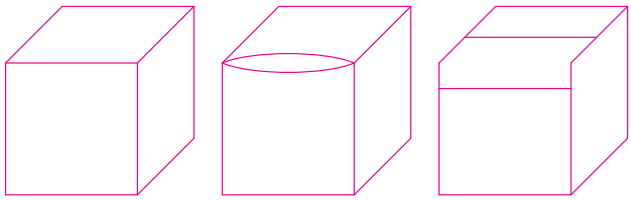


Fig. 11. Creating the topology of a blend face.

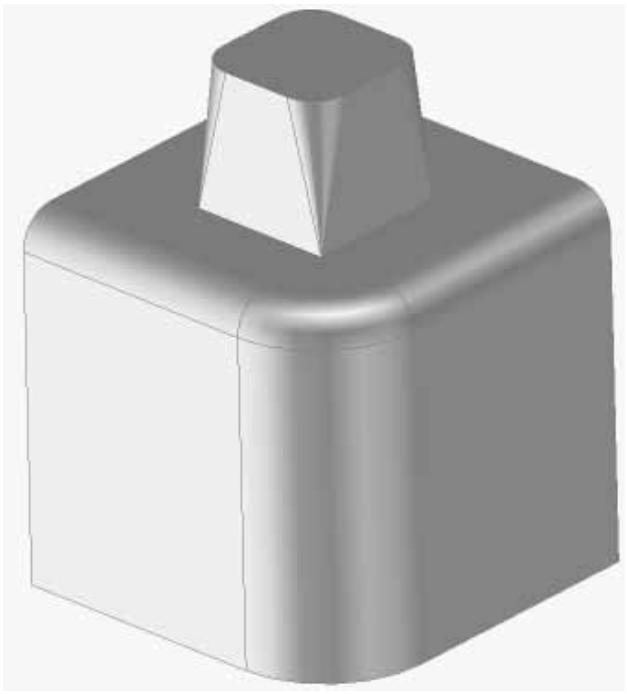


Fig. 12. A model containing only analytic blend surfaces: cylinder, cones, and toruses.

data for the cross section—boundary points and tangent directions—we use an optimal approximation based on a method described by Dokken.[8] The boundary curve information and parameterization transfer directly to the surface (Fig. 13).

Trimming the Blend Surfaces

The last step in integrating the blend surface into the solid model is to trim it at the ends. The goal is to keep the trim area as simple as possible.

Unfortunately, the authors of many edge-blend algorithms assume that they are dealing with a trimmed-face surface model and they offer no suggestion about what to do at the ends of the edge to be blended. The topological and geometrical issues are quite complex, especially when multiple edges meet at a common vertex.

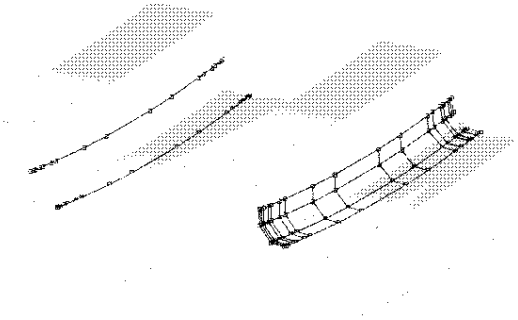


Fig. 13. Creating the geometry of a freeform blend surface: the control polygon of the blend boundaries (left) and the resulting blend surface (right).

The simplest type of termination issue arises when there is only one edge to be blended. Both boundaries must be joined at the ends of the blend face. The easiest way to do this is to intersect the blend face with all edges and faces connected to its end (Fig.14). Intersecting the blend face with these edges creates intersection points which are to be connected to form the boundary of the blend face.

The intersection points are calculated by curve/surface intersections between the blend surface and the curves of the edges at the end of the blend face. In general, a curve/surface intersection will result in multiple intersection points. In this case, the one chosen is the one closest to the vertex of the edge to be blended at this end.

If there is no intersection point of the blend face and an edge this edge is removed from the model using the Euler operator KEV. If this edge is the last one of its face, the face is removed using the Euler operator KBFV. Removing an edge means disconnecting it from its vertices and filling the gap by connecting other edges to these vertices. The newly connected edges have to be intersected with the blend face, too. However, if one of these edges is also to be blended, an intersection between its blend boundaries and the blend face is calculated. The intersection points are then connected by intersection tracks of the blend surface and the adjacent ones. In general, the result of this surface/surface intersection calculation is a set of intersection tracks. Tracks that do not contain the intersection points described above are filtered because they are not needed. The remaining tracks are sorted by the distance between two intersection points. The shortest arc is the one chosen because it minimizes the trim area at this end.

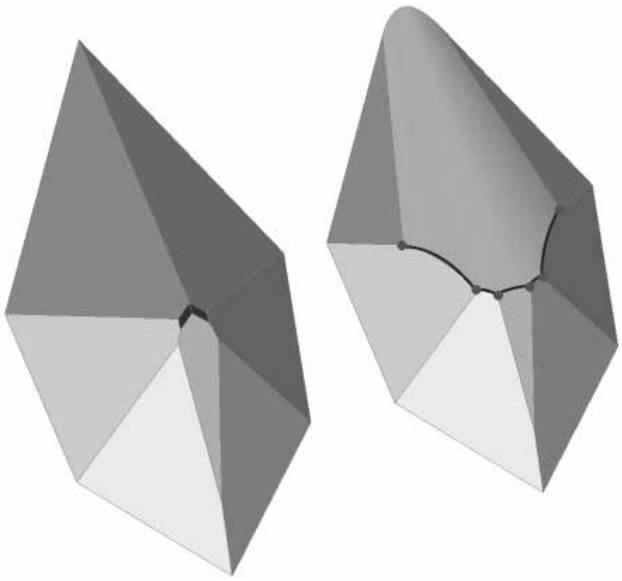


Fig. 14. Trimming a blend surface involves a number of curve/surface intersections (red points) and surface/surface intersections (blue curves). Note how the faces marked dark red are "eaten up" by the blend face.

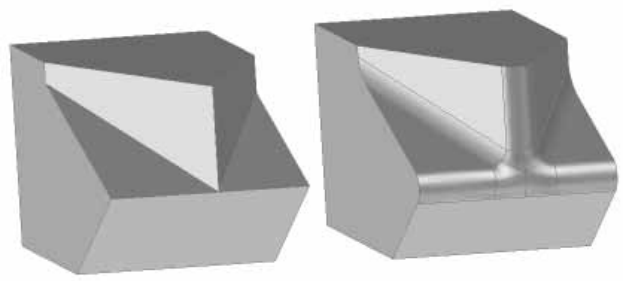


Fig. 15. When more than two edges to be blended meet at a common vertex, a vertex region is inserted to connect the blends smoothly.

Vertex Regions

A totally different situation occurs when more than two edges meet at a common vertex. In this case a set of additional faces and surfaces must be created to build a transition patch that smoothly connects all of the blend faces meeting there. This set of faces is called a vertex region (Fig. 15).

In some special cases a vertex region has only one face, which is an analytic surface (sphere or torus). In general, however, a vertex region will contain three or more faces. In HP PE/SolidDesigner the number of faces in a vertex region is currently limited to six.

Topology of Freeform Vertex Regions. At a vertex where five edges to be blended meet each other, the topology shown in Fig. 16a arises after extending the blend faces as described above. The blending algorithm transforms this topological situation by integrating five faces, each having four edges, as shown in Fig. 16b. Transforming the topology requires the use of the Euler operators KEV, ADEV, and ADED to kill an edge, add an edge, and add a whole face. Fig. 17 shows the sequence of Euler operators.

Topology of Analytic Vertex Regions. When a sphere or torus fits a vertex region the topology is changed in another way. Instead of the "star" where the blend faces meet, a single face will be created using KEV and ADED, as shown in Fig. 18. Fig. 19 illustrates the algorithm, showing the transformation step by step.

Geometry of Freeform Vertex Regions. After creating the topology of a vertex region, the corresponding geometry must be constructed and integrated. To provide a smooth transition,

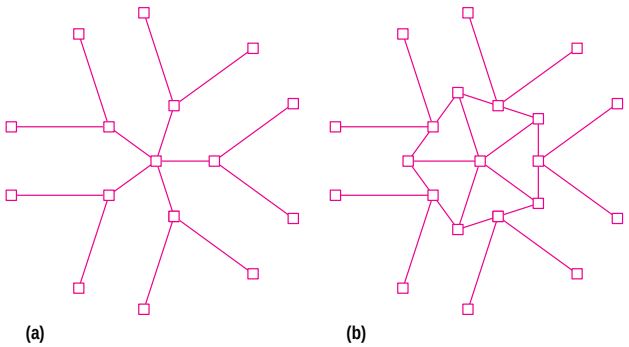


Fig. 16. (a) Topology of a vertex region where five faces meet after extending the edges. (b) Topology created for the representation of the vertex region.

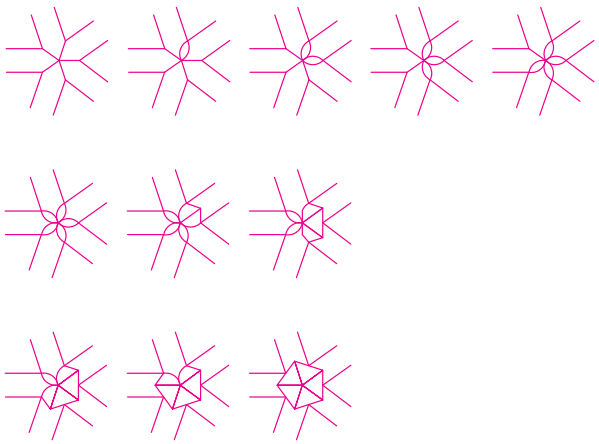


Fig. 17. Sequence of Euler operators used to transform the topology of Fig. 16a to the one of Fig. 16b .

the surfaces must satisfy two constraints. First, their boundaries must match the ones of the adjacent surfaces. Secondly, the vertex regions and the blend surfaces should possess the same tangent planes along their common boundaries. The construction of vertex regions satisfying those constraints is a classical problem in geometric modeling.[9] Among the many solutions, we mention the one proposed by Charrot and Gregory.[10] They fill a vertex region by a procedurally defined surface, that is, a surface that does not have an analytic mathematical representation but rather is defined by a method of generating it. Since the geometry kernel of HP PE/SolidDesigner does not support this type of surface, we employ an algorithm that generates a set of four-sided B-spline surfaces. More precisely, for filling an n-sided hole, we use n B-spline surfaces of polynomial degree 6 in both parameter directions.

Geometry of Analytic Vertex Regions. From the geometrical point of view analytic vertex regions are quite easy to compute because only one surface is needed and the surface type will be either a sphere or a torus.

Transition Curves. When large radii are combined with very small radii, a vertex region can look very strange, deformed, or even self-intersecting, like the left solid in Fig. 20. In such cases, instead of a three-sided vertex region, a four-sided one is used, giving a result like the right solid in Fig. 20. In general, an (n+1)-sided region is used instead of an n-sided region. This is done by introducing a transition curve between two boundaries sharing the same face. The transition curve is used whenever an intersection of two boundaries is

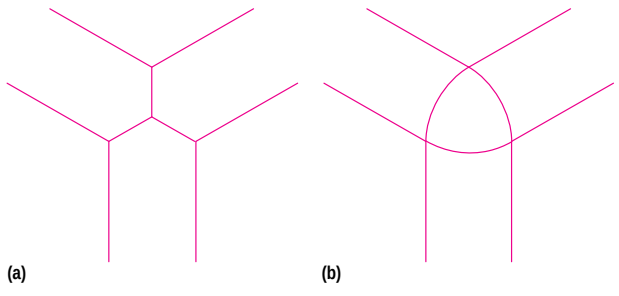


Fig. 18. When part of a sphere fits as a vertex region, a single face is created.

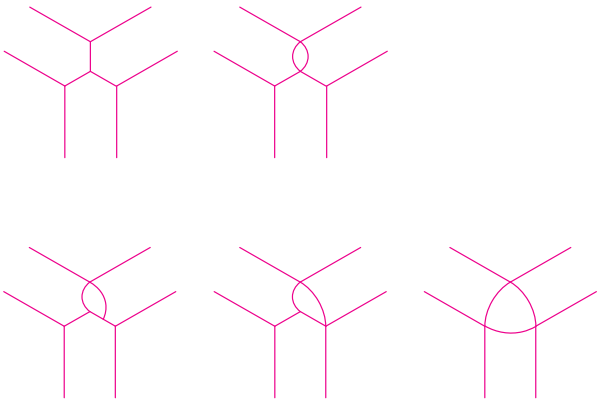


Fig. 19. Sequence of Euler operators used to create the face of Fig. 18.

"behind" the direct connection of its neighboring intersection points.

Geometrically, the transition curve is a B-spline curve defined at its endpoints by tangency conditions to both boundaries and in between by a tangency condition to the corresponding face. This curve is created using an adaptive curvature-controlled bisection algorithm similar to the one used to create the blend boundaries. The endpoints of the transition curves are constructed such that the cross section of the resulting blend surface is an isoparametric of this surface. In certain cases it is also necessary to insert a transition curve to smoothly connect two nonintersecting adjacent blend boundaries. Fig. 21 shows an example.

Special Cases

A reliable blending algorithm must be able to handle various topological and geometrical special cases predictably. Four major special cases are tangential intersections, apex creation, a singularity at the end of a blend surface, and closed curves.

Tangential Intersections. Real-life solid models often contain edges connected tangentially at a vertex to another edge. Blending these edges will result in very complex and time-consuming surface/surface intersections in the process of trimming the blend faces at the common vertex, especially when their radii differ only slightly.

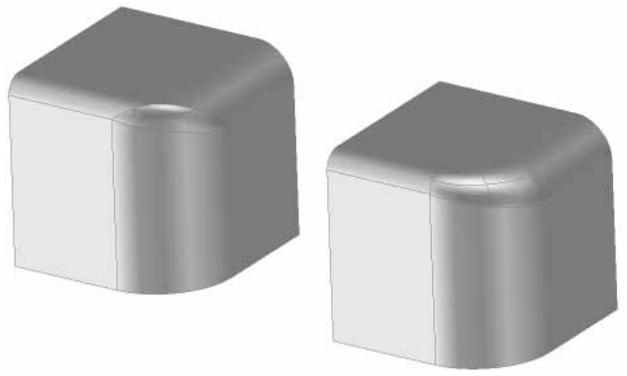


Fig. 20. When the vertex region would be too badly deformed, an additional transition curve is inserted to provide a smoother transition.

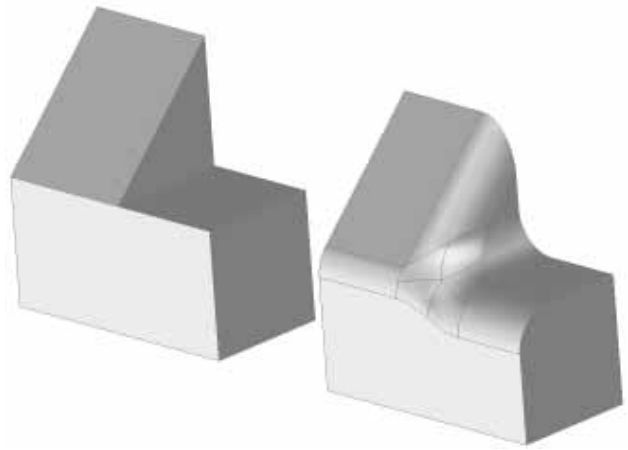


Fig. 21. A transition curve (lower edge of the vertex region) is also inserted when two adjacent blend boundaries around a vertex region don't intersect.

If two boundaries are tangential to others, the intersection point calculation is numerically very unreliable and expensive. In addition, both blend surfaces share a common region of partial coincidence, so the intersection track calculation is even more expensive than the intersection point calculation.

To avoid these problems, two edges to be blended are handled in a totally different way. No curve/surface or surface/surface intersections need to be calculated. Rather, an additional face is created that smoothly connects the two surfaces (Fig. 22).

Apex Creation. If an edge to be blended is concave, material is added to the solid model. This means that other edges become longer and faces become larger, and sometimes a singular point moves into a face. HP PE/SolidDesigner requires a topological entity, a vertex, right at the apex in this case. Therefore, after creating the blend face the required vertex is added (Fig. 23).

Singularity at the End of a Blend Surface. Sometimes at an endpoint of the edge to be blended the surface normals of the adjacent surfaces are equal—for example, two cylinders with the same radius intersected orthogonally (Fig. 24). In this case both boundaries of the blend surface meet at a common point where both surface normals are equal. There is

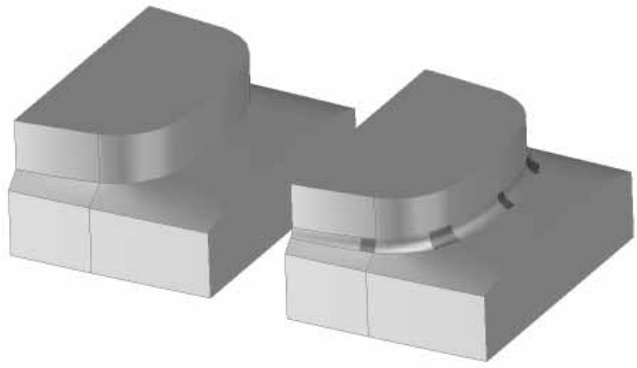


Fig. 22. Additional faces (red) are inserted where adjacent blends are tangentially connected.

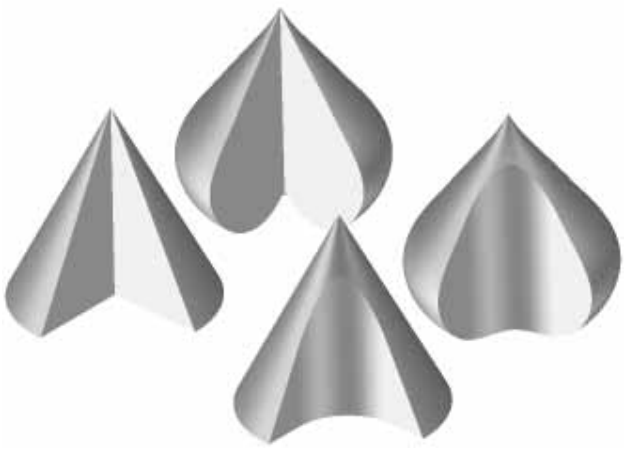


Fig. 23. When material is added by blending an edge, an apex might move from the boundary to the interior of a face.

no need to execute the trimming part of the blend algorithm because the solid is already closed at that end.

However, from the geometrical point of view, the blend surface is degenerate. One side or in this case both sides of the blend surface are degenerate isoparametric boundaries. This means that evaluating any parameter space point at this surface boundary results in the same object space point. This object space point is the position where both blend boundaries meet and the adjacent surfaces have the same surface normal.

Data Size and Performance versus Accuracy

The size of the data structures that represent freeform geometry mainly depends on the number of control points defining the curve or surface. In practice, curves can have hundreds of control points and surfaces many more. As an example, let's consider a medium-size surface with 500 control points with three coordinates each. In double-precision format, such a surface requires $3 \times 8 \times 500$ bytes or approximately 12K bytes of memory. In fact, a real-life model may contain many freeform surfaces. It is therefore quite important to reduce both the number of such surfaces and the number of control points used to represent them.

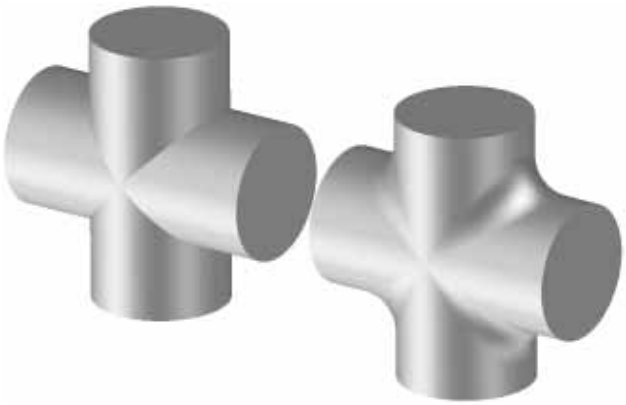


Fig. 24. Another example of two adjacent surfaces that have the same normal at an endpoint of an edge to be blended. The trimming part of the algorithm is not needed.

The size of a freeform blend surface is basically determined by the complexity of its boundaries. Boundaries with n control points lead to surfaces with 4n control points. Consequently, it is critical to generate approximations of the "true" blend boundaries with a minimal amount of data. On the other hand, the creation of the blend boundaries is one of the major factors determining the algorithm's overall performance. Finding an acceptable compromise between the conflicting requirements of speed and quality of the solution is an important design decision in the algorithm.

The same applies to the surfaces used for filling the vertex regions. The size of such a surface is quadratically dependent on the size of its boundary curves. Let's again consider an example. Assume that the boundary curves of a three-sided vertex region are general intersection curves between the primary blend surfaces and planes. It is not uncommon for approximations of those curves to contain 50 control points (HP PE/SolidDesigner works with an accuracy of up to 10^{-6}). This would lead to a vertex region of $3 \times 25 \times 25 = 1875$ control points (three surface patches of 25×25 control points each), requiring $3 \times 8 \times 1875$ bytes or approximately 44K bytes of data. Clearly, this is unacceptable for nontrivial models.

There are several possibilities for reducing the amount of data. The most critical factor is the approximation tolerance used in the system. For example, reducing the accuracy from 10^{-6} to 10^{-3} typically reduces the size of freeform data structures by a factor of ten. Not only are the geometric calculations speeded up considerably when using a lower accuracy but also the overall performance of the system is improved because of the reduced demand for memory management. HP PE/SolidDesigner offers the user the ability to select the accuracy in a range of 10^{-2} to 10^{-6}. This allows the user to choose between high-precision modeling and a faster but less precise approach.

Secondly, the handling of special cases can reduce the amount of data significantly. Let's again take a look at freeform vertex regions. If the primary blend surfaces are created such that the boundaries of the vertex regions are isoparametric curves of the primary blends (the procedure for doing this is beyond the scope of this article), the 50 control points can be reduced to 4. The vertex region will then contain $3 \times 7 \times 7 = 147$ control points (the additional control points along the boundaries—seven rather than four—are the result of the mathematical construction), for a total of approximately 3.5K bytes.

Another example is the trimming of a blend face. In this step a number of surface/surface intersections must be calculated. In general, an intersection of two surfaces will result in not only one curve, but several intersection points, curves, or even surfaces. However, in the blending context there is important knowledge about the blend surface and the face it intersects. At least one and in some cases two points on the intersection track are known from the preceding curve/surface intersections. Providing these points as "seeds" to the intersection routines increases both the speed and the reliability significantly. In addition, boxes in the parameter space of the surface are used to limit the calculation of intersection information to regions that are of interest.

From these examples we see that the good overall performance of the algorithm is mainly guaranteed by appropriate special case handling at critical points. In fact, a large portion of the code in the blending module was developed to deal with these situations.

Acknowledgments

The development and implementation of the blending algorithm in its early versions was largely conducted by our former colleagues Hermann Kellermann and Steve Hull. A part of the code was developed at SI/Sintef in Oslo, Norway.

References

1. A. Rockwood and J. Owen, "Blending Surfaces in Solid Modeling," in *Geometric Modeling: Algorithms and New Trends*, G. Farin, ed., SIAM, 1987, pp. 367-384.

2. J.R. Woodwark, "Blends in Geometric Modeling," in *The Mathematics of Surfaces II*, R.R. Martin, ed., Oxford University Press, 1987, pp. 255-297.

3. J. Vida, R.R. Martin, and T. Varady, "A survey of blending methods that use parametric surfaces," *Computer-Aided Design*, Vol. 5, no. 5, 1994, pp. 341-365.

4. B. Baumgart, *Geometric Modeling for Computer Vision*, PhD Thesis, Stanford University, 1974.

5. I. Braid, R.C. Hillyard, and I.A. Stroud, "Stepwise Construction of Polyhedra in Geometric Modeling," in *Mathematical Methods in Computer Graphics and Design*, K.W. Brodlie, ed., Academic Press, London, 1980, pp. 123-141.

6. M. Mantyla, *An Introduction to Solid Modeling*, Computer Science Press, Rockville, 1988.

7. W. Boehm and H. Prautzsch, *Geometric Concepts for Geometric Design*, AK Peters, Willesley, 1994.

8. T. Dokken, M. Daehlen, T. Lyche, and M. Morken, "Good approximation of circles by curvature continuous Bezier curves," *Computer-Aided Geometric Design*, Vol. 7, 1990, pp. 30-41.

9. J. Hoschek and D. Lasser, *Fundamentals of Computer-Aided Geometric Design*, AK Peters, Willesley, 1993.

10. J.A. Gregory, "N-sided surface patches," in *The Mathematics of Surfaces*, J.A. Gregory, ed., Clarendon Press, 1986, pp. 217-232.