The Direct Manipulation of Pasted Surfaces

by

Marryat Ma

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Technical Report CS-2000-15

Waterloo, Ontario, Canada, 2000

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

Surface pasting is a method for creating complex composite surfaces by adding local detail or feature surfaces to a base surface. Previous surface pasting editors enabled users to paste, translate, rotate, and resize feature surfaces, but none allowed users to directly manipulate a surface in the pasting hierarchy. In this thesis, I propose a multiresolution direct manipulation technique that allows the user to pick a point on a surface and move it to a new location and have the shape of the surface change appropriately. My method provides a more flexible modelling paradigm for pasted surfaces by allowing the user to pick both the new location of the selected point and the granularity of the change that is applied to the composite surface.

Acknowledgements

I would like to thank my supervisor, Stephen Mann, for his guidance throughout my research, his patience in reading the many drafts of this thesis, and his friendship. Special thanks to Richard Bartels for his contributions to this research. I would also like to thank both my readers, Richard Bartels and Donald Cowan, for taking the time to read my thesis and offering helpful suggestions. I am grateful to my friends for making my years at the University of Waterloo very enjoyable, and to my parents for their unconditional love and support.

This research was made possible by grants from NSERC and CITO.

Finally, I would like to thank my fiancé, Blair Conrad, for his endless love, patience, and support.

Trademarks

SGI and OpenGL are registered trademarks of sgi. Houdini is a registered trademark of Side Effects Software, Inc. All other products mentioned in this thesis are trademarks of their respective companies.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Hierarchical modelling is currently an active area for research. Many surfaces have varying levels of detail, and modelling techniques that explicitly represent these levels of detail are useful both in terms of reduced storage and in interactive modelling paradigms where users want to interact with their models at different levels of detail.

Tensor product B-spline surfaces are commonly used in the computer industry because they can be represented by little information and have attractive continuity properties. Local detail can be added to tensor product surfaces via knot insertion [Boe80, CLR80]. However, the knot insertion technique globally increases the complexity of a surface and so tensor product surfaces are poorly suited to multiresolution editing.

Hierarchical B-splines were developed by Forsey and Bartels [FB88] for adding areas of local detail to a tensor product B-spline surface. A parametrically aligned region of the surface is locally refined to increase its control point density. The control points in the refined region are displaced to create the local detail. Hierarchical B-splines allow multiresolution editing and maintain a high level of continuity, but the local details can not be translated, rotated, or resized.

Surface pasting, developed by Bartels and Forsey [BF91], is a generalization of hierarchical B-splines that allows the insertion of local detail to a tensor product surface without changing the structure of the underlying surface. In surface pasting, the area of local detail is represented

as a tensor product surface, called a *feature*. The feature is placed on an existing surface, called the *base*, to produce a composite surface. Additional features can be pasted hierarchically on the composite surface to create more complex composite surfaces. Using surface pasting, it is easy to maintain a library of features to paste onto a base surface. Once pasted, these features can be translated, rotated, and scaled on the base surface. Surface pasting has been integrated into Side Effects' *Houdini* software, where it has been successfully used in character animation.

Although surface pasting is a hierarchical modelling method, the user interfaces implemented for previous research have concentrated on positioning the features upon the base surfaces and adjusting—i.e., translating, rotating, and resizing—the features once they have been pasted. In this thesis, I propose a technique for the direct manipulation of pasted surfaces that allows the user to edit the surfaces at any resolution in the hierarchy.

Chapter 2

Background

In this chapter, I provide background information for hierarchical surface pasting and the direct manipulation of B-spline curves and surfaces. First, I give a brief introduction to B-spline curves and tensor product B-spline surfaces [Far97]. Then I describe the surface pasting process for tensor product surfaces. Next, I mention the user interfaces for early pasting editors and a limited direct manipulation scheme for hierarchical B-splines. Finally, I explain a method for the direct manipulation of B-spline curves and how I extended that method to tensor product surfaces.

2.1 B-spline Curves

A degree m B-spline curve is a piecewise polynomial curve whose polynomial segments typically meet with $\mathcal{C}^{m-1}$ continuity. The curve is defined by a set of control points P_i and a set of basis functions $\mathbf{N}_i^m$, for $i = 0, \ldots, M$. Each of the basis functions is a piecewise polynomial of degree m. The value of a B-spline curve at u is the sum of its control points weighted by their corresponding B-splines evaluated at u:

$$C(u) = \sum_{i=0}^{M} \mathbf{N}_i^m(u) P_i.$$

The B-splines are determined by their degree and by the *knot vector*, a non-decreasing sequence of domain values:

$$u_0 \leq u_1 \leq u_2 \leq \cdots \leq u_{M+m-2} \leq u_{M+m-1},$$

where $u_i < u_{i+m}$ for any i. The i^{th} B-spline is defined as follows:

$$\begin{aligned} \mathbf{N}_i^0(u) &= \begin{cases} 1 & \text{if } u_{i-1} \leq u < u_i \\ 0 & \text{otherwise} \end{cases} \\ \mathbf{N}_i^m(u) &= \frac{u - u_i}{u_{i+m} - u_i}\mathbf{N}_i^{m-1}(u) + \frac{u_{i+m+1} - u}{u_{i+m+1} - u_{i+1}}\mathbf{N}_{i+1}^{m-1}(u), \text{ if } m > 0. \end{aligned}$$

From the above equations, it can be seen that $\mathbf{N}_i^m$ is non-zero over $[u_i, u_{i+m})$.

Throughout my thesis, I will refer to a set of values, called the *Greville abscissae*, that are associated with a B-spline. The i^{th} Greville abscissa γ_i is the domain value where the i^{th} B-spline attains its maximum. The Greville abscissa can be calculated as

$$\gamma_i = \frac{u_i + u_{i+1} + \cdots + u_{i+m-1}}{m}.$$

2.2 Knot Multiplicity and Continuity

The *multiplicity* of a knot is defined to be the number of times it is repeated in a knot vector. If a knot has multiplicity equal to the degree of the B-spline, it is said to have *full multiplicity*. In this thesis, all B-splines are assumed to have knot vectors whose first and last knots have full multiplicity.

B-spline curves with full end knot multiplicity can be joined together with varying levels of *continuity* or smoothness. Two curves can be joined with $\mathcal{C}^0$ continuity by ensuring that the first control point of one curve and the last control point of the other curve are the same. A $\mathcal{C}^1$ join is obtained when:

- the curves meet with $\mathcal{C}^0$ continuity,
- the vector difference of the last two control points of one curve is parallel to the vector

difference of the first two control points of the second curve, and

- the ratio of the lengths of the vectors mentioned above is equal to the corresponding ratio of the parameterization length of the curve segments divided by the ratio of the curve degrees.

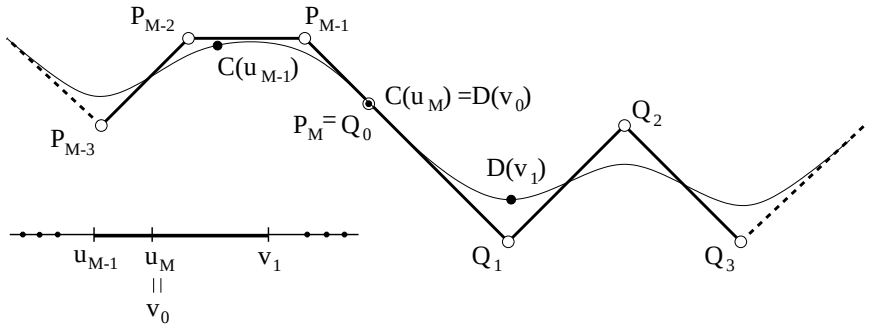


Figure 2.1: Example of $\mathcal{C}^1$ curve continuity

An example of two B-spline curves meeting with $\mathcal{C}^1$ continuity can be found in Figure 2.1. If C is a degree m B-spline curve and D is a degree n B-spline curve, then

$$\frac{m(P_M - P_{M-1})}{u_M - u_{M-1}} = \frac{n(Q_1 - Q_0)}{v_1 - v_0}. \tag{2.1}$$

2.3 Tensor Product Surfaces

B-spline curves can be extended to form a representation of surfaces. Tensor product B-spline surfaces are commonly used in modelling applications and form the basis of hierarchical surface pasting. A tensor product B-spline surface is a piecewise polynomial surface that is defined by a grid of control points $P_{i,j}$ and a set of basis functions $\mathbf{N}_{i,j}$. The surface is expressed in the following formula:

$$S(u,v) = \sum_{i=0}^{M} \sum_{j=0}^{N} \mathbf{N}_{i,j}(u,v) P_{i,j},$$

where

$$\mathbf{N}_{i,j}(u,v) = \mathbf{N}_i^m(u) \mathbf{N}_j^n(v),$$

and the $\mathbf{N}_i^m$ and $\mathbf{N}_j^n$ are degree m and n B-splines for the two parametric domain directions. Note that $\mathbf{N}_i^m$ and $\mathbf{N}_j^n$ are determined by two knot vectors $u_0, \ldots, u_{M+m-1}$ and $v_0, \ldots, v_{N+n-1}$.

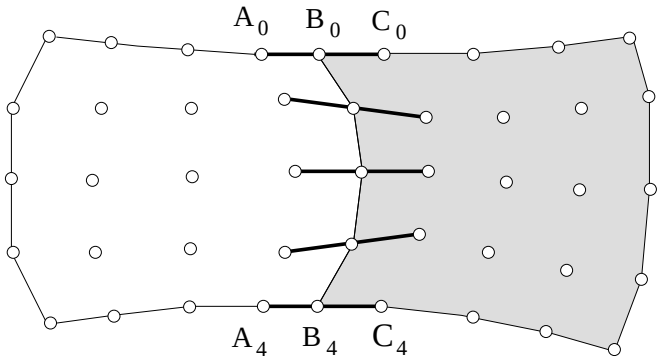


Figure 2.2: Example of $\mathcal{C}^1$ surface continuity

The continuity of B-spline surfaces can be described in terms of the continuity of B-spline curves. The continuity of the join between two B-spline surfaces can be determined by considering each control point row (or column) of the surfaces as a B-spline curve. If each of the curves meets with $\mathcal{C}^0$ or $\mathcal{C}^1$ continuity, then so will the surfaces. An example of two surfaces meeting with $\mathcal{C}^1$ continuity can be found in Figure 2.2. For each i, the points A_i, B_i, and C_i are collinear. In addition, the ratios $\frac{|\overline{A_iB_i}|}{|\overline{B_iC_i}|}$ conform to Equation 2.1 in Section 2.2.

2.4 Surface Pasting

Surface pasting is a technique developed by Bartels and Forsey [BF91] for adding local detail to surfaces. The procedure works by attaching a tensor product surface, called a *feature*, to a second tensor product surface, called a *base*. Once a feature is pasted, the feature reflects both its original shape and the topography of the underlying surface, which remains unchanged.

Barghiel [Bar94, BBF95] extended this technique to produce hierarchical surface pasting in which the base surface can be a composite surface formed by previous surface pasting operations. In the remainder of my thesis, I will deal exclusively with hierarchical surface pasting.

In this section, I explain how a feature surface is pasted onto a base surface. First, I describe

the *diffuse coordinate system* and *Greville displacement B-splines*. Then, I outline the steps involved in pasting Greville displacement B-splines.

2.4.1 The Diffuse Coordinate System

The surface pasting process uses the *diffuse coordinate system* to ensure that the pasted features reflect the shape of the underlying surface. In the diffuse coordinate system, there is a *diffuse coordinate space* where each control point has an associated local coordinate frame $\mathcal{F}_{i,j}$. Each local coordinate frame consists of an origin $\mathcal{O}_{i,j}$ and three vectors $\vec{x}_{i,j}$, $\vec{y}_{i,j}$, and $\vec{z}_{i,j}$ that define the coordinate frame.

In surface pasting, each control point can be rewritten as an origin $\mathcal{O}_{i,j}$ plus an offset or displacement vector $\vec{d}_{i,j}$ that is expressed relative to $\mathcal{F}_{i,j}$:

$$\begin{aligned} S(u,v) &= \sum_{i=0}^{M}\sum_{j=0}^{N}\mathbf{N}_{i,j}(u,v)P_{i,j} \\ &= \sum_{i=0}^{M}\sum_{j=0}^{N}\mathbf{N}_{i,j}(u,v)(\mathcal{O}_{i,j}+\vec{d}_{i,j}). \end{aligned}$$

The choice of origin and offset vector is important to the quality and behaviour of the pasting process. In the next section, I describe how they are chosen by explaining Greville displacement B-splines.

2.4.2 Greville Displacement B-splines

In this section, I describe the conversion of ordinary B-splines into Greville displacement B-splines [Bar94]. This involves determining how $\mathcal{O}_{i,j}$ and $\vec{d}_{i,j}$ are chosen for each feature control point $P_{i,j}$.

The first step is to embed the feature's two dimensional domain $\mathcal{D}$ into its range space. Each of the feature's domain points is now represented as a three dimensional point $(u,v,0)$, where $(u,v) \in \mathcal{D}$. I will refer to the point $(u,v,0)$ as an embedded domain point.

Each feature control point $P_{i,j}$ has an associated Greville point $\gamma_{i,j}$. A *Greville point* is a domain point consisting of two Greville abscissae (defined in Section 2.1), one for each of the two parametric domain directions:

$$\gamma_{i,j} = (\gamma_i, \gamma_j).$$

The surface point $S(\gamma_i, \gamma_j)$ is the point on the surface that is maximally influenced by $P_{i,j}$. The embedded Greville point is chosen to be the origin $\mathcal{O}_{i,j}$ of the local coordinate frame $\mathcal{F}_{i,j}$:

$$\mathcal{O}_{i,j} = (\gamma_{i,j}, 0) = (\gamma_i, \gamma_j, 0).$$

Now, the displacement vector $\vec{d}_{i,j}$, known as the *Greville displacement*, represents the vector between $\mathcal{O}_{i,j}$ and $P_{i,j}$:

$$\vec{d}_{i,j} = P_{i,j} - \mathcal{O}_{i,j}.$$

The diffuse representation of a tensor product B-spline surface is

$$S(u, v) = \sum_{i=0}^{M} \sum_{j=0}^{N} \mathbf{N}_{i,j}(u, v)(\mathcal{O}_{i,j} + \vec{d}_{i,j}).$$

Recall from Section 2.4.1 that each local coordinate frame consists of three vectors in addition to the origin:

$$\mathcal{F}_{i,j} = (\mathcal{O}_{i,j}, \vec{x}_{i,j}, \vec{y}_{i,j}, \vec{z}_{i,j}).$$

The vectors $\vec{x}_{i,j}$ and $\vec{y}_{i,j}$ are the two parametric domain directions at $\mathcal{O}_{i,j}$ and $\vec{z}_{i,j} = \vec{x}_{i,j} \times \vec{y}_{i,j}$. The Greville displacement can be represented as a linear combination of the three coordinate frame vectors,

$$\vec{d}_{i,j} = \rho_{i,j}\vec{x}_{i,j} + \sigma_{i,j}\vec{y}_{i,j} + \tau_{i,j}\vec{z}_{i,j},$$

where $\rho_{i,j}$, $\sigma_{i,j}$, and $\tau_{i,j}$ are the scalar components of $\vec{d}_{i,j}$. A tensor product B-spline surface can be thought of as a linear combination of Greville displacements. The formula for a Greville

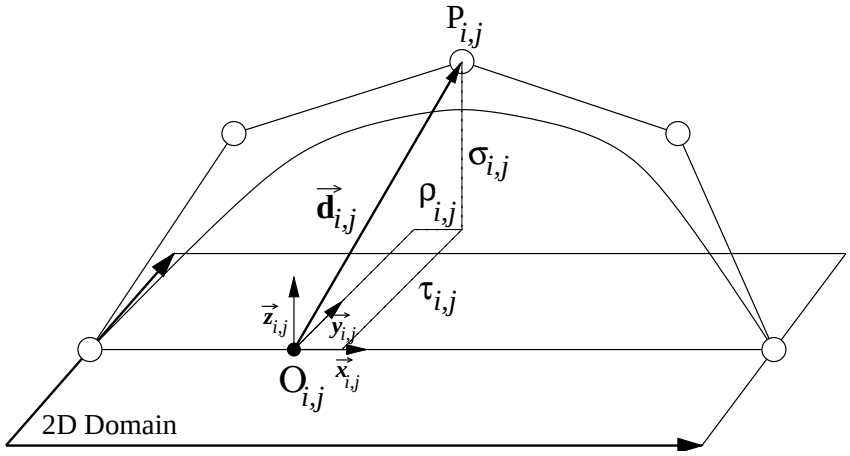


Figure 2.3: A Greville displacement B-spline surface

displacement B-spline, the type of B-spline used in surface pasting, is

$$S(u,v) = \sum_{i=0}^{M} \sum_{j=0}^{N} \mathbf{N}_{i,j}(u,v)\vec{d}_{i,j}.$$

An example of a Greville displacement B-spline is shown in Figure 2.3. It illustrates how a control point ($P_{i,j}$), its local coordinate frame ($\mathcal{F}_{i,j} = \{\mathcal{O}_{i,j}, \vec{x}_{i,j}, \vec{y}_{i,j}, \vec{z}_{i,j}\}$), and its Greville displacement ($\vec{d}_{i,j}$) are related.

2.4.3 Pasting Greville Displacement B-splines

The surface pasting process (illustrated in Figure 2.4) is a computationally inexpensive method for adding local detail. In this section, I describe how to paste a feature surface S_F onto a base surface S_B, where S_F and S_B have separate domains. The pasting process involves finding new control point locations for S_F, which is expressed as a Greville displacement B-spline relative to S_B.

The first step is to map the feature's domain into the base's domain using an invertible transformation T. This transformation determines the location and size of the feature surface relative to the base surface. For each feature control point, the corresponding Greville point is

mapped into the base domain:

$$T(\gamma_{i,j}) = T(\gamma_i, \gamma_j) = (\gamma'_i, \gamma'_j) = \gamma'_{i,j}.$$

T is also applied to $\vec{x}_{i,j}$ and $\vec{y}_{i,j}$ at each Greville point to give the vectors $\vec{x'}_{i,j}$ and $\vec{y'}_{i,j}$.

Now, the base surface S_B is evaluated at (γ'_i, γ'_j) to form $S_B(\gamma'_i, \gamma'_j)$ on the surface. A new local coordinate frame $\mathcal{F}^B_{i,j} = \{S_B(\gamma'_i, \gamma'_j), \vec{r}_{i,j}, \vec{s}_{i,j}, \vec{t}_{i,j}\}$ is constructed. The point $S_B(\gamma'_i, \gamma'_j)$ is the new origin on the base surface and the vectors $\vec{r}_{i,j}$ and $\vec{s}_{i,j}$ are the two partial derivatives at $S_B(\gamma'_i, \gamma'_j)$ in the $\vec{x'}_{i,j}$ and $\vec{y'}_{i,j}$ directions:

$$\begin{aligned} \vec{r}_{i,j} &= \frac{\partial}{\partial \vec{x'}_{i,j}} S(\gamma'_i, \gamma'_j), \\ \vec{s}_{i,j} &= \frac{\partial}{\partial \vec{y'}_{i,j}} S(\gamma'_i, \gamma'_j), \text{ and} \\ \vec{t}_{i,j} &= \vec{r}_{i,j} \times \vec{s}_{i,j}. \end{aligned}$$

This choice of local coordinate frame ensures that the pasted control point $P'_{i,j}$ will reflect the shape of the base surface.

The feature control point is placed by expressing its displacement vector relative this new local coordinate frame $\mathcal{F}^B_{i,j}$. The new control point location $P'_{i,j}$ is found using the following point-vector addition equation:

$$P'_{i,j} = S_B(\gamma'_i, \gamma'_j) + \vec{d'}_{i,j},$$

where

$$\vec{d'}_{i,j} = \rho_{i,j}\vec{r}_{i,j} + \sigma_{i,j}\vec{s}_{i,j} + \tau_{i,j}\vec{t}_{i,j}. \tag{2.2}$$

This procedure, which is performed for all the feature's control points, gives the pasted feature.

2.4.4 Continuity of Pasted Surfaces

To ensure that the boundary of the feature lies near the base surface, the first layer of the feature's control points (the black points of Figure 2.5) are placed at their Greville points so they have zero

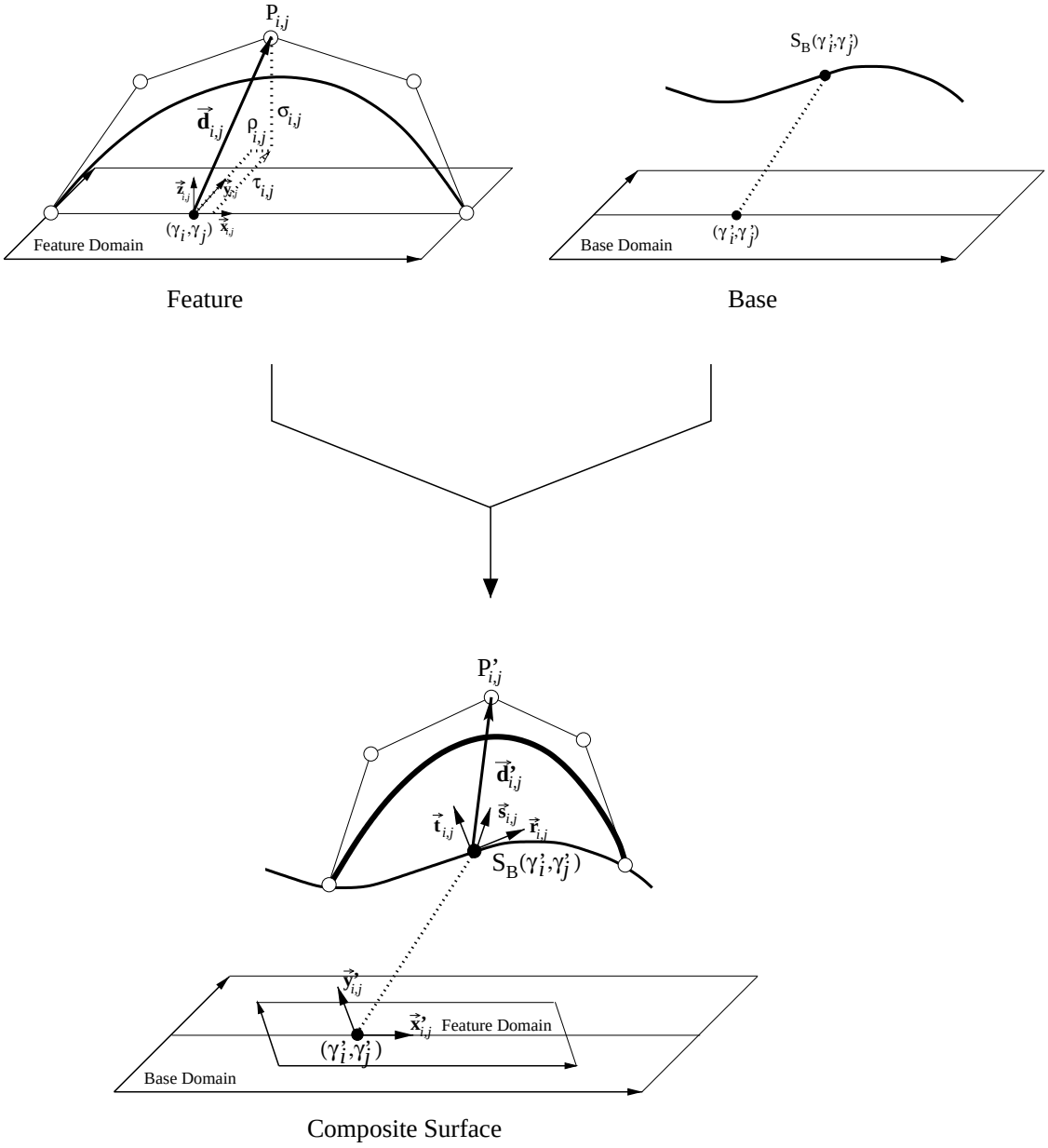


Figure 2.4: Pasting a Greville displacement B-spline surface

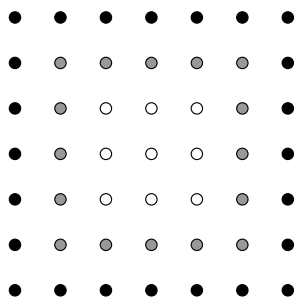


Figure 2.5: Outer two layers of feature's control points placed at Greville points

displacement vectors. After pasting, these feature control points will lie on the base surface, and the boundary of the feature will lie near the base. By inserting knots into the feature surface, the discontinuity between the feature and the base can be made as small as desired.

By placing a second layer of the feature's control points (the grey points of Figure 2.5) at their Greville points, an approximate $\mathcal{C}^1$ join between the feature and the base is achieved. Conrad [CM00] gives a further discussion of continuity issues of pasted surfaces, and shows how to use quasi-interpolation to further reduce both the $\mathcal{C}^0$ and $\mathcal{C}^1$ discontinuity between the feature and the base.

2.5 User Interfaces

In this section, I describe the user interface for previous surface pasting editors. Then, I present Forsey's approach to the direct manipulation of hierarchical B-splines [FB88, For90].

2.5.1 Previous Pasting Editors

Early pasting editors [BF91, BBF95] allowed users to paste, translate, rotate, and resize features via a domain space user interface while observing the results in a world space window. However, the domain space user interface was inconvenient since the user's attention was divided between the domain space and world space representations. Positioning a feature was accomplished through trial and error. A superior interface would allow the user to interact with the three dimensional

model directly. Chan [CMB97] developed such an interface that mapped three dimensional user actions into two dimensional domain operations. The world space user interface allowed users to focus solely on the model and gave better feedback for positioning features on a base. However, direct manipulation of pasted surfaces was not supported, i.e., there was no way to reliably move a given point on a surface to a specified point in space.

2.5.2 Direct Manipulation of Hierarchical B-splines

Forsey [FB88, For90] implemented a limited direct manipulation scheme for hierarchical B-splines. The user is able to manipulate the surface at *edit points*, which are the images of the Greville points. The control point associated with the Greville point is moved to cause the edit point to follow the mouse cursor. Note that in this interface, the edit points, not the control points of the surface, are seen by the user.

When the user has selected an edit point, the region of the surface that will be affected by the manipulation is highlighted. The user then has the option to lower the resolution level to expand the affected region. If the resolution level is lowered, the view of the current level is removed from the display. The user can then select an edit point at the lower resolution level and manipulate the surface, or lower the resolution level again. Unfortunately, with these levels stripped out of the display, the user is not able to see how his manipulation is affecting the higher level surfaces.

2.6 Direct Manipulation

One of my goals for the direct manipulation of pasted surfaces is to provide the user with the ability to manipulate any point on the composite surface and to allow him to view the entire pasting hierarchy while editing. My method for the direct manipulation of pasted surfaces is a variation of a technique developed by Bartels and Beatty [BB89] for the direct manipulation of spline curves. In Section 2.6.1, I describe their method and in Section 2.6.2, I describe how I extended their method for the direct manipulation of tensor product surfaces. Bartels and Beatty dealt with arbitrary B-spline curves while my discussion is restricted to full multiplicity end knot

curves and surfaces since they are applicable to pasted surfaces.

2.6.1 B-spline Curves

Traditionally, curve manipulation was accomplished through the manipulation of control points, but the control points must be displayed, which increases the clutter on the screen. In addition, manipulating control points is unintuitive; the change observed in the curve due to the motion of a control point is difficult for an inexperienced user to predict.

The direct manipulation of B-spline curves eliminates these drawbacks by enabling the user to select a point on the curve and move it to a new location. The control points of the curve are updated to produce a curve that passes through the new location. Bartels and Beatty [BB89] explained that there are many ways in which the control points can be moved to bring about a specified change and recommended the following procedure to update the control points.

Given a curve $C(u) = \sum_{i=0}^{M} \mathbf{N}_i^m(u) P_i$, suppose a curve point $C(\overline{u})$ is moved by $\overrightarrow{\Delta P}$. To update the control points, a weight w_i is computed for each control point:

$$w_i = \frac{\mathbf{N}_i^m(\overline{u})}{\sum_{j=0}^{M} \left(\mathbf{N}_j^m(\overline{u})\right)^2}.$$

The Bartels-Beatty method then updates each control point as follows:

$$P_i' = P_i + w_i \overrightarrow{\Delta P} \ .$$

The resulting curve $C'(u) = \sum_{i=0}^{M} \mathbf{N}_i^m(u) P_i'$ is such that $C'(\overline{u}) = C(\overline{u}) + \overrightarrow{\Delta P}$. Note that the selected point is not necessarily the point of maximal change in the curve, but in most cases it is close enough to the point of maximal change that the difference is undetectable to the user.

For some applications, it is advantageous to fix the endpoints of a curve. It is possible to modify the direct manipulation method so the endpoints do not move. This is done by ensuring that the first and last control points use a zero weight to calculate their displacements. For the updated curve to pass through the new location, the other weights must be calculated in a slightly

different manner:

$$w_i = \frac{\mathbf{N}_i^m(\overline{u})}{\sum_{j=1}^{M-1} \left(\mathbf{N}_j^m(\overline{u})\right)^2},$$

where $w_0 = w_M = 0$. This equation is the same as the original one except that the first and last control points do not contribute to the point on the curve and their basis functions are not part of the calculations. As an alternative to fixing the endpoints, it is possible to fix any subset of the control points. Suppose $\mathcal{I}$ is the set of indices of the control points to be modified. Then

$$w_i = \begin{cases} \dfrac{\mathbf{N}_i^m(\overline{u})}{\sum_{j\in\mathcal{I}} \left(\mathbf{N}_j^m(\overline{u})\right)^2} & \text{if } i \in \mathcal{I} \\ 0 & \text{otherwise.} \end{cases}$$

One problem that can arise from fixing the endpoints of a curve occurs when the user has selected a point close to a fixed endpoint. There is a nearby section of the curve over which the moving control points have more influence than they do over the selected point. This section will move farther than the selected point, introducing an unsightly distortion in the curve, as described by Bartels and Beatty [BB89].

A second problem can arise from fixing too many of the curve's control points. Bartels and Beatty showed that their direct manipulation method was unstable when only the single most influential control point was moved. This instability occurs because there will be a point on the curve to the left of which control point P_i will have the most influence, and to the right of which control point P_{i+1} will have the most influence. Picking near this division can have markedly different results depending on which side was picked. A minimum of two most influential control points must be moved for their method to be stable.

Bartels and Beatty provide a theoretical justification for their method. Specifically, they use the Householder transformation to show that their w_i minimize the change in the positions of the control points.

2.6.2 Tensor Product Surfaces

A number of people have investigated techniques for the direct manipulation of tensor product surfaces. For example, Fowler proposed a method for directly manipulating positions, normal vectors, and partial derivatives at any point on a curve or surface [FB93, Fow92]. He also found that the system of equations that must be solved to perform direct manipulation of tensor product surfaces is underdetermined.

I have chosen to calculate new control point locations using a generalization of Bartels and Beatty's curve manipulation technique [BB89] where the extra degrees of freedom are used to reduce the overall change in the position of the surface's control points. In this section, I describe how I extended and altered Bartels and Beatty's method so that it can be applied to the direct manipulation of tensor product B-spline surfaces.

Given a surface $S(u,v) = \sum_i \sum_j \mathbf{N}_{i,j}(u,v) P_{i,j}$, suppose a surface point $S(\overline{u},\overline{v})$ is moved by a vector $\overrightarrow{\Delta P}$. It is necessary to find a surface S' such that

$$S'(\overline{u},\overline{v}) = S(\overline{u},\overline{v}) + \overrightarrow{\Delta P} \; . \tag{2.3}$$

The first step is to find a block of control points that has the most influence over the picked surface point. For each control point $P_{i,j}$ in this block, a weight $w_{i,j}$ is calculated that is proportional to the control point's contribution to the surface point $S(\overline{u},\overline{v})$:

$$w_{i,j} = \frac{\mathbf{N}_{i,j}(\overline{u},\overline{v})}{\sum_k \sum_l \left(\mathbf{N}_{k,l}(\overline{u},\overline{v})\right)^2}, \tag{2.4}$$

where the double summation is over the block of control points being modified. Then each feature control point is updated as follows:

$$P'_{i,j} = P_{i,j} + w_{i,j} \, \overrightarrow{\Delta P} \; . \tag{2.5}$$

Just as in the curve case, this method does not guarantee that the selected point is the point

of maximal change, but except in the distortion case mentioned below, it is close enough to the point of maximal change that the difference is undetectable to the user.

It is easy to verify that the above equations produce the desired direct manipulation. Suppose the user has selected the point $S(\overline{u},\overline{v})$ to manipulate, and $\mathcal{I}$ is the set of indices of the control points to be modified. Then $S(\overline{u},\overline{v}) = \sum_{i=0}^{M}\sum_{j=0}^{N}\mathbf{N}_{i,j}(\overline{u},\overline{v})P_{i,j}$ and it must be shown that $S'(\overline{u},\overline{v}) = S(\overline{u},\overline{v}) + \overrightarrow{\Delta P}$:

$$
\begin{aligned}
S'(\overline{u},\overline{v}) &= \sum_{i=0}^{M}\sum_{j=0}^{N}\mathbf{N}_{i,j}(\overline{u},\overline{v})P'_{i,j} \\
&= \sum_{i=0}^{M}\sum_{j=0}^{N}\mathbf{N}_{i,j}(\overline{u},\overline{v})[P_{i,j} + w_{i,j}\,\overrightarrow{\Delta P}] \\
&= \sum_{i=0}^{M}\sum_{j=0}^{N}\mathbf{N}_{i,j}(\overline{u},\overline{v})P_{i,j} + \overrightarrow{\Delta P}\ \underbrace{\left[\sum_{i=0}^{M}\sum_{j=0}^{N}\mathbf{N}_{i,j}(\overline{u},\overline{v})w_{i,j}\right]}_{\text{Note that } w_{i,j}=0 \text{ when } (i,j)\notin\mathcal{I}.} \\
&= S(\overline{u},\overline{v}) + \overrightarrow{\Delta P}\left[\sum_{(i,j)\in\mathcal{I}}\mathbf{N}_{i,j}(\overline{u},\overline{v})\frac{\mathbf{N}_{i,j}(\overline{u},\overline{v})}{\sum_{(k,l)\in\mathcal{I}}\left(\mathbf{N}_{k,l}(\overline{u},\overline{v})\right)^2}\right] \\
&= S(\overline{u},\overline{v}) + \overrightarrow{\Delta P}\left[\frac{\sum_{(i,j)\in\mathcal{I}}\left(\mathbf{N}_{i,j}(\overline{u},\overline{v})\right)^2}{\sum_{(k,l)\in\mathcal{I}}\left(\mathbf{N}_{k,l}(\overline{u},\overline{v})\right)^2}\right] \\
&= S(\overline{u},\overline{v}) + \overrightarrow{\Delta P}\ .
\end{aligned}
$$

This method produces the desired result, but there is a potential problem. Just as at least two control points are needed to produce a stable change in the curve case, at least two control points are needed in each of the two parametric domain directions to get a stable direct manipulation method for surfaces. I chose to adjust a 2×2 block of control points since this small block size restricts the locality of change. A larger block may be used to modify a larger area of the surface.

As in the curve case, it is also possible to fix the boundary control points or any subset of

the control points of a surface. However, when a point close to a fixed control point is moved, distortions similar to those in the curve case will occur.

Chapter 3

Direct Manipulation of Pasted Surfaces

3.1 Motivation and Goals

Previous surface pasting editors allowed the user to paste features on top of base surfaces and translate, rotate, and resize pasted features. None of these editors allowed a user to directly manipulate surfaces in the pasting hierarchy. The motivation for my research is to give the user a more flexible paradigm for modelling pasted surfaces by providing an interface that allows him to 1) pick any point on a surface and move it to a new location, and 2) choose the granularity of the change effected by the manipulation.

In this chapter, I explain how the technique for the direct manipulation of tensor product surfaces from Section 2.6.2 can be applied to pasted surfaces. I first describe a problem associated with a naive approach to manipulating simple pasted surfaces and how to avoid the problem. Then, I present five methods for the direct manipulation of hierarchical pasted surfaces and talk about issues that arise with each method. At the end, I show colour plates that illustrate these methods.

3.2 Potential Problems and Issues

To begin, I will look at certain problems that may arise if the direct manipulation technique is applied to a pasted surface. When the user picks a surface point near the boundary, the control points in the two outermost rings may be affected. Moving control points in the second outermost ring will reduce the approximate $\mathcal{C}^1$ continuity between the feature and the base, and the composite surface will look less smooth. If control points in the outermost ring are moved, then the feature may detach from its underlying base. I prevent problems that could arise from the movement of the two outermost rings of control points by ensuring that these control points are unmoveable.

If the user attempts to move a surface point whose most influential 2×2 block of control points intersects the two outermost rings, there are two options: disallow the direct manipulation, or find the closest block that does not overlap the two outermost rings. In the latter case, the control points that are changed have less influence on the selected surface point, and thus they must be displaced farther to move the picked surface point to its new location. This can cause unsightly distortions, similar to those described by Bartels and Beatty [BB89], in the area of the surface over which these alternative control points have a higher influence; see Figure 3.9 on page 36 for an example. These distortions would likely confuse the user since the maximal change in the surface would not occur at the picked point. For all but one of my direct manipulation methods, I chose to disallow direct manipulation of a surface when the most influential 2×2 block of control points intersects the two outermost rings of the surface's control points.

In the remainder of my thesis, I use the term *unmodifiable region* to refer to the area of a surface where any point picked from the area has a 2×2 block of control points that intersects the surface's two outermost rings of control points. The term *modifiable region* has an analogous definition.

3.3 Direct Manipulation of Simple Pasted Surfaces

In this section, I discuss a method for the direct manipulation of a pasted feature. This low level technique is used as a tool in the direct manipulation of composite pasted surfaces. Several high level methods for directly manipulating composite surfaces are discussed in Sections 3.4–3.8.

The first step in the direct manipulation of a pasted feature is to update the control points of the surface the user wishes to modify using direct manipulation of a tensor product surface as described in Section 2.6.2. Then the displacement vector $\vec{d}_{i,j}$ is updated for each control point $P_{i,j}$.

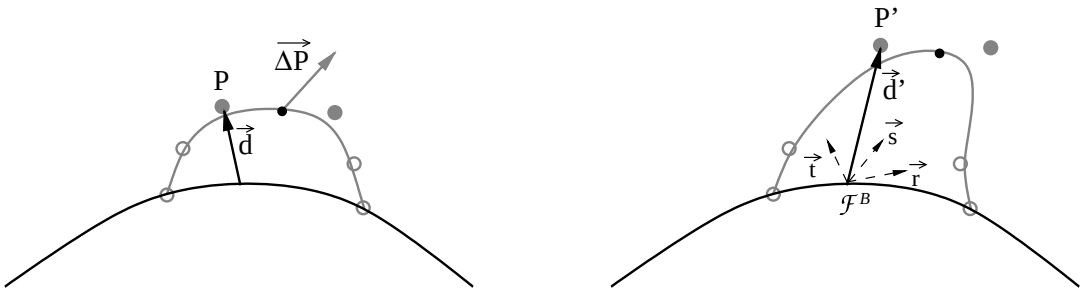


Figure 3.1: Updating a control point's displacement vector

To recalculate each control point's displacement vector, the local coordinate frame $\mathcal{F}^B_{i,j} = \{\mathcal{O}'_{i,j}, \vec{r}_{i,j}, \vec{s}_{i,j}, \vec{t}_{i,j}\}$ on the base surface must be reconstructed. The difference $\vec{d'}_{i,j}$ between the new control point location $P'_{i,j}$ and the origin of the coordinate frame $\mathcal{O}'_{i,j}$ is found and expressed in terms of $\mathcal{F}^B_{i,j}$:

$$\vec{d'}_{i,j} = \rho'_{i,j}\vec{r}_{i,j} + \sigma'_{i,j}\vec{s}_{i,j} + \tau'_{i,j}\vec{t}_{i,j}.$$

The formula for $\vec{d'}_{i,j}$ can be written as a product of matrices:

$$\begin{bmatrix} d'_{i,j,x} \\ d'_{i,j,y} \\ d'_{i,j,z} \end{bmatrix} = \begin{bmatrix} r_{i,j,x} & s_{i,j,x} & t_{i,j,x} \\ r_{i,j,y} & s_{i,j,y} & t_{i,j,y} \\ r_{i,j,z} & s_{i,j,z} & t_{i,j,z} \end{bmatrix} \begin{bmatrix} \rho'_{i,j} \\ \sigma'_{i,j} \\ \tau'_{i,j} \end{bmatrix}.$$

From this the formula for $\rho'_{i,j}$, $\sigma'_{i,j}$, and $\tau'_{i,j}$ can be derived:

$$\begin{bmatrix} \rho'_{i,j} \\ \sigma'_{i,j} \\ \tau'_{i,j} \end{bmatrix} = \begin{bmatrix} r_{i,j,x} & s_{i,j,x} & t_{i,j,x} \\ r_{i,j,y} & s_{i,j,y} & t_{i,j,y} \\ r_{i,j,z} & s_{i,j,z} & t_{i,j,z} \end{bmatrix}^{-1} \begin{bmatrix} d'_{i,j,x} \\ d'_{i,j,y} \\ d'_{i,j,z} \end{bmatrix}.$$

After the new displacement vectors are calculated, the feature may be translated and the underlying surface may be changed, and the results of the direct manipulation are preserved.

The following methods represent stages in the evolution of the direct manipulation of pasted surfaces. Each of the first four is an experimental method that served as a stepping stone in the development of the fifth method, hierarchical direct manipulation. This final method is the most flexible, combining ideas from the first three.

3.4 Top Level

This is the simplest of the schemes for directly manipulating a pasted surface. Direct manipulation is applied to the top level surface at the picked point. A disadvantage to this method is that if the picked point lies in the unmodifiable region of the surface, it is left unchanged. An advantage to this method is that the user can edit the composite surface at various resolutions. For example, if a composite surface has multiple resolution levels, the user can edit at any of the resolutions by selecting a top level point from the desired resolution level. However, for a given surface point, the user can only manipulate the composite surface at a single resolution. Consider Figure 3.2, in which the shaded areas indicate the regions of the surfaces that can not be manipulated with the top level method, and each unshaded area is labelled with the available resolution level in that region. If the user has selected a point in the region indicated by 3, he can not reduce the resolution level to 2 or 1; he would have to select a point from one of the regions labelled 2 or 1 to make such a broad change.

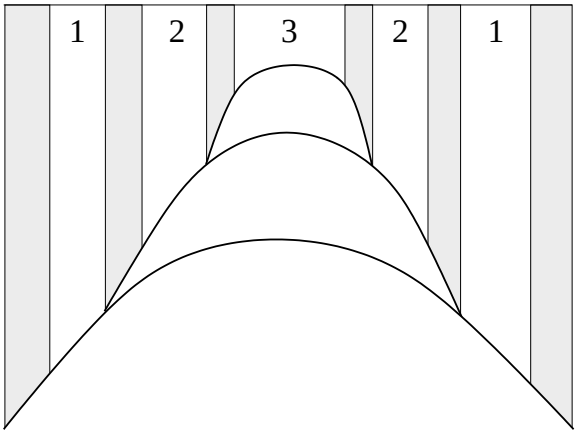


Figure 3.2: Example of available resolution levels with the top level method

3.5 Base Only

The base only direct manipulation method is also simple. The picked point's domain value is mapped into the base's domain and the base surface point with that domain value is moved by $\vec{\Delta P}$. The surfaces on top of the base change shape as a result of the modification to the base. If the selected surface is not the base, then this method does not yield direct manipulation and there will be a noticeable gap between the mouse cursor and the picked point. Note that if the point on the base surface is in the unmodifiable region, no manipulation is performed.

3.6 Full Hierarchy

The base only direct manipulation method does not give true direct manipulation. In many cases, the picked point will not follow the mouse cursor. An ideal solution would be to apply a specific change to the base surface that would cause the selected point to move by the desired amount. Such a change could be calculated by deriving for the pasting hierarchy formulae similar to Equations 2.3, 2.4, and 2.5. Expanding Equation 2.3 for a pasted surface yields

$$\sum_i \sum_j N_{i,j}(\overline{u},\overline{v})\left(\mathcal{O}''_{i,j} + \vec{d}''_{i,j}\right) = \sum_i \sum_j N_{i,j}(\overline{u},\overline{v})\left(\mathcal{O}'_{i,j} + \vec{d}'_{i,j}\right) + \vec{\Delta P}\,. \tag{3.1}$$

The direct manipulation equations can be derived by expanding $\vec{d}''_{i,j}$ (Equation 2.2) and $\mathcal{O}''_{i,j}$ (an evaluation of the base surface). Unfortunately, $\vec{d}''_{i,j}$ depends upon the control points of the base surface in a non-linear manner as seen from Equation 2.2 and the formula for $\vec{t}_{i,j}$. Thus, this method for directly manipulating hierarchical pasted surfaces is more expensive than desired, and as the depth of the hierarchy is increased, the equations become more complicated.

As an alternative to the method described above, I chose to augment the base only direct manipulation technique so the picked point will always move by the desired amount. The idea behind the full hierarchy method is to push all the work down the pasting hierarchy so a large change is made at a coarse resolution. Smaller adjustments are made at higher resolutions to achieve direct manipulation.

Suppose there is a hierarchy of pasted surfaces, $S_0, \ldots, S_H$, with S_0 being the coarsest resolution, and a point $S_H(\overline{u}, \overline{v})$ has been selected and moved by $\overrightarrow{\Delta P}$. The composite surface is updated according to the following procedure.

1. Descend the hierarchy of surfaces under the picked point until surface S_0 is reached.
2. Adjust S_0 so that $S_0(\overline{u}, \overline{v})$ is moved by $\overrightarrow{\Delta P}$.
3. Then for each surface S_i, for $i = 1$ to H: (See Figure 3.3 for an illustration of this technique applied to a curve)
   (a) The adjustment made to S_{i-1} caused S_i to change, giving a new surface S'_i. In general, $S_i(\overline{u}, \overline{v})$ will not have moved by $\overrightarrow{\Delta P}$.
   (b) Compute the difference between the desired change and the actual change in S_i, giving a *correction factor* $\overrightarrow{\Delta P'} = \overrightarrow{\Delta P} - \Big(S'_i(\overline{u}, \overline{v}) - S_i(\overline{u}, \overline{v})\Big)$.
   (c) Directly manipulate S'_i by $\overrightarrow{\Delta P'}$, giving a new surface S''_i such that $S''_i(\overline{u}, \overline{v}) = S_i(\overline{u}, \overline{v}) + \overrightarrow{\Delta P}$.

The point $(\overline{u}, \overline{v})$ is taken from the domain of the selected surface. In the preceding discussion, $S_i(\overline{u}, \overline{v})$ is not to be interpreted as the evaluation of surface S_i at $(\overline{u}, \overline{v})$. Rather, I use the notation $S_i(\overline{u}, \overline{v})$ to refer to the evaluation of S_i at the image of $(\overline{u}, \overline{v})$ mapped into the domain of

S_i according to the invertible domain transformations defined by the surface pasting operation. I will use the same convention when discussing the hybrid and hierarchical direct manipulation methods.

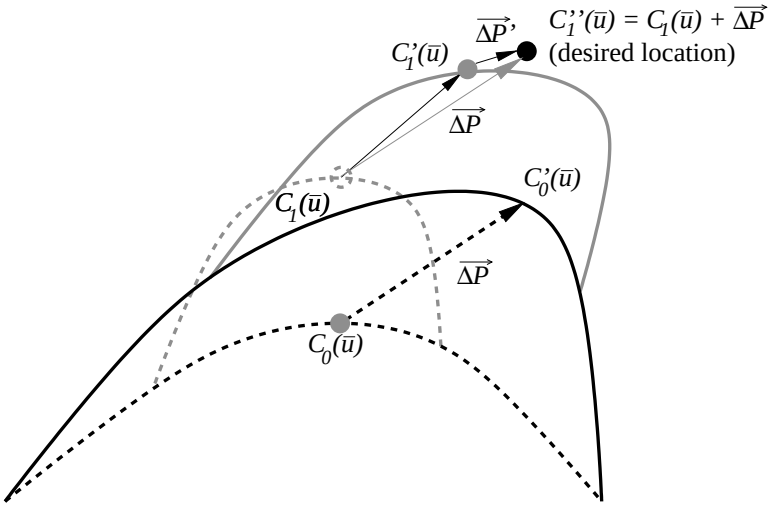


Figure 3.3: Detailed view of steps involved in the full hierarchy method

An example of the full hierarchy method for a curve is illustrated in Figure 3.4; the construction for surfaces is analogous. Initially, the user has chosen a point $C_1(\overline{u})$ to manipulate and the amount $\overrightarrow{\Delta P}$ by which to move the point. In diagram (b), the point $C_0(\overline{u})$ is located and moved by $\overrightarrow{\Delta P}$. The shape of the base changes, causing the feature's shape and the location of the picked point to change, as shown in (c). Since $C_1'(\overline{u})$ is not at its desired position, a correction factor is applied, giving $C_1''(\overline{u})$. The last diagram shows the resulting composite curve.

An advantage to this method is that a user can directly manipulate the composite surface as if it were a single entity. In most cases, this method bypasses the problem encountered in the top level method when the user is prohibited from moving a point in the unmodifiable region of the surface. Note that if the top level surface is unmodifiable, then this method will not move the picked point to the position of the cursor; there is likely to be a small amount of drift. The base only and hierarchical direct manipulation methods (see Section 3.8) suffer from this problem as well. Another problem with the full hierarchy method is that the base surface does most of the

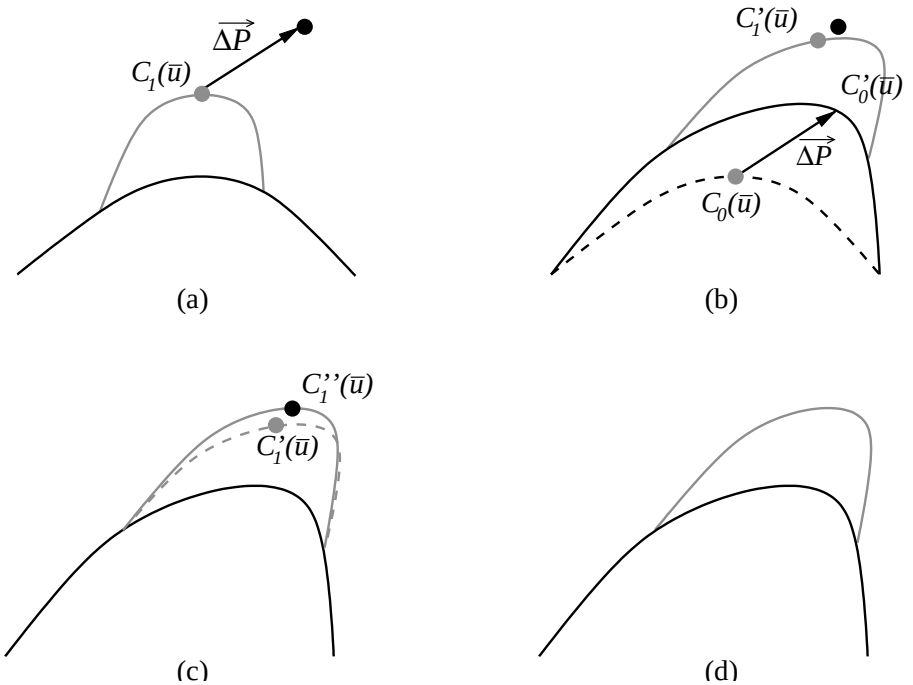


Figure 3.4: Example of the steps involved in the full hierarchy method

work and the change resulting from the direct manipulation occurs at a low resolution; i.e., it is not a multiresolution editing method.

3.7 Hybrid

The top level direct manipulation method works well when the selected point is in the modifiable region of a surface but is ineffective when a point in the unmodifiable region is selected. The full hierarchy method allows direct manipulation anywhere on the composite surface but at the expense of restricting the user to making broad changes. The hybrid method was intended to combine these two methods into a direct manipulation technique that could be used over the whole surface without forcing broad changes to be made in all situations.

The naive approach to implementing the hybrid method — invoking the top level method in the modifiable region and the full hierarchy method in the unmodifiable region — suffers from two drawbacks. First, the transition between the areas that use the top level method and the areas that use the full hierarchy method is too abrupt. There needs to be a smooth transition between these two extremes. The second drawback arises because the original full hierarchy method causes the base surface to undergo the most deformation. In many cases, it is desirable to adjust surfaces higher up the pasting hierarchy because it brings about a more localized change. In the hybrid method, when work is passed down the pasting hierarchy, it is not necessarily passed all the way to the base surface.

The idea is to find the block of control points that has the most influence over the picked surface point and that does not overlap the two outermost rings of the surface's control points. The contribution of these control points is measured against two user-defined threshold values, min and max. If the contribution is greater than max, then the surface containing the actual picked point will do all the work and if the contribution is less than min, then all the work is passed down to the next lower surface in the pasting hierarchy. If the contribution is between min and max, then the work will be split between the surface on which the selected point lies and the next lower level surface. This procedure is performed for progressively lower level surfaces until

no more work needs to be passed down the pasting hierarchy or the base surface is reached. As mentioned at the beginning of this chapter, choosing alternative control points to move can result in unsightly distortions in the surface. I had hoped that these distortions would be minimized by passing some work down the pasting hierarchy when the block of control points contribute little to the picked point.

The hybrid direct manipulation method works as follows. Suppose there is a hierarchy of pasted surfaces, $S_0, \ldots, S_H$, with S_0 being the coarsest resolution. Let the picked point be $S_h(\overline{u}, \overline{v})$ on a pasted surface at resolution h. A description of the amount of work assigned to each affected surface in the picked point's pasting hierarchy appears below.

1. Set $\alpha = 1$, $i = h$, and $\overrightarrow{\Delta P}_i = \overrightarrow{\Delta P}$.
2. Find the block of control points that does not intersect the two outermost rings of S_i's control points and that has the highest contribution c to the selected point.
   - If $c > max$, then this surface is the lowest one that needs to move, so set $\alpha = 0$.
   - Else, if $min \leq c \leq max$, then S_i can deform to some degree. Let $\alpha = \frac{max - c}{max - min}$.
   - Otherwise, $c < min$, and this surface should not deform at all, so set $\alpha = 1$.
3. If $i = 0$ or $\alpha = 0$, then set $lowest = i$ and stop.
4. Otherwise, let $\overrightarrow{\Delta P}_{i-1} = \alpha \overrightarrow{\Delta P}_i$.
5. Set $i = i - 1$ and return to step 2.

Each of the affected surfaces in the picked point's pasting hierarchy has now been assigned an amount of work that is a fraction of $\overrightarrow{\Delta P}$. The following procedure explains how the work is applied to each of the affected surfaces to move the user-selected point to its new location.

1. Set $i = lowest$.
2. Adjust S_i so that $S_i(\overline{u}, \overline{v})$ is moved by $\overrightarrow{\Delta P}_i$.
3. Repeat until $i = h$:

(a) Set $i = i + 1$.

(b) The adjustment made to S_{i-1} caused S_i to change, giving a new surface S'_i. In general, $S_i(\overline{u}, \overline{v})$ will not have moved by $\overrightarrow{\Delta P}_i$.

(c) Compute the difference between the desired change and the actual change in S_i, giving a *correction factor* $\overrightarrow{\Delta P}' = \overrightarrow{\Delta P}_i - \Big(S'_i(\overline{u}, \overline{v}) - S_i(\overline{u}, \overline{v})\Big)$.

(d) Directly manipulate S'_i by $\overrightarrow{\Delta P}'$, giving a new surface S''_i such that $S''_i(\overline{u}, \overline{v}) = S_i(\overline{u}, \overline{v}) + \overrightarrow{\Delta P}_i$.

The hybrid direct manipulation method works well when the most influential 2×2 block of control points does not intersect the boundary control points of the selected surface. Whenever alternative blocks of control points were found, distortions occurred. I had hoped that the distortions caused by using alternative blocks of control points would be offset by passing some of the work down the pasting hierarchy. However, the distortions were not reduced enough.

I have considered several possible improvements to the hybrid method. When the picked point was in the transition area of the surface, I calculated the amount of work to be passed down the hierarchy using a linear blending function, and in certain situations the surface became distorted. It is possible the distortions could be eased if a different blending function is used, such as a quadratic one. Alternatively, the distortions might be reduced by refraining from applying correction factors in certain circumstances. In the implementation of the hybrid method, I chose to apply correction factors to each of the affected surfaces, even when alternative blocks of control points were found. Smaller distortions might have occurred if correction factors were not applied in this case, at the cost of not having true direct manipulation when alternative control points are found on the top level surface. Another improvement to this method might be to push all of the work to a lower level surface when the most influential block of control points intersects the boundary control points of the surface. Then, only the correction factor would be applied to the higher level surface and so the distortions might be lessened.

However, after interacting with the hybrid method, I decided that it was not a good method for manipulating pasted surfaces. Consequently, I abandoned this line of research in favour of

investigating a different method that gives the user the ability to manipulate the picked point at any level in the pasting hierarchy; a method that I discuss in the next section.

3.8 Hierarchical

The hierarchical direct manipulation method, a generalization of the full hierarchy direct manipulation method, was created to give the user more control over the resolution level of the manipulation. In the full hierarchy method, the lowest modifiable surface is selected to do the majority of the work and the higher surfaces are updated by smaller correction factors. The hierarchical direct manipulation method extends full hierarchy by allowing the user to choose the surface that does the greatest amount of work. The lower the chosen surface is in the hierarchy of pasted surfaces, the broader the change in the manipulation.

The idea behind this method is to push the work down the pasting hierarchy, make a large change at the lowest level desired, and then ascend the hierarchy making small adjustments as needed. Suppose there is a hierarchy of pasted surfaces, $S_0, \ldots, S_H$, with S_0 being the coarsest resolution. Let the picked point be $S_h(\overline{u}, \overline{v})$ on a pasted surface at resolution h, and suppose the user wishes to edit at resolution r, with $h \geq r$.

1. Descend the hierarchy of surfaces under the picked point until surface S_r is reached.

2. Adjust S_r so that $S_r(\overline{u}, \overline{v})$ is moved by $\overrightarrow{\Delta P}$.

3. Then for each surface S_i, for $i = r + 1$ to h:

   (a) The adjustment made to S_{i-1} caused S_i to change, giving a new surface S_i'. In general, $S_i(\overline{u}, \overline{v})$ will not have moved by $\overrightarrow{\Delta P}$.

   (b) Compute the difference between the desired change and the actual change in S_i, giving a *correction factor* $\overrightarrow{\Delta P'} = \overrightarrow{\Delta P} - \Big(S_i'(\overline{u}, \overline{v}) - S_i(\overline{u}, \overline{v})\Big)$.

   (c) Directly manipulate S_i' by $\overrightarrow{\Delta P'}$, giving a new surface S_i'' such that $S_i''(\overline{u}, \overline{v}) = S_i(\overline{u}, \overline{v}) + \overrightarrow{\Delta P}$.

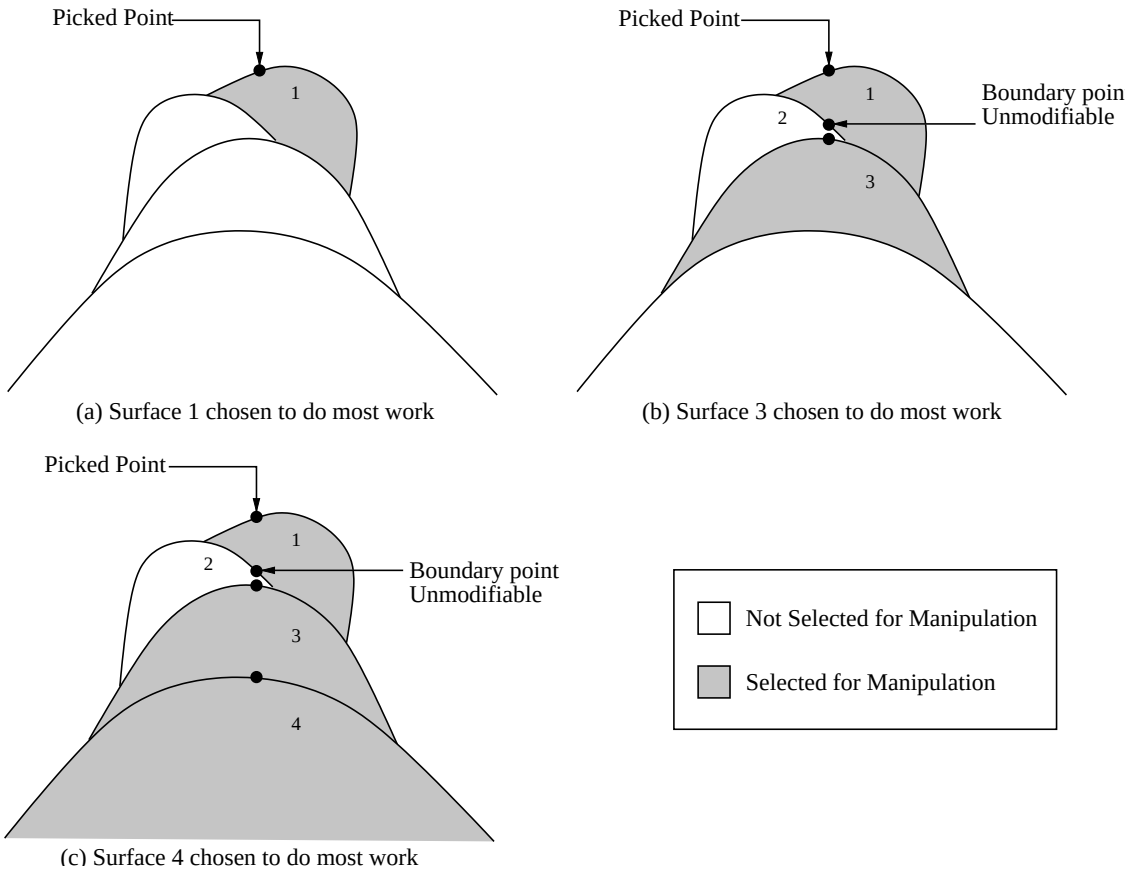


Figure 3.5: Example of choosing a resolution level for manipulation

Figure 3.5 shows an example of a pasting hierarchy of four surfaces, where the user has selected a point on the topmost surface to manipulate. The user needs to determine which surface will do the most work by choosing a resolution level, i.e., he picks the surface in the pasting hierarchy that will do the most work. The user can choose between surfaces 1, 3, or 4, but not 2, since the corresponding picked point lies in its unmodifiable region. In Figure 3.5(a), the topmost surface is chosen to do the most work, as in the top level direct manipulation method. The next lower modifiable surface is selected to do the most work in Figure 3.5(b) and in Figure 3.5(c), the user has chosen the lowest modifiable surface to do the most work, as in the full hierarchy direct manipulation method.

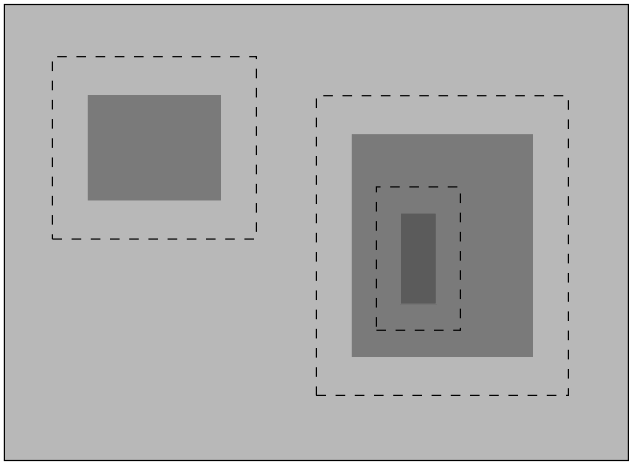


Figure 3.6: Domain space view of surface regions coloured according to resolution levels

It is necessary to give the user feedback concerning the number of different resolution levels at which a selected surface point can be edited. In my direct manipulation editor, I provide a *hierarchical colour mode*, in which the colour of the surfaces indicates the range of resolution at which editing may take place. The darker the shade of grey, the higher the number of surfaces in the pasting hierarchy that can be manipulated.

All surface regions that have the same range of resolution levels are the same colour, as demonstrated in Figure 3.6. For example, the modifiable region of the base might be light grey, as would the unmodifiable boundary region of a surface pasted directly on the base. The interior or

modifiable part of a surface pasted on the base might be a darker grey to indicate that there are two resolution levels (the surface and the base), and surface regions with greater ranges of resolution levels would be correspondingly darker shades of grey. Note that the dashed lines indicate the boundaries of the surfaces and are not seen in the actual direct manipulation interface. The manner in which the colours are assigned is described in Section 4.5.

The advantage of the hierarchical direct manipulation method is that the user has more control when modelling because he chooses the resolution level at which a composite surface will be manipulated. In addition, the colours of the surfaces indicate the valid range of resolution levels and so the user can easily tell which surfaces may be modified by selecting a given point.

Contrast the hierarchical direct manipulation method with that of Forsey's approach to the direct manipulation of hierarchical B-splines, described in Section 2.5.2, in which the upper levels of the hierarchy were stripped out of the display as the user descended the hierarchy. Unfortunately, with these levels removed, the user was not given any feedback about the appearance of the surface as a whole. The user was only able to view the composite surface when the manipulation was finished and the stripped levels were displayed once more.

3.8.1 Convergence of the Correction Factors

The goal of the hierarchical direct manipulation method was to apply all of the desired change to the chosen resolution level. Unfortunately, this was not feasible because the $\vec{d}''_{i,j}$ in Equation 3.1 is not linear in the base surface's control points. Determining the necessary modifications to the surface at the chosen resolution level that would produce the desired change in the composite surface is too difficult. Instead, most of the change is applied to the desired resolution level with successive adjustments made to the surfaces up the pasting hierarchy. Ideally, the magnitude of the correction factors would decrease as the hierarchy of pasted surfaces is ascended, as this would ensure that most of the change is at or near the desired resolution level. In practice, however, the magnitudes of the correction factors do not always decrease as the pasting hierarchy is ascended. After initial testing, I found that when the correction factors increased, it was by a small amount compared to the original $\overrightarrow{\Delta P}$. More research needs to be done to determine exactly why correction

factors sometimes increase and if the amount of increase is always by a small factor. Note that empirical testing did not show the increasing magnitudes of the correction factors to be a problem in the hierarchical direct manipulation technique.

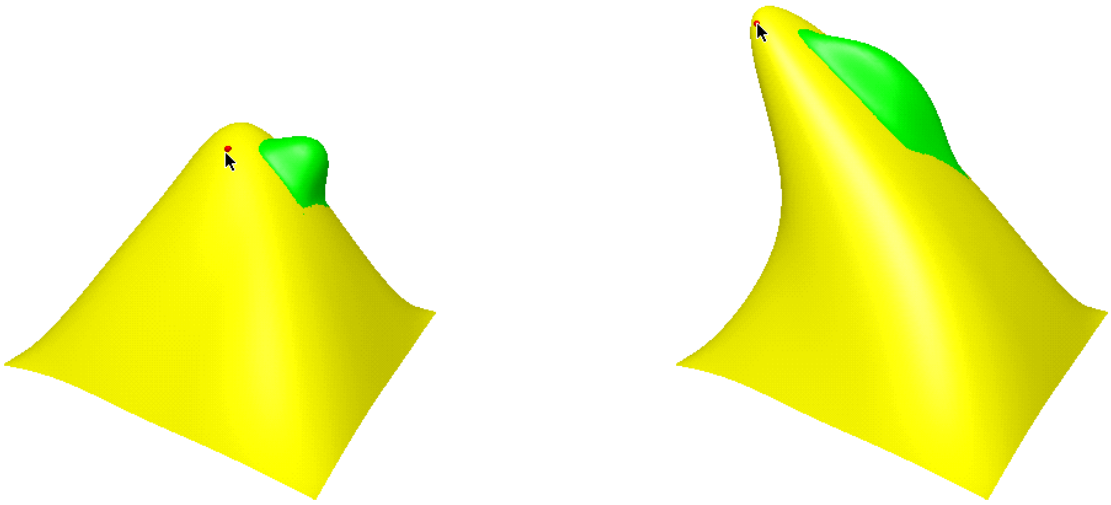
3.9 Results

The following colour plates illustrate the direct manipulation methods discussed earlier. In these plates, the selected point (indicated in red) lies upon the yellow surface.

Examples of the top level direct manipulation technique are shown in the following three figures. Figure 3.7 shows that when a point on the base surface is selected and moved, the feature follows along. In Figure 3.8, directly manipulating the feature leaves the base unchanged. Figure 3.9 shows that if an alternative block of control points is moved when a point too near the feature's boundary is selected, then although that point can be directly manipulated, other parts of the feature distort.

Figure 3.10 shows that if only the lowest level surface in the pasting hierarchy is modified, as in the base only method, direct manipulation is not attainable. The full hierarchy method is demonstrated in Figure 3.11, where the lowest level surface is modified and a correction factor is applied to the feature surface to achieve direct manipulation.

Examples of the hybrid direct manipulation method are shown in the three subsequent figures. In Figure 3.12, the selected point is far from the boundary of the feature so all the work is done by the feature surface. Figure 3.13 shows what happens when an alternative block of control points is found for the selected point, but no work is passed down. Note that the feature surface is distorted, i.e., the picked point is not very close to the point of maximal change. And finally, when the selected point is near the boundary of the surface, some of the work is passed down to the underlying surface as shown in Figure 3.14. The amount of work passed down is determined by the values of the minimum and maximum thresholds. In this example, the minimum threshold is set to 0.1 and the maximum threshold is set to 0.3. Thresholds are described in more detail in Section 3.7. In addition to passing work down, an alternative block of feature control points is

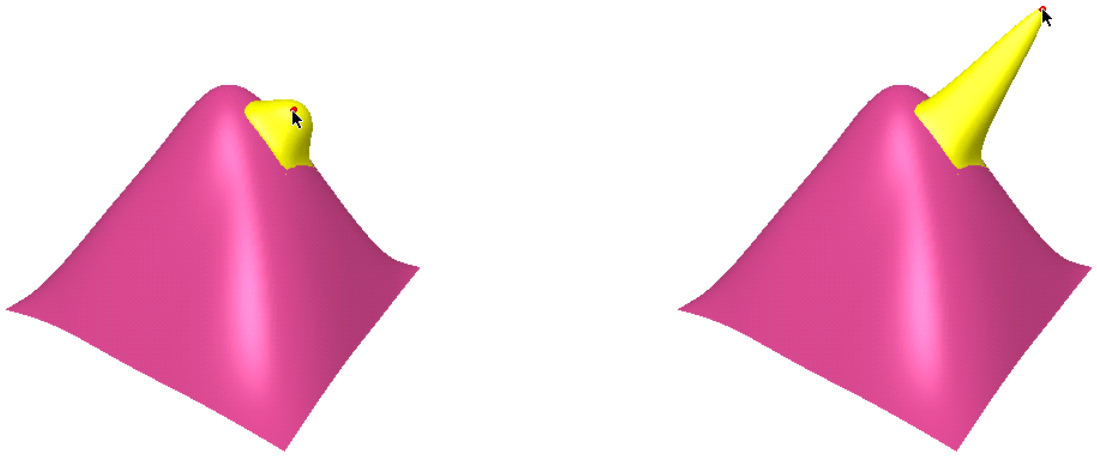


(a) (b)

Figure 3.7: Top level direct manipulation of a base surface

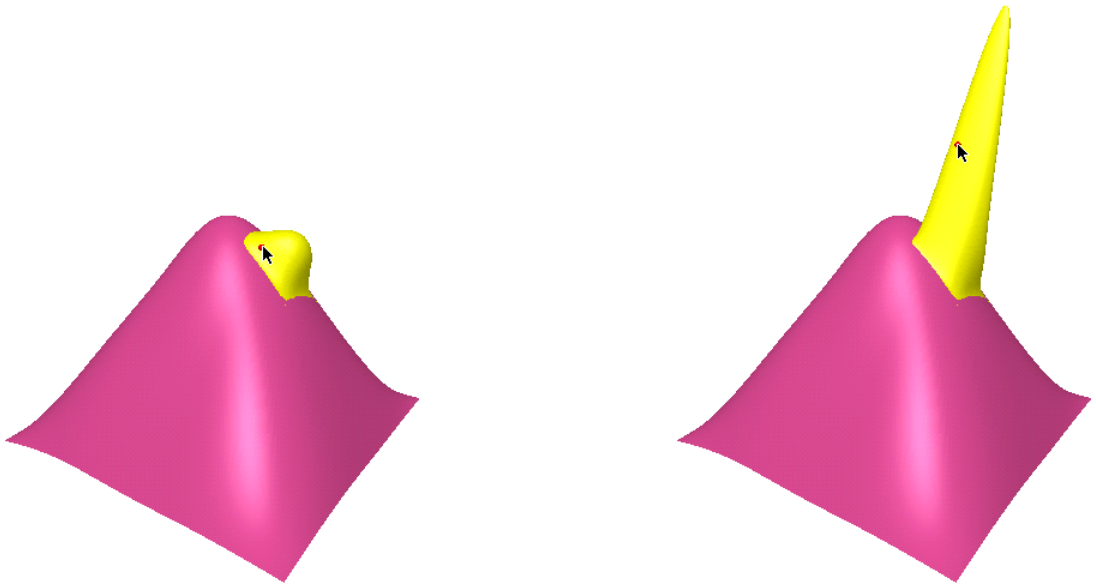
found for the selected point. Note that there is a large distortion in the feature surface and that the base surface has changed slightly.

The five examples in Figure 3.15 illustrate a sequence of events that occurs when performing the hierarchical direct manipulation method on the yellow surface in Figure 3.15(a). The dark blue surface(s) in Figures 3.15(b)–3.15(d) are affected at each stage of the manipulation. In each case, the bottommost blue surface undergoes the greatest change, while the other blue surfaces only have correction factors applied to them. The more surfaces that are coloured dark blue, the broader the change in the composite surface. In Figure 3.15(b), only the topmost surface is affected and the selected point was moved to the right and down. In Figure 3.15(c), two surfaces are affected and the selected point was moved up and to the right. Three surfaces are affected and the selected point is moved to the right and up in Figure 3.15(d). Figure 3.15(e) shows the result of the manipulation.



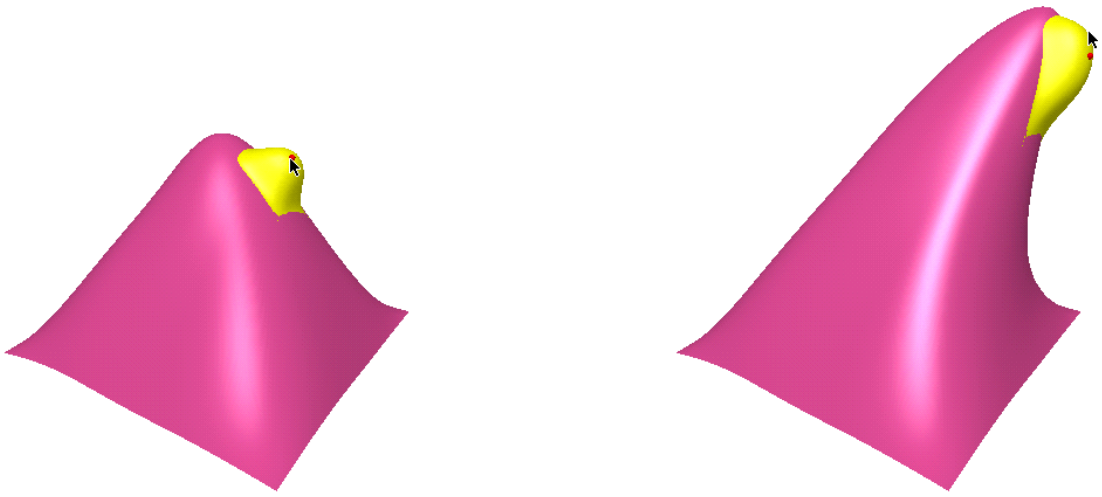
(a) (b)

Figure 3.8: Top level direct manipulation of a feature surface



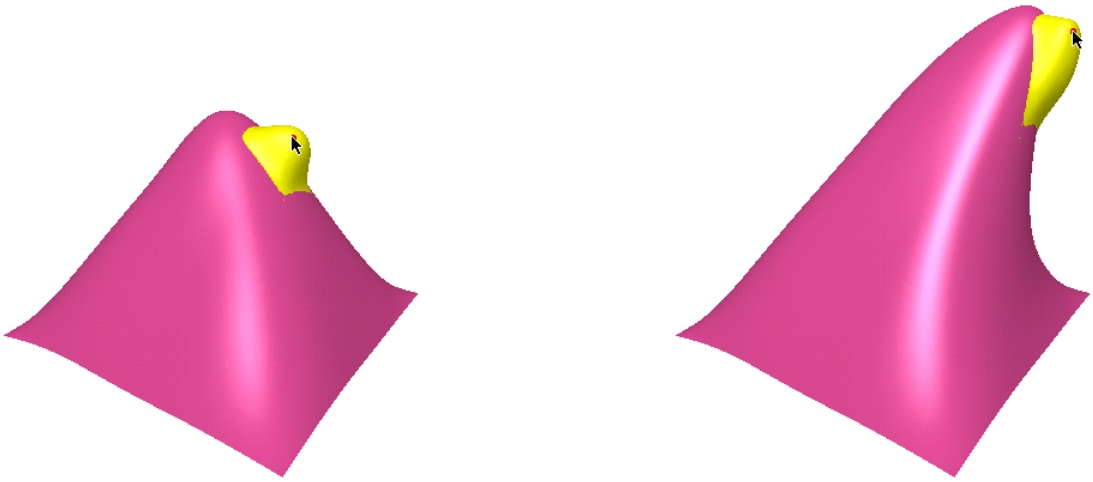
(a) (b)

Figure 3.9: Distortion of a feature surface caused by choosing alternative control points



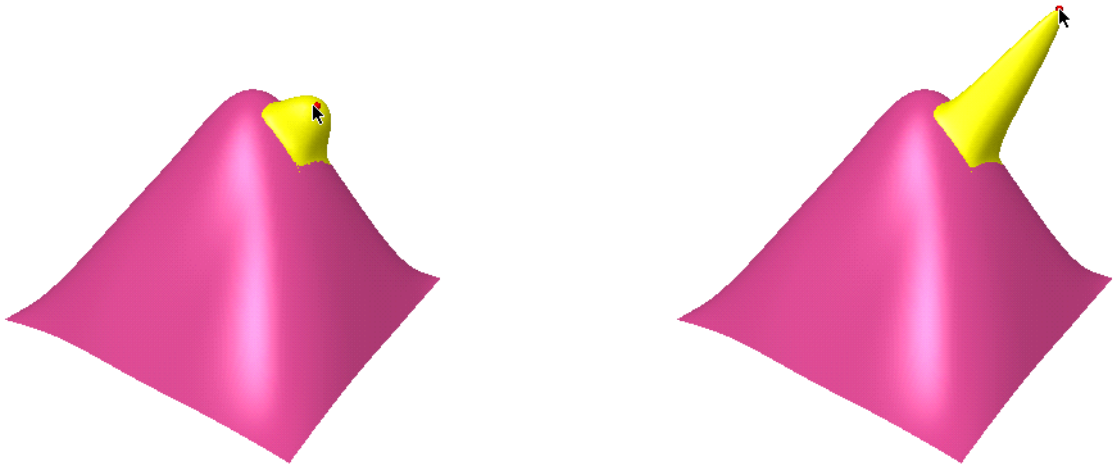
(a) (b)

Figure 3.10: Example of the base only direct manipulation method



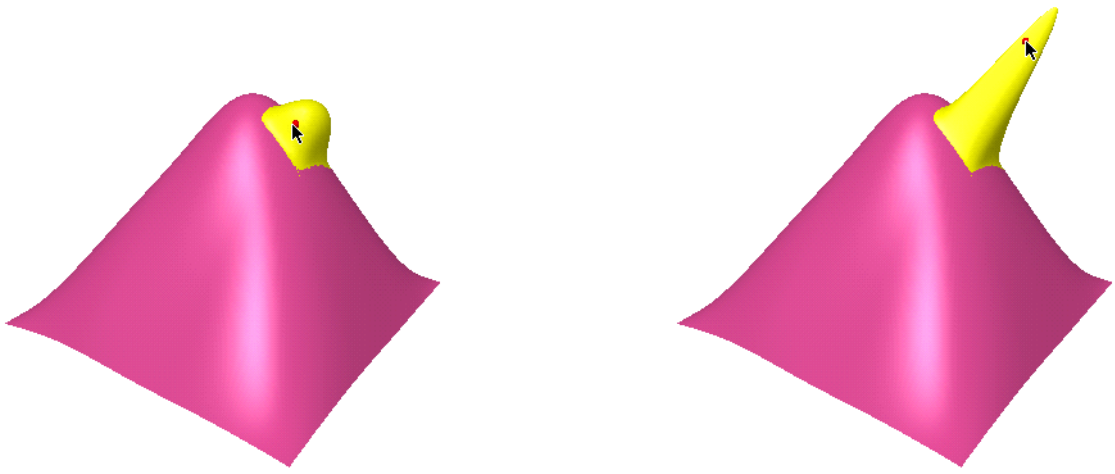
(a) (b)

Figure 3.11: Example of the full hierarchy direct manipulation method



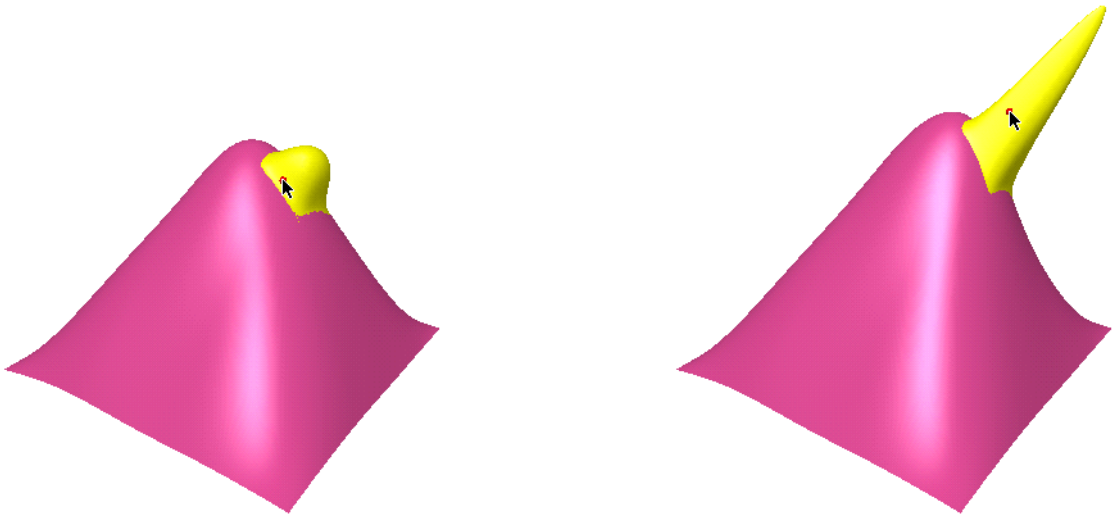
(a) (b)

Figure 3.12: Hybrid example with manipulation far from the boundary



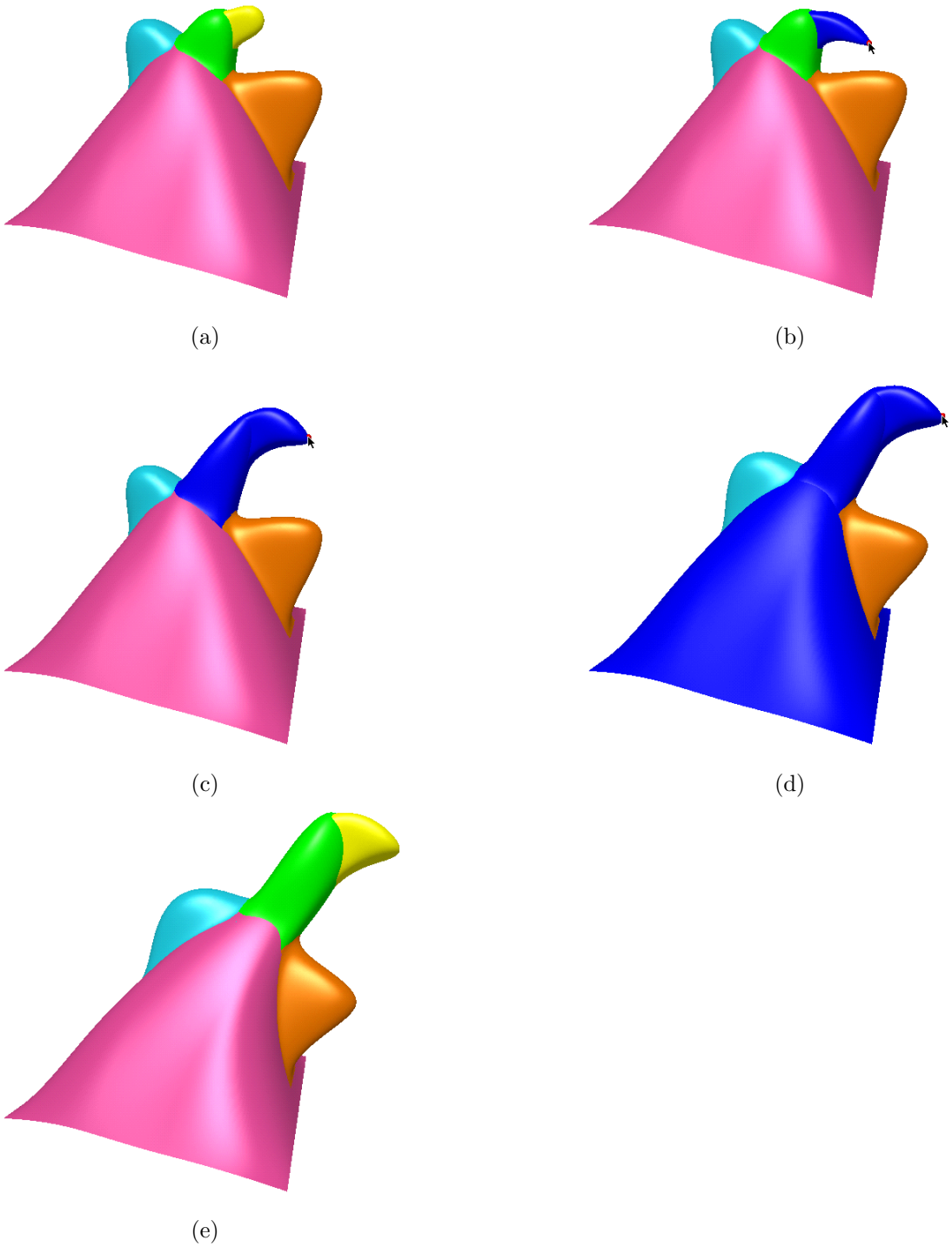
(a) (b)

Figure 3.13: Hybrid example with manipulation close to the boundary



(a) (b)

Figure 3.14: Hybrid example with manipulation very close to the boundary



(a) (b) (c) (d) (e)

Figure 3.15: Example of the hierarchical direct manipulation method

Chapter 4

Implementation Details

In this chapter I give an overview of some implementation details of my direct manipulation editor. I begin with a description of my editor, explaining how to run my program and the available menu options. Section 4.2 contains a description of the file format that my editor uses to store pasted surfaces. Then, in Section 4.3, I describe the OpenGL constructs I use to represent surfaces in the pasting hierarchy. Next, I explain the calculations involved in the selection and movement of the picked point. In Section 4.5, I discuss how I use colour to display resolution levels for the hierarchical direct manipulation method. Finally, I describe the interface for selecting the resolution level of the manipulation.

4.1 Description of User Interface

My editor allows a user to model surfaces by hierarchically pasting non-uniform tensor product B-spline surfaces using a world space user interface. At present, only rectangular, unrotated domains are allowed. A number of operations can be performed on pasted surfaces using my editor. Standard operations such as *transform scene* can be applied to a hierarchy of pasted surfaces, i.e., the scene can be translated, zoomed in and out, and rotated. Other operations that can be performed on pasted surfaces include *paste*, *unpaste*, *refine*, *translate*, and *directly*

manipulate, the last two of which each contain five different methods.

As a project for the computer science course CS788, I investigated several methods for translating pasted features. The goal was to give the user the feeling of sliding a feature across a base surface. The basic idea is to transform the mouse motion vector on the screen into a translation vector in the base's domain. Initially, I implemented and experimented with Chan's projective-translation method [CMB97]. In my editor, his original method is called *Chan1* and his improved method is called *Chan2*. The Chan2 method introduced additional problems which I attempted to fix by developing three new possible translation methods, *Mann1*, *Mann2*, and *Bartels*. See Appendix A for details.

To run my program, type `./Paste UI.gr`. Two windows will appear; the large window in Figure 4.1 is my main editor where the user can select operations to be applied to surfaces in the world space view. The buttons that appear on the right side of the world space window are shortcuts to the indicated menu options. The full list of available menu options, with descriptions, is presented in Table 4.1. The smaller window in Figure 4.2 shows the domain space view of the composite surface and does not allow user interaction. It is there for debugging purposes and to provide the user with additional information.

The following items need further explanation:

Translate Feature There are five methods available for translating a feature: Chan's projective-translation (*Chan1*)[CMB97], an improved projective-translation (*Chan2*)[CMB97], Stephen Mann's suggested method (*Mann1*), Richard Bartels' suggested method (*Bartels*), and an improvement to *Mann1* (*Mann2*).

Transform Scene The left mouse button allows the user to translate the composite surface in the x and y directions. The middle mouse button lets the user zoom in and out of the composite surface in the z direction. Lastly, the right mouse button gives the user the ability to rotate the composite surface like a trackball.

Refine Surface The *refine* option allows the user to increase the control point density of the active surface, thereby allowing the user to manipulate the surface at a finer level of detail.

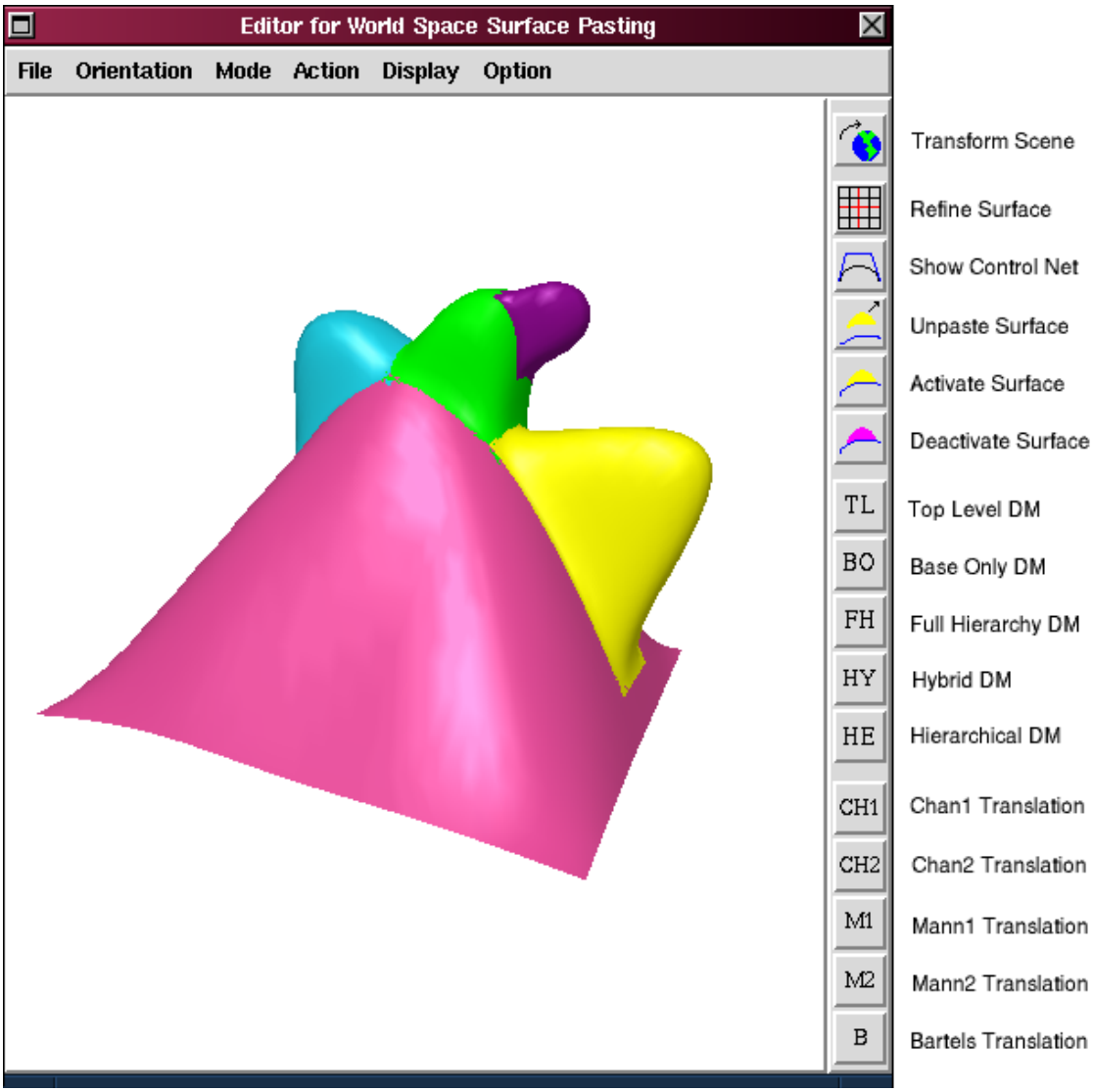


Figure 4.1: World space view

Menu Option	**Operation**	**Description**
File	Clear	clears the screen
	Load	loads a base, detail, or a composite surface
	Save	saves a composite surface
	Quit	quits the program
Orientation	Reset	resets the view of the composite surface to its original orientation
	Set	saves the current orientation of the composite surface
	Load	loads the saved orientation
Mode	Translate Feature$\star$	translates an active feature in the world space using either Chan1, Chan2, Mann1, Mann2, or Bartels methods
	Direct Manipulation	allows the user to directly manipulate the active surface using either the Top Level, Base Only, Full Hierarchy, Hybrid, or Hierarchical direct manipulation methods
	Transform Scene$\star$	transforms the scene, i.e. translate in the x and y directions, zoom in and out, and rotate
Action	Activate Surface	sets a surface active by colouring it yellow
	Deactivate Surface	deactivates the active surface
	Refine Surface$\star$	refines the active surface
	Unpaste Surface	unpastes the active surface
Display	Control Net	displays the control net of the active surface
	Colour Mode$\star$	sets the colour mode for the composite surface to be either Uniform, True Colour, or Hierarchical
Option	Hybrid Threshold$\star$	sets the threshold values for the hybrid direct manipulation method

Table 4.1: Available menu options in my editor. Items marked by a $\star$ are described in more detail in Section 4.1.

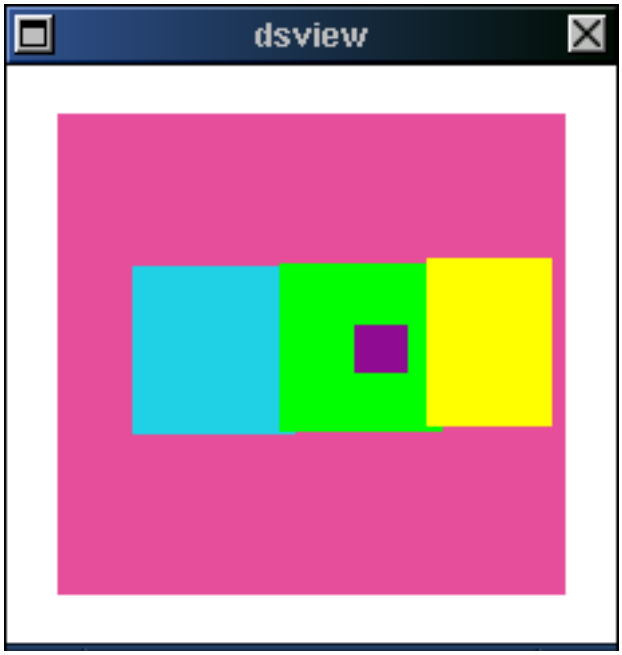


Figure 4.2: Domain space view

The refinement process works by splitting each of the domain intervals in half for each of the two parametric domain directions, thus quadrupling the number of subpatches which form the surface. The surface is refined by placing new knots halfway between each of the knots in the area over which the surface is defined. For example, if a surface had knot structure

$$\{u_0, \ldots, u_0, u_1, u_2, \ldots, u_{k-1}, u_k, \ldots, u_k\}$$

in the u direction, the refined surface would have knot structure

$$\{u_0, \ldots, u_0, \frac{u_0 + u_1}{2}, u_1, \frac{u_1 + u_2}{2}, u_2, \ldots, u_{k-1}, \frac{u_{k-1} + u_k}{2}, u_k, \ldots, u_k\}$$

and a similar transformation is applied in the v direction. After the refined surface is constructed, new displacement vectors are calculated for each of its control points as described in Section 3.3.

Colour Mode – This option allows the user to choose between three colour modes for the composite surface. If *true colour* is selected, surfaces are drawn with their user-defined colour. *Uniform* means the composite surface will be painted a light steel blue. If *hierarchical* is selected, the user is given information on the number of different resolutions levels at which a selected surface point can be edited. The surfaces are coloured from light steel blue to correspondingly darker shades of steel blue. The higher the number of resolution levels, the darker the area of the surface. See Section 3.8 for details.

Hybrid Threshold This option allows the user to select minimum and maximum threshold values for the hybrid direct manipulation method as described in Section 3.7. The values for both thresholds are clamped to $[0, 1]$.

4.2 Data File Format

In this section, I explain the file format read and written by my pasting editor. An example of a data file that describes a feature is shown in Figure 4.3. Note that the "→" characters and all text to the right of them are not part of the data file; they serve as comments only. A base surface is described in the same way, but the values in the last line that specify the image of its domain are ignored because there is no need to map the base domain. The data file for a composite surface contains a sequence of surface descriptions, $B, F_1, F_2, \ldots, F_k$. The composite surface can be reconstructed by taking B to be the base and hierarchically pasting surface F_i onto the composite surface formed from $B, F_1, \ldots, F_{i-1}$, for $i = 1, \ldots, k$.

4.3 OpenGL's NURBS Surfaces

I use OpenGL's NURBS (Non-Uniform Rational B-spline) surfaces [WND97] to render the pasted surfaces in my direct manipulation editor. In the following sections, I refer to several features of OpenGL's NURBS surfaces that I use in the implementation of my editor, so I give a brief description of the process of creating and rendering a NURBS object here.

```
3 3                                  → degree in u and v directions
7 7                                  → number of control points in u and v directions
0 0 0 0.75 1.5 2.25 3 3 3            → knots in u direction
0 0 0 0.75 1.5 2.25 3 3 3            → knots in v direction
0 0 0                                → control points as x y z, grouped by column
0 0.25 0
0 0.75 0
0 1.5 0
0 2.25 0
0 2.75 0
0 3 0
0.25 0 0                             → column 2 control points begin here
0.25 0.25 0
0.25 0.75 0
0.25 1.5 0
0.25 2.25 0
0.25 2.75 0
0.25 3 0
0.75 0 0                             → column 3 control points begin here
0.75 0.25 0
0.75 0.75 0.390625
0.75 1.5 0.507812
0.75 2.25 0.390625
0.75 2.75 0
0.75 3 0
1.5 0 0                              → column 4 control points begin here
1.5 0.25 0
1.5 0.75 0.507812
1.5 1.5 0.660156
1.5 2.25 0.507812
1.5 2.75 0
1.5 3 0
2.25 0 0                             → column 5 control points begin here
2.25 0.25 0
2.25 0.75 0.390625
2.25 1.5 0.507812
2.25 2.25 0.390625
2.25 2.75 0
2.25 3 0
2.75 0 0                             → column 6 control points begin here
2.75 0.25 0
2.75 0.75 0
2.75 1.5 0
2.75 2.25 0
2.75 2.75 0
2.75 3 0
3 0 0                                → column 7 control points begin here
3 0.25 0
3 0.75 0
3 1.5 0
3 2.25 0
3 2.75 0
3 3 0
0 0 1                                → colour as r g b in the range [0, 1]
0.6 0.4 0.8 0.4 0.8 0.7 0.6 0.7      → image of feature domain in base domain, as four
                                       u v points counterclockwise from the bottom left,
                                       assuming the base domain is [0, 1] × [0, 1]
```

Figure 4.3: Sample data file with comments

The first step is to create a NURBS object using `gluNewNurbsRenderer`. Next, rendering properties for the NURBS object are set using zero or more calls to the `gluNurbsProperty` function. Examples of these properties are the *sampling tolerance* and the *display mode*. NURBS surfaces are rendered by drawing polygons, each of whose edges is no longer than the sampling tolerance measured in pixels. The display mode determines how the surfaces should be rendered, either as filled polygons or wireframe.

```
void gluNurbsSurface(GLUnurbsObj* nobj,
                     GLint        u_knot_count,
                     GLfloat*     u_knots,
                     GLint        v_knot_count,
                     GLfloat*     v_knots,
                     GLint        u_stride,
                     GLint        v_stride,
                     GLfloat*     ctlarray,
                     GLint        u_order,
                     GLint        v_order,
                     GLenum       type);
```

Figure 4.4: The gluNurbsSurface function

Once the NURBS properties have been set, `gluBeginSurface` is called to begin the rendering of the NURBS object. Then, one or more calls are made to `gluNurbsSurface` to generate and render the NURBS object. The `gluNurbsSurface` function declaration appears in Figure 4.4. Note that OpenGL requires the first and last knots in the knot vectors to be repeated an additional time, so the knot vectors passed to `gluNurbsSurface` each have two additional values. A brief explanation of the arguments to this function follows:

nobj - a pointer to the NURBS object created by `gluNewNurbsRenderer`

u_knot_count - the number of knots in the u direction

u_knots - a list of the knots in the u direction

v_knot_count - the number of knots in the v direction

v_knots - a list of the knots in the v direction

u_stride - the number of floating-point values between successive **ctlarray** entries in the u direction

v_stride - the number of floating-point values between successive **ctlarray** entries in the v direction

ctlarray - the $(u\_knot\_count - u\_order) \times (v\_knot\_count - v\_order)$ array of control points that define the position, normal, colour, or texture coordinate of each surface point. These control points depend on the **type** parameter listed below.

u_order - the order (one more than the degree) of the surface in the u direction

v_order - the order (one more than the degree) of the surface in the v direction

type - the type of data being generated. In my direct manipulation pasting editor, I used three data types: GL_MAP2_VERTEX_3, GL_MAP2_TEXTURE_COORD_2, and GL_MAP2_COLOR_4. The MAP2 indicates the use of a two dimensional evaluator type, VERTEX_3 generates three spatial coordinates for each vertex, TEXTURE_COORD_2 assigns two dimensional texture coordinates, and COLOR_4 generates RGBA[1] values.

Finally, `gluEndSurface` is called to conclude the definition of the NURBS object.

Once the NURBS object is not needed anymore, `gluDeleteNurbsRenderer` is called to release the system resources used by the object.

4.4 Selecting a Point for Manipulation

In this section, I discuss how I implemented the selection and subsequent movement of a surface point. I first explain how to choose the active surface in Section 4.4.1. Then, I discuss how to find the user-selected point for manipulation on the active surface in Section 4.4.2 and finally in Section 4.4.3, I describe how to make the selected point follow the mouse cursor.

[1]The A stands for transparency. All the surfaces in my editor are fully opaque so A is always 1.

4.4.1 Choosing the Active Surface

Before a user can pick a point for manipulation, he must select a surface in the pasting hierarchy to be the active surface. Once the user has clicked the left mouse button, the selection procedure begins. I used OpenGL's *selection mode* [WND97] to determine the active surface the user has chosen. In this mode, the frame buffer is not updated but any object drawn in a certain region generates a *hit*. I specify this region by restricting the viewing volume to a small area around the mouse cursor. Next, all surfaces are rendered and assigned a unique identifier. OpenGL then returns a *hit list*, a list of surfaces that intersect the restricted region around the mouse cursor. The hit list is processed and the hit with the lowest depth value, the one closest to the viewer, is returned. The unique identifier associated with the hit record indicates which surface the user has chosen to be the active surface.

4.4.2 Finding the Picked Point

Before a user can directly manipulate a pasted surface, he must select a point on the active surface for manipulation. The user selects the point by clicking the left mouse button anywhere on the surface. In this section, I explain how the mouse click is converted into a surface point.

The idea is to draw the active surface in the back buffer, assigning a unique colour to every surface point.[2] When the user clicks the left mouse button, I find the colour of the point under the cursor and then convert the colour into the domain value corresponding to the surface point. Others, such as Levoy and Hanrahan [LH96] and Hanrahan and Haeberli [HH90], have also used colours or textures to invert transformations.

The first step is to assign a mapping between domain values and colour. A colour has red, green, and blue components that each range from 0 to 1 inclusive; I vary red and green to distinguish between domain values. The blue component is set to a constant value β that is different from the blue component of the background colour. Thus, the colour of the point under the mouse cursor clearly indicates whether or not the user has clicked on the active surface. The red and green components are set in the following manner. Suppose the surface domain is

[2]Thanks to Tim Lahey and Blair Conrad for suggesting this method.

$[a_u, b_u] \times [a_v, b_v]$. A surface point $S(u, v)$ is assigned the colour $C(u, v)$, where

$$C(u, v) = \left(\frac{u - a_u}{b_u - a_u}, \frac{v - a_v}{b_v - a_v}, \beta \right). \tag{4.1}$$

Figure 4.5 shows a top-down view and an oblique view of a surface coloured as described above.

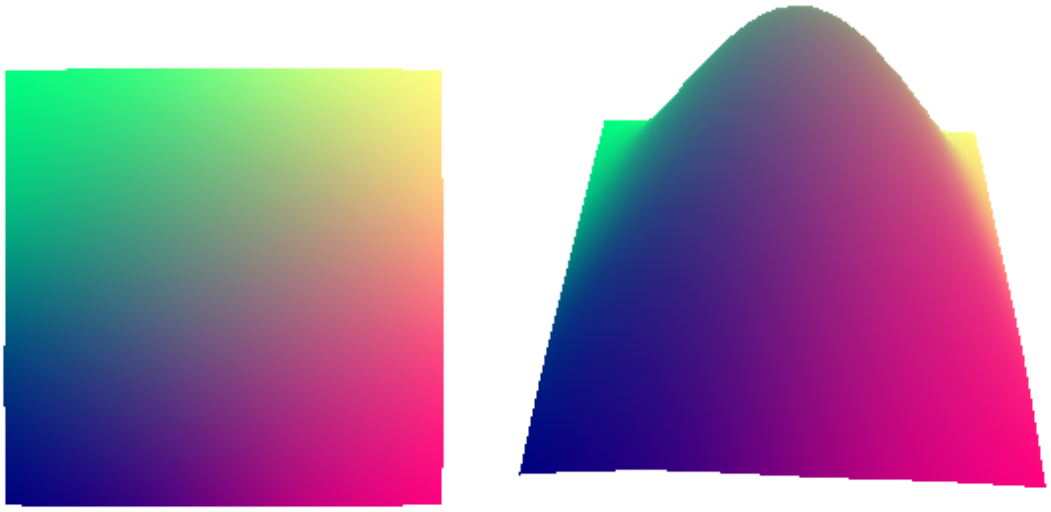


Figure 4.5: A mapping between domain values and colour

The OpenGL NURBS construct allows a colour to be assigned to each control point, not each surface point. When OpenGL renders the NURBS surface, it interpolates the colours of the control points to determine the colour for a given surface point. For surface point $S(u, v)$ to have colour $C(u, v)$, it is necessary to assign control point $P_{i,j}$ the colour $C(\gamma_{i,j})$, where $\gamma_{i,j}$ is the $(i, j)^{\text{th}}$ Greville point. Control point colours are assigned by calling the `gluNurbsSurface` function with *type* set to GL_MAP2_COLOR_4. For a detailed description of NURBS surfaces and the `gluNurbsSurface` function, see Section 4.3.

When the user clicks the left mouse button, the active surface is drawn in the back buffer using the colours described above. OpenGL's lighting is disabled when the surface is drawn so the true surface colours are rendered. Next, the colour of the pixel under the mouse cursor, say (R, G, B), is obtained and if $B \neq \beta$, then the user did not select a point on the active surface.

Otherwise, Equation 4.1 is used to determine the point's domain value from its colour:

$$\begin{aligned} u &= R(b_u - a_u) + a_u \\ v &= G(b_v - a_v) + a_v. \end{aligned}$$

The effectiveness of using colours to determine the picked point depends on the colour depth of the display. For example, this technique works well when 8 bits are allocated to each colour component. However, when I used the picking procedure on an *SGI Octane* that had 5 bits for each colour component, the resolution was too coarse and so the selected point was offset from the mouse cursor. It is possible to overcome the limitations of a low colour display by performing additional picking steps. The first application of the technique would provide a reduced area of the surface that contained the selected point. The full colour range could then be applied to this reduced area to determine a closer approximation.

Note that as an alternative interface, the blue colour component could be used to select a surface rather than using OpenGL's selection mode as described in Section 4.4.1. Each surface in the pasting hierarchy can be assigned a unique β value that is different from the blue component of the background colour. Then the colour under the mouse cursor indicates the selected surface via the blue component and the domain value via the red and green components. This technique will only work when the blue component of the frame buffer has enough bits to support the number of surfaces in the hierarchy. If there are too many surfaces, it will be necessary to perform multiple passes to positively identify the selected surface.

4.4.3 Making the Picked Point Follow the Mouse Cursor

In this section, I describe how I made the user-selected picked point follow the mouse cursor. Before I explain this method, I will give a brief explanation of the view and model spaces. The model space is an arbitrary coordinate system and the position of each object in the scene is specified relative to this model space. The view space consists of a coordinate system based on the viewer. The viewer's eye is assumed to be at $(0, 0, 0)$, looking in direction $(0, 0, -1)$, with

$(0, 1, 0)$ as the up vector.

Figure 4.6 shows the top-down view of the viewing volume. P_v represents the picked point in view space and pos_1 represents the picked point in screen coordinates, while N_v represents the new location of the picked point in view space and pos_2 represents the new location in screen coordinates. The screen size is $width \times height$ and fov is the field of view. The distance to the picked point in the z direction in view space is $dist_2$ and the distance to the picked point on the screen is $dist_1$.

The picking procedure described in the previous section returns the domain value of the picked point. The position of the picked point in model space P_m is obtained by evaluating the surface at this domain value. The modelview matrix M describes the transformation from the model space to the view space, so the position of the picked point in view space is $P_v = MP_m$ and the position of the point's new location in view space is $N_v = MN_m$.

I want to determine the amount P_m needs to move in model space to have its image in the view space P_v move to N_v:

$$\overrightarrow{\Delta P}_m = N_m - P_m = M^{-1}N_v - P_m. \tag{4.2}$$

The values M, M^{-1}, and P_m are known. The only unknown is N_v.

I describe the procedure performed on the x-coordinate of N_v using the triangle outlined in bold in Figure 4.6. The values $fov/2$ and $width/2$ are known. From the triangle, we see that:

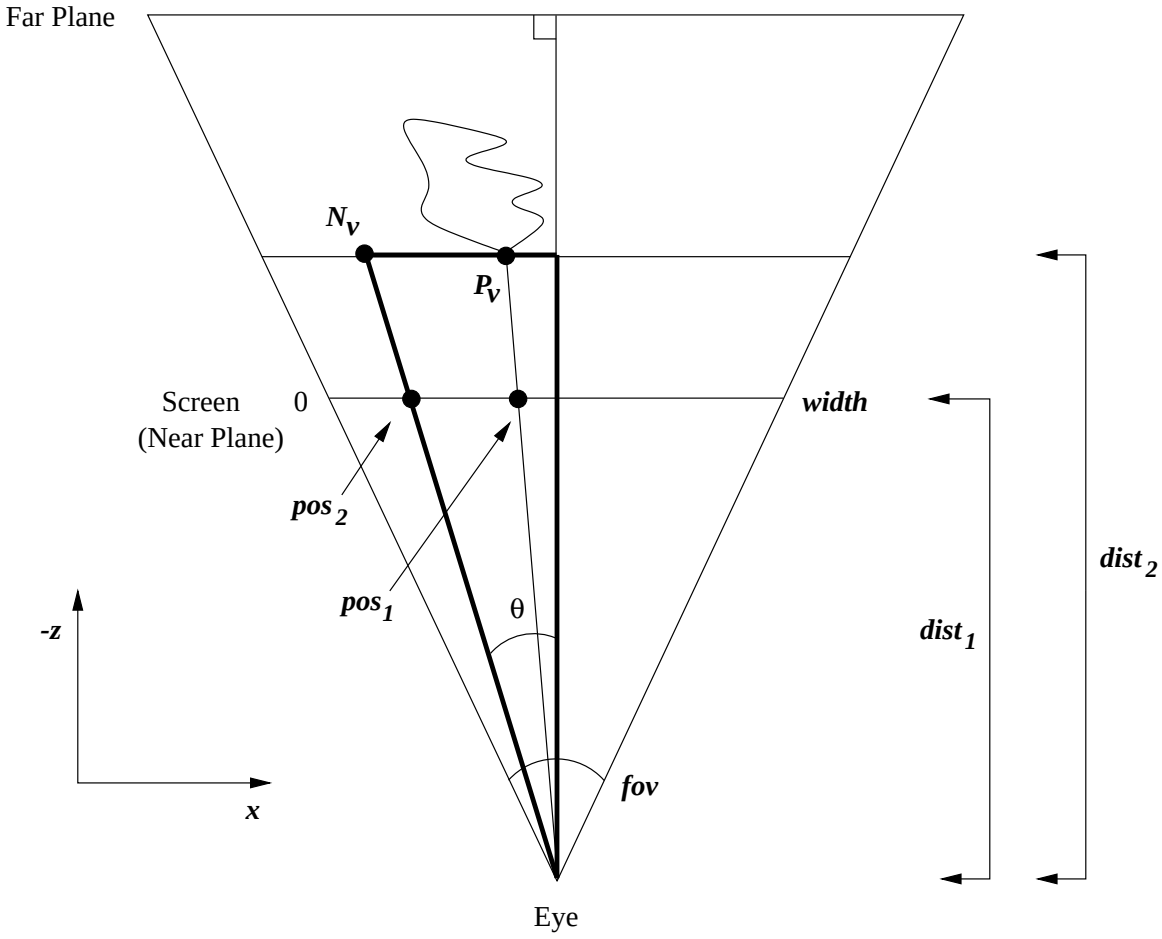
$$N_v.x = \tan(\theta.x) \cdot dist_2.$$

The variable $dist_2$ is $-P_v.z$. The value of $\tan\theta.x$ can be found from:

$$\tan(\theta.x) = \frac{pos_2.x}{dist_1}$$

where

$$dist_1 = \frac{width}{2\tan(fov/2)}.$$



P_v	- picked point in view space
N_v	- new position of picked point in view space
pos_1	- screen position of P_v
pos_2	- screen position of N_v
$dist_1$	- distance in z direction to picked point on screen
$dist_2$	- distance in z direction to picked point
fov	- field of view

Figure 4.6: Top-down view of viewing volume

A similar procedure is performed for $N_v.y$ and by design $N_v.z$ is $P_v.z$. Once N_v is known, $\overrightarrow{\Delta P}_m$ can be calculated using Equation 4.2.

4.5 Using Colour to Display Resolution Levels

I use OpenGL texture maps to assign the colours that indicate the resolution levels in the hierarchical direct manipulation method. A texture map consists of an image that can be applied to a polygon to give it the appearance of a more detailed surface. The first step is to define a texture image and the second is to define texture coordinates that map the image onto the composite surface.

4.5.1 Defining the Texture Image

The goal is to create a texture image that can be used to change the colour intensity on a hierarchy of pasted surfaces. After experimenting with different texture image sizes, I chose to use a two dimensional texture image of size 128×128, where each element is a luminance value. A luminance value ranges from 0 to 1 inclusive and represents a greyscale value; 0 representing black and 1 representing white. The texture image is used to modulate surface colours. Each luminance value is multiplied by the original surface colour, so areas of the surface whose corresponding texture areas have low values will appear dark. Likewise, surface areas whose corresponding texture areas have high values will appear lighter.

First, I created an array of luminance values to represent the texture image with each value initialized to 1. The following procedure is applied to every surface in the pasting hierarchy. The modifiable region of the surface is found and then mapped into the array that represents the texture image. The value of each element in the array that is covered by the image of the modifiable region is reduced by a fixed amount. The end result is that the areas of the texture image that correspond to totally unmodifiable regions of a surface will remain white whereas the image areas corresponding to a hierarchy of modifiable regions will be progressively darker depending on the number of modifiable regions.

4.5.2 Applying the Texture Image

The texture image is applied to both the domains in the domain space view and surfaces in the world space view when the user has selected the hierarchical colour mode. Every domain and surface point must be assigned a two dimensional texture coordinate, the components of which range from 0 to 1 inclusive. The corners of the base surface and domain will have texture coordinates $(0, 0)$, $(0, 1)$, $(1, 1)$, and $(1, 0)$. It is not necessary to specify texture coordinates explicitly for every surface or domain point. When using OpenGL textures, texture coordinates are specified for a few key surface or domain points, like polygon corners and NURBS surface control points, and OpenGL interpolates these values to obtain texture coordinates for intermediate points. To generate texture coordinates for a NURBS surface, I called the `gluNurbsSurface` function with *type* set to GL_MAP2_TEXTURE_COORD_2. For a detailed description of OpenGL's NURBS surfaces and the `gluNurbsSurface` function, see Section 4.3.

Applying the texture map to the domains is straightforward. Texture coordinates are assigned to the corners of each of the surfaces' domains by first mapping the corner points into the base domain and then remapping the points into the $[0, 1]$ square. The position of each corner point in this square is used as its texture coordinate.

Applying the texture map to the world space surfaces is slightly more complicated than the procedure for the domain space surfaces. The polygons that make up the surfaces are indirectly specified using OpenGL NURBS surfaces. It is necessary to calculate appropriate texture coordinates for each of the control points of the surfaces, since the control points define the surface.

Texture coordinates for a surface point should vary linearly with the point's domain values. Otherwise, the texture map will not accurately represent the resolution levels of the pasting hierarchy. This linear dependence is accomplished by assigning to each control point a texture coordinate based on the control point's Greville point. The surface's domain is mapped into the base domain and the location of the mapped Greville point is found. The base domain is subsequently mapped into a $[0, 1]$ square and the position of the Greville point in this square is taken to be the texture coordinate of the control point.

4.6 Selecting the Resolution Level

As I mentioned in Section 3.8, when the user has invoked the hierarchical direct manipulation method, a selected surface point can be manipulated at one of several possible levels of resolution. Once the user selects a point for manipulation, he can cycle through the valid levels of resolution. When the user depresses the left mouse button, the topmost surface under the mouse cursor is highlighted. With the left mouse button depressed, and clicking (depressing and releasing) the right mouse button, the user can cycle through the other possible resolution levels.

An example of choosing a resolution level is illustrated using Figure 3.5. Figure 3.5(a) shows the initial configuration when the user has depressed the left mouse button; surface 1 is highlighted. In Figure 3.5(b), the user has clicked the right mouse button without releasing the left, causing surfaces 1 and 3 to be highlighted. The last example shows the result of right clicking one more time; surfaces 1, 3, and 4 are highlighted. Clicking the right mouse button once more brings the user back to the state in Figure 3.5(a). The more surfaces that are highlighted, the broader the change caused by the direct manipulation. Once the user is satisfied with the chosen resolution level, he can manipulate the surface by dragging the mouse without releasing the left mouse button.

Chapter 5

Conclusion

5.1 Summary

Surface pasting is a technique for adding local detail to surfaces to create complex composite surfaces. Being able to manipulate pasted surfaces directly would greatly aid the modelling process. I have investigated several techniques for directly manipulating pasted surfaces, the last of which (hierarchical direct manipulation) proved to be the best. The hierarchical direct manipulation technique for pasted surfaces is the only method that gives the user a flexible interface for manipulating hierarchical pasted surfaces. The user is able to pick any point on a surface in the pasting hierarchy, decide the resolution level of the manipulation, and move the selected point to a new location. The flexibility offered by this method gives the user complete control over the granularity of change made to the composite surface.

5.2 Case Study: Loch Ness Monster

I have created the Loch Ness Monster model in Figure 5.1 using the hierarchical direct manipulation method in my pasting editor. The water surface acts as the base of the pasting hierarchy. Some waves have been added to the water using direct manipulation while others have been added

by first pasting a feature surface and then directly manipulating the feature to change its shape.

The Loch Ness Monster emerges from the water in two sections. The tail section of the Loch Ness Monster is composed of two hierarchically pasted surfaces. Direct manipulation was applied to the top surface of the tail to increase its length. Then the resolution level was decreased to include the base of the tail, and the entire tail was shaped.

The construction of the Loch Ness body was more involved than that of the tail. The large green surface acts as the main portion of the body and has three different coloured bumps pasted on. Next, the neck, head, and snout were added to the body. The neck was pasted on the body and then the head was pasted on the neck. As is often the case when a feature is pasted over a region of high curvature, the head ballooned out, becoming much wider than I desired. I used direct manipulation to elongate and slim the head. Then the snout was pasted on the front of the head and shaped using the highest resolution level of hierarchical direct manipulation. I then adjusted the resolution level to include the head and neck so that I could pull the head forward from the body. Finally, I pasted the eyes and nostrils on, but they did not require any direct manipulation.

I made the following observations during the construction of my Loch Ness Monster.

- The editor would have been easier to use if there had been an undo button and if non-rectangular domains were allowed. Chan [CMB97] provides a more feature-rich interface that makes modelling more efficient.
- The translation methods were oversensitive when sliding features whose domains were mapped to very small regions of the base.
- Overall, the direct manipulation method worked fairly well.
  - Hierarchical direct manipulation performed best when operating on a multilevel hierarchy where it was possible to vary interactively the depth at which the manipulation occurred.
  - When directly manipulating a single surface, the granularity of change is defined solely by the density of the control points, and it was sometimes difficult to find an appropriate

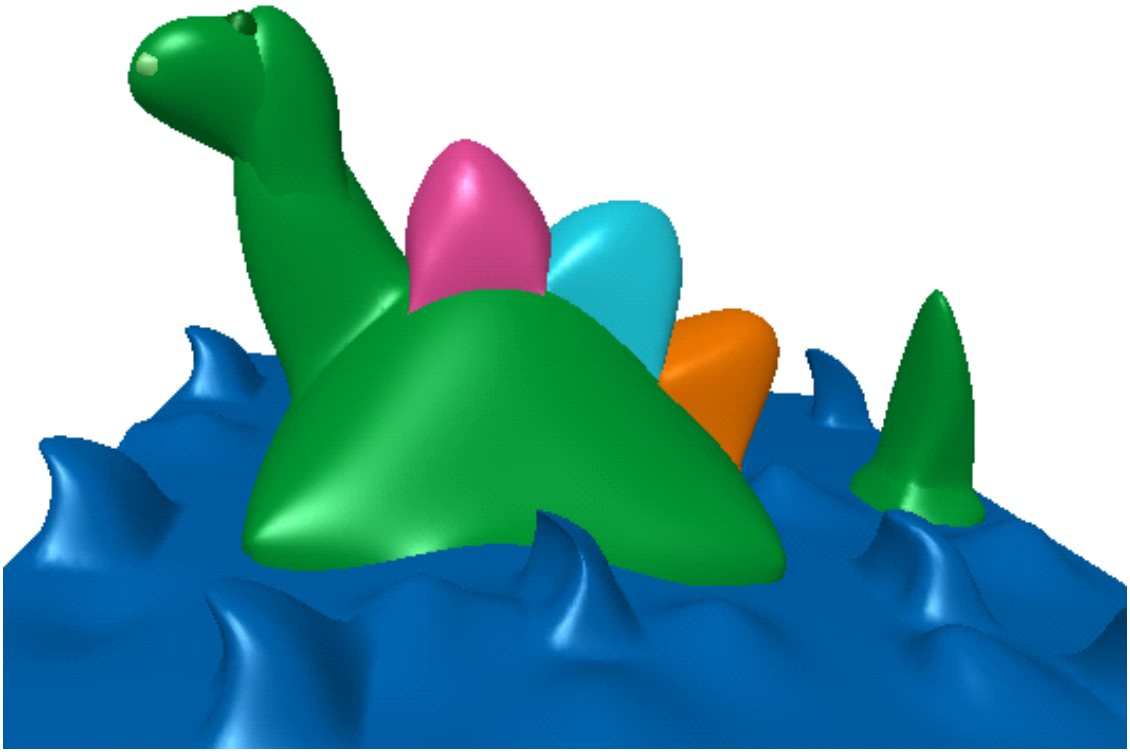


Figure 5.1: Loch Ness Monster

density. Part of this problem arose from the inflexibility of the refinement scheme, as the area of manipulation often changed from too coarse to too fine when the surface was refined.

– It was easier to construct waves from pasted surfaces than it was to simply pull waves up from the water surface. I constructed cresting waves by first pasting a feature onto the water and then manipulating the feature to change its shape. The alternative approach, pulling waves up from the water surface and attempting to manipulate them so they appear to crest, fails because the water surface's control points are too sparse to allow fine-grained changes. The density of the control points could be increased to allow such manipulations, but at the cost of globally refining the water surface.

5.3 Future Work

An improvement to the hierarchical direct manipulation interface would be to allow the user to increase the resolution level for a picked point. For example, suppose there are n levels in the pasting hierarchy and the user wishes to manipulate the picked point at a resolution level of $n+1$. Then a "null surface", with increased control point density, could be pasted under the picked point, and this new surface, rather than the original one, would be manipulated. The ability to edit a surface at a higher resolution would allow the user to add local detail to a region without explicitly pasting a feature surface.

An area for future work would be to research the convergence of the correction factors. It would be interesting to determine why the correction factors sometimes increase when the pasting hierarchy is ascended, how these successive adjustments compare to the original $\overrightarrow{\Delta P}$, and whether the increasing magnitude of the correction factors has a detrimental effect on the manipulation.

Another area for future work would be to test the hierarchical direct manipulation method on a focus group of people who might typically use a pasting editor. The feedback, whether positive or negative, would be invaluable.

Appendix A

Translation Methods for Pasted Surfaces

This appendix describes translation methods proposed by Leith Chan, Stephen Mann, and Richard Bartels for pasted surfaces in a world space user interface. In Section A.1, I explain Chan's projective-translation method and the problems associated with it. Then I discuss two proposed translation methods by Mann and Bartels, with comments on their relative strengths and weaknesses, and possible areas for future work. Note that in this appendix, I will refer to a feature's base surface to mean its immediate underlying surface, not the bottommost surface in the pasting hierarchy.

A.1 Leith Chan's Projective-Translation

In his world space user interface pasting editor, *PasteInterface*, Chan [CMB97] implemented a method for translating a pasted feature called *projective-translation*. Projective-translation was designed to give the user the feeling of sliding a feature across the base surface; the intention is to have the feature follow the direction of the mouse movement. After the mouse button has been depressed, the following procedure is performed at each time interval.

1. Find the centre point of the feature's domain and map it into the base's domain. Then evaluate the base surface at the mapped point and call it P. P will represent the feature's location.

2. Calculate the tangent plane at P using the u and v directional derivatives.

3. Project the mouse movement vector from the view plane to the tangent plane.

4. Further project this projected vector onto the directional derivatives to get Δu and Δv, the domain displacements.

5. Finally, the feature's domain is displaced by $(\Delta u, \Delta v)$ in the base's domain.

This method works reasonably well in most, but not all, situations. For example, suppose a user wants to move a feature to the back of the base surface as shown in Figure A.1. Initially, when the mouse button is depressed, the tangent plane on the surface faces the viewer so an upward mouse motion causes the feature to move upward, as desired. When the feature crosses over the top of the base, the tangent plane begins to face away from the viewer, so an upward mouse motion causes the feature to move in the opposite direction (backwards). Rather than continuing to the other side of the base as desired, the feature reaches the top and moves back and forth a small amount.

Chan's solution is to fix the tangent plane when the mouse button is depressed. Then, when the mouse pointer is moved in an upward direction, the feature progresses up over the top of the base and down to the other side as if the user were sliding it.

However, the use of a fixed tangent plane introduces another problem. For example, suppose the base surface is rainbow-shaped like the one in Figure A.2. At any given point on the surface, the directional derivative with respect to u points towards the centre of the rainbow and the directional derivative with respect to v points in the direction of the curve. Consider the action of moving a feature from the left part of the rainbow to the right part of the rainbow. Ideally, the translation would be accomplished by moving the mouse to the right, but this will not work. The feature will start moving to the right and once it reaches the other end, it will begin moving leftward. The reason for this behaviour will be examined below.

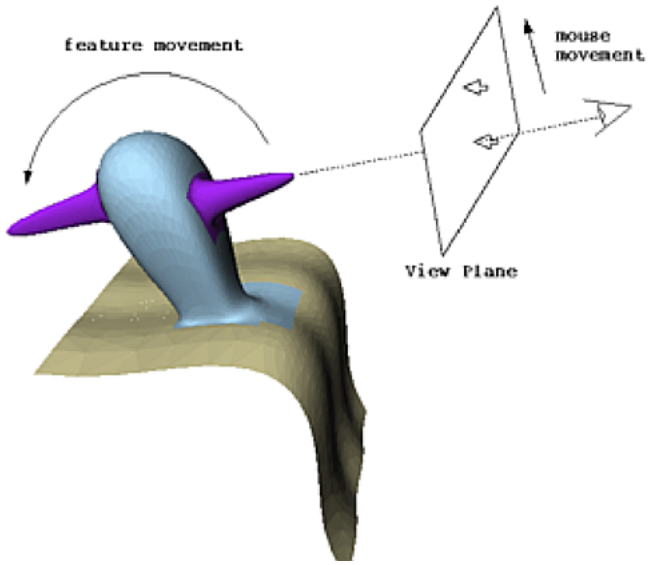


Figure A.1: Sliding a feature behind a base with projective-translation [CMB97]

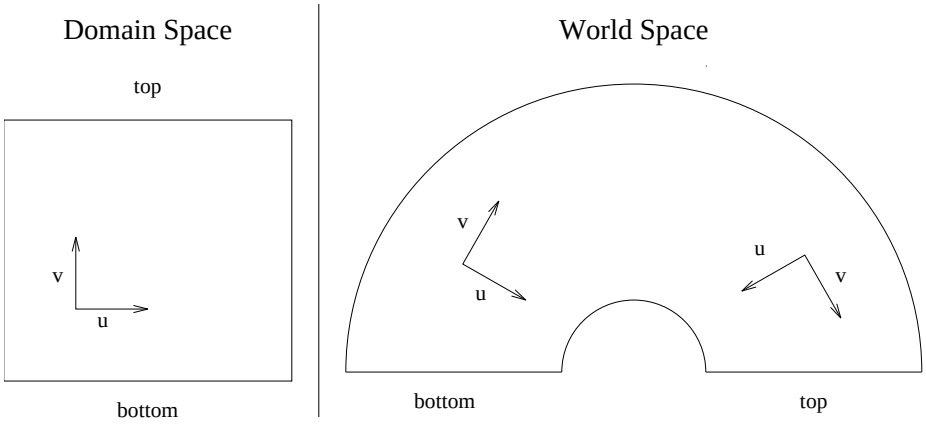


Figure A.2: The rainbow problem

Suppose the mouse button is depressed while the feature is in the left part of the rainbow, and the feature is moved to the right. At each time interval, the mouse movement vector will get mapped to the directional derivatives at the feature's original position and will be converted into a displacement α in the u direction and a displacement β in the v direction. When the mouse is moved to the right, the α and β values will be positive. In the right part of the rainbow, if the feature domain is translated by positive (α, β) values, the feature will move to the left and down in the world space because the orientation of the partial derivatives have changed.

It is interesting to note, however, that Chan's original method, in which the tangent plane is updated at each time interval, behaves as one would expect on the rainbow surface, giving an intuitive motion.

There is a further problem with Chan's projective-translation. Suppose a feature is moved up over the horizon of a base so that it resides on the back part of the base. Within a click-drag-release cycle, it is necessary to move the mouse cursor upward to get the feature to the back of the base and to move the cursor downward to get the feature to return to the front of the base. Suppose the mouse is released when the feature is at the back of the base. To move the feature to the front of the base, the mouse cursor must be moved in an upward direction because the tangent plane at the feature's location when the mouse was depressed is facing away from the viewer. Having to switch the direction of the mouse motion depending on which side of the base the feature was located when the mouse button was depressed is very confusing.

A.2 Stephen Mann's Idea

Stephen Mann has proposed an idea that is intended to give the user a more intuitive sliding motion by ensuring that a mouse movement to the right results in the feature moving to the right, and a mouse movement upward results in the feature moving upward, if the feature is at the front of the base. If the feature is located at the back of the base, these directions will be altered so the motion remains intuitive, e.g., if a feature is moved upward over the horizon of a base, it will continue to move in the same direction along the base so long as the mouse movement continues

upward. After the mouse button has been depressed, the following procedure is performed at each time interval.

1. Find the centre point of the feature's domain and map it into the base's domain. Then evaluate the base surface at the mapped point and call it P. P will represent the feature's location.

2. Calculate the tangent plane at P using the u and v directional derivatives.

3. Choose either the *up* direction or *right* direction, whichever is closer to the tangent plane, to be the principal direction $\vec{d}$ used in future calculations.

4. Take $\vec{x}$ to be the projection of $\vec{d}$ into the tangent plane.

5. Take $\vec{y}$ to be a vector in the tangent plane orthogonal to $\vec{x}$.

6. Map the mouse movement vector (α, β) into the tangent plane as follows:

   - If $\vec{d}$ is the *right* direction, let $\vec{v}_p = \alpha\vec{x} + \beta\vec{y}$.
   - Otherwise, let $\vec{v}_p = \beta\vec{x} + \alpha\vec{y}$.

7. Project $\vec{v}_p$ onto the directional derivatives to get Δu and Δv, the domain displacements.

8. Finally, displace the feature's domain by $(\Delta u, \Delta v)$ in the base's domain.

Mann's method works fairly well in most cases, and possesses none of the problems that Chan's projective-translation exhibited in Section A.1. However, I discovered a small problem with Mann's method. Occasionally, the feature's movement is erratic when the tangent plane is edge-on to the viewer. The feature will move smoothly, then make a small jump in a different direction, and continue moving smoothly from that location.

The problem results from the transition between using the *up* and the *right* directions in the calculations of the domain displacements. In most situations, the translations produced by using the *up* direction and those produced by using the *right* direction as the principal direction are very similar. However, when the tangent plane is edge-on to the viewer, the difference is significant,

producing a translation in an unexpected direction at the transition point. In Section A.4, I describe an improved version of Mann's translation method that overcomes this transition problem.

A.3 Richard Bartels' Idea

All the methods discussed so far have encountered significant problems when the tangent plane is perpendicular or nearly perpendicular to the view plane. One way to get around this problem is to ensure that the tangent plane never becomes perpendicular to the view plane. This is the basic idea behind a translation method that Richard Bartels has proposed.

The domain translations are calculated using one of the previously discussed methods (in my editor, Mann's method is used), but the view is transformed so that the eyepoint is somewhere above the feature and the tangent plane is not perpendicular to the view plane. In particular, the view is updated so that the position of the eyepoint with respect to the middle of the feature remains nearly fixed. This allows the user to select a desired orientation and distance from which to view the feature, and this orientation and distance will be maintained when possible.

The orientation of the eyepoint relative to the feature is not strictly maintained in all cases, since this would result in large changes in the eyepoint's apparent position when the feature is moved across a region of high curvature, or over a point on a surface that has less than $\mathcal{C}^1$ continuity, such as the boundary of a feature surface on a base. To limit the apparent movement of the eyepoint in these circumstances, the angle by which the view direction must change is clamped to a small value. By clamping the angle by which the view direction changes and keeping the distance to the feature constant, the apparent movement of the eyepoint is controlled.

Immediately after a feature is translated over an area of high curvature, the eyepoint's relative orientation to the feature will not be as desired, due to this clamping. The difference in the desired and current orientation will gradually be reduced when the feature is moved over a well-behaved region, as the maximum allowable change in the viewing angle will be applied until the desired viewing orientation is restored.

This method essentially gives the user the feeling of driving the feature over the base surface

since the feature will appear nearly stationary while the base flows by underneath it. It is important to note, however, that the user interface is essentially unchanged; moving the mouse to the right will cause the feature to shift to the right, rather than have it turn to the right as, for example, a car might.

There are several problems associated with the current implementation of Bartels' method.

- If large regions of the base surface have high curvature, the feature may get ahead of the eyepoint, causing the tangent plane to become perpendicular to the view plane, or causing the feature to move out of sight.

- No consideration is given to the position of surfaces in the scene, other than the active feature; it is possible to transform the view so another surface (or a distant part of the base) comes between the eyepoint and the feature.

- The transformation of the eyepoint's position is jerky at times, and can lead to disorientation; a more sophisticated momentum-based approach might improve this aspect of the method.

A.4 An Improvement to Mann's Translation

Recall that while Mann's method had several improvements over Chan's projective-translation, there was a small problem encountered when the tangent plane was edge-on and the principal direction changed from the *right* direction to the *up* direction or vice-versa. I considered using one of two methods to solve the problem:

- find some way to soften the transition, perhaps by blending translations produced by using the *up* direction and translations produced using the *right* direction when the tangent plane is almost equally close to both directions, or

- adjust the method so such a transition would never occur.

While both these methods could possibly produce solutions, the former seemed to be somewhat messy, so the latter idea was pursued. My first observation was that there was nothing special

about using the *up* and *right* directions. *Any* pair of orthogonal directions would probably function as well. The second observation was that if any directions could be used, why not pick one that matches the tangent plane *exactly*, thus eliminating the need for a second direction at all. After the mouse button has been depressed, the following procedure is performed at each time interval.

1. Find the centre point of the feature's domain and map it into the base's domain. Then evaluate the base surface at the mapped point and call it P. P will represent the feature's location.

2. Calculate the tangent plane at P using the u and v directional derivatives.

3. Calculate a vector $\vec{x}$ that is parallel to both the view plane and the tangent plane.

4. Take $\vec{y}$ to be a vector in the tangent plane orthogonal to $\vec{x}$.

5. Let α be the magnitude of the mouse movement vector projected onto $\vec{x}$, and β be the magnitude of the mouse movement vector projected onto a vector in the view plane that is orthogonal to $\vec{x}$ ($\vec{x} \times$ viewdirection).

6. Map the mouse movement vector into the tangent plane as follows: $\vec{v}_p = \alpha\vec{x} + \beta\vec{y}$.

7. Project $\vec{v}_p$ onto the directional derivatives to get Δu and Δv, the domain displacements.

8. Finally, displace the feature's domain by $(\Delta u, \Delta v)$ in the base's domain.

Initial testing of the improvement to Mann's method has not shown any problems. This method behaves correctly when presented with situations that revealed deficiencies in Chan's projective-translation, and does not display the jumping sometimes associated with Mann's original method.

Bibliography

[Bar94] C. Barghiel. Feature oriented composition of B-spline surfaces. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1994. Available on WWW as `ftp://cs-archive.uwaterloo.ca/cs-archive/CS-94-13/`.

[BB89] R. Bartels and J. Beatty. A technique for the direct manipulation of spline curves. In *Proceedings of Graphics Interface*, pages 33–39, 1989.

[BBF95] C. Barghiel, R. Bartels, and D. Forsey. Pasting spline surfaces. In M. Daehlen, T. Lyche, and L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces*, pages 31–40. Vanderbilt University Press, 1995.

[BF91] R. Bartels and D. Forsey. Spline overlay surfaces. Technical Report CS-92-08, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1991.

[Boe80] W. Boehm. Inserting new knots into a B-spline curve. *Computer-Aided Design*, 12:199–201, 1980.

[Cha96] L. K. Y. Chan. World space user interface for surface pasting. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1996. Available on WWW as `ftp://cs-archive.uwaterloo.ca/cs-archive/CS-96-32/`.

[CLR80] E. Cohen, T. Lyche, and R. Riesenfeld. Discrete B-splines and subdivision techniques in computer aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14(2):87–111, 1980.

[CM00] B. Conrad and S. Mann. Better pasting via quasi-interpolation. In Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker, editors, *Curve and Surface Design: Saint-Malo, 1999*, pages 27–36, Nashville, TN, 2000. Vanderbilt University Press.

[CMB97] L. K. Y. Chan, S. Mann, and R. Bartels. World space surface pasting. In *Proceedings of Graphics Interface*, pages 146–154, Kelowna, BC, May 1997.

[Con99] B. Conrad. Better pasting through quasi-interpolation. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1999. Available on WWW as `ftp://cs-archive.uwaterloo.ca/cs-archive/CS-99-14/`.

[Far97] G. E. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 4th edition, 1997.

[FB88] D. Forsey and R. Bartels. Hierarchical B-spline refinement. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):205–212, 1988.

[FB93] B. Fowler and R. Bartels. Constraint-based curve manipulation. *IEEE Computer Graphics and Applications*, 13(5):43–49, September 1993.

[For90] D. Forsey. Motion control and surface modeling of articulated figures in computer animation. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1990.

[Fow92] B. Fowler. Geometric manipulation of tensor product surfaces. *1992 Symposium on Interactive 3D Graphics*, 25(2):101–108, March 1992. ISBN 0-89791-467-8.

[HH90] P. Hanrahan and P. Haeberli. Direct wysiwyg painting and texturing on 3d shapes. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):215–223, 1990.

[LH96] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics Proceedings, Annual Conference Series, 1996. (SIGGRAPH '96 Proceedings)*, pages 31–42, New Orleans, Louisiana, August 4–9, 1996.

[WND97] M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide*. Addison-Wesley Developers Press, Don Mills, Ontario, 2nd edition, 1997.