2-D AND 3-D INTERACTIVE COMPUTER MODELLING SYSTEMS

Richard H. Bartels
John C. Beatty
Kellogg S. Booth
Ines Hardtke

Computer Graphics Laboratory
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
(519) 886-1351

ABSTRACT

We have implemented interactive systems for defining, manipulating and rendering curves and surfaces based on a variety of mathematical techniques including B-spline and Beta-spline representations. The talk will address the underlying mathematics, numerical properties, efficient techniques for rendering, shading and texturing methods, and ergonomic issues regarding the interactive manipulation of the curves and surfaces through their representations. This work was performed in the Computer Graphics Laboratory during the past two and one half years under NSERC Strategic Grant funding.

KEYWORDS: B-spline, Beta-spline, freeform surfaces, interactive modelling.

1. Computer Modelling of Curves and Surfaces

One of the earliest success stories in computer graphics was the modelling of mathematical surfaces based on the pioneering efforts of Coons, Bézier and others. That work was performed in the automative and aerospace industries about twenty years ago. The costs associated with the hardware in those systems kept them from widespread application in areas where computers were not already in use. The general availability of computer graphics equipment at a price affordable by smaller companies has led to new interest in surface modelling within recent years. The likely success stories this time will concern systems that integrate mathematical theory with sophisticated display hardware in a user-friendly environment. A three-part research program is underway within the Computer Graphics Laboratory at the University of Waterloo exploring the application of freeform surface techniques to Computer Aided Engineering.

The first phase of the project has concentrated on the underlying theory used to represent smooth curves and surfaces. Our goal was to reformulate the "classical" mathematical results concerning B-splines in terms more familiar to computer scientists, and to extend those results to the newer theory of Beta-splines, which have greater flexibility because of additional shape parameters. Section 2 summarizes our work in this area and provides a brief introduction for the terminology used later.

Proceeding in parallel with the theoretical work, the second phase of the project has been the design and implementation of interactive software for manipulating curves and surfaces. The primary aim is to explore practical applications of the theory. A fairly primitive prototype was implemented using a raster display, after which a second version was implemented using line-drawing hardware to provide real-time interaction. The software was designed as a teaching aid and to supply hands-on experience comparing the various control parameters available with B-splines and Beta-splines. An easy-to-use interface was a major goal of the design. Section 3 provides a description of the 2-D system currently in operation and of the 3-D extensions being planned.

The final phase of the project, also pursued in parallel, has been the rendering of freeform surfaces and scenes created from them. This work has been performed on our raster display system to take full advantage of colour and shading. Much of this work has involved appropriate division of labour between the host processor (a VAX) and the bit-slice microprocessor attached to the graphics bus. Section 4 provides an overview of the rendering techniques that have been implemented for viewing objects produced using the modelling systems. Plans for further work are outlined in Section 5.

2. The Mathematics of Splines

Curves and surfaces used in Computer Aided Engineering usually do not have simple, closed form descriptions. Modelling systems based entirely on polygons or conic sections have been used, but for representing *freeform surfaces* such as an automobile body, the hull of an airplane, or a soft drink bottle a different representation is usually desirable. A common technique is to use *piecewise polynomial* functions. For curves these are a sequence of polynomial *segments* meeting at *joints* where one segment ends and the next begins. Surfaces are constructed using *patches* formed from the Cartesian product of piecewise polynomial curves.

The cubic B-splines are one set of basis functions for the vector space of piecewise polynomials defined on a *knot sequence* (the parametric values corresponding to joints, the places at which the segments or patches change from one polynomial to the next). Here we will only sketch the main ideas behind B-splines in sufficient detail for the sections that follow. A comprehensive review of our work on splines is available as a technical report [Bartels83b].

B-Splines

Continuity and full differentiability is automatic for piecewise polynomials, everywhere except at the knots, because the functions are polynomials everywhere except at the knots. Care must be taken to select the individual pieces to insure continuity across knots. Depending upon the application, the curves will usually be required to have zero, first and second degree continuity, although sometimes only first degree or zero degree continuity will be needed.

The class of splines that we are interested in are defined by a sequence of *control vertices* forming a *control polygon*. Each control vertex is associated with a knot in parameter space. For surfaces the control vertices form a *control grid* corresponding to the Cartesian product of two knot sequences. In general, knots can be any non-decreasing sequence of parameter values, but frequently the special case of *uniform knot spacing* is used. Distinct knots and control vertices always define curves with at least second degree continuity. Allowing the same knot value to correspond to more than one control vertex (*knot multiplicity*) reduces the degree of continuity at that point on the curve.

Various *end conditions* can be forced. Common conditions are that the curve pass through two specified points (*interpolation*), that it start and end at two specified points (*position*), or that it have particular tangents at its ends (*derivative*). It is also possible to control end conditions by repeating the first and last control vertex one or two times (*double* and *triple endpoints*).

Each of these modifications produces a different family of curves. The key aspect of B-splines that separates them from some other techniques is their *local control*. Changing control vertices or knots affects the curve only in a small neighborhood of the change. This can be very important for modelling applications. It means that minor changes to the headlight housing will not effect the rear trunk of an automobile body. The property of local control follows from the mathematical definition of the B-splines, but it is equally important to have an intuitive understanding of this fact. The latter is best learned from experience with an interactive system such as the one we have implemented.

Beta-Splines

The Beta-splines form a different set of functions. They are one way of relaxing the continuity constraints on B-splines, less general than using multiple knots. The change in Beta-splines from B-splines is the addition of two *shape parameters* that control *bias* and *tension*. Tension has an easy intuitive definition. As the tension parameter is increased the curve (surface) converges to the control polygon (grid). Bias is less intuitive (one reason we have implemented an interactive system that shows the effect of changing bias). It is a weighting of tangent vectors between two adjoining segments (patches) of the curve (surface). Mathematically the difference can be described as substituting a notion of *geometric continuity* (unit tangent vector and unit curvature vector) for the *algebraic continuity* (in terms of the parameterization) required in B-splines.

Although Beta-splines define a subspace of curves and surfaces achievable by B-splines with multiple knots, they are preferable to B-splines for many applications because of the shape parameters. With B-splines the only method available is the placing of control vertices and the manipulation of multiplicities. The two shape parameters are a more natural way of achieving most (but not all) of this control.

Again, the properties of Beta-splines have been studied mathematically, but some of the results are best understood by experimenting with the shape parameters. Large values of the bias parameter, for example, lead to *kinks* in the curve, where it overshoots a control vertex and has to "double back". This is predicted by the equations, but a picture on the screen is often easier to believe than a formal proof.

New Results

Two generalizations of the Beta-splines have been explored to date. The *continuous Beta-splines* allow the shape parameters to vary at each control vertex, allowing even more local control [Barsky82], [Barsky83a], [Barsky83b]. In this case the shape parameters can either be set *globally* with the same value for every control vertex (the normal Beta-spline definition) or *locally* with a different value for each control vertex. There is at least one other way to generalize Beta-splines to have local shape, called the *discrete Beta-splines* [Bartels83a]. It also has different bias and tension parameters for each control vertex, but the class of curves is different from those obtained with the continuous Beta-splines.

Recurrence relations for the B-splines and Beta-splines can be derived from *divided difference* formulas. These lead naturally to subdivision algorithms for rendering based on the *Oslo algorithm*. Refinements on these techniques are being developed using symbolic computation tools to handle the often tedious formal calculations.

This aspect of our research is being pursued both for its potential use in Computer Aided Engineering and for its intrinsic mathematical interest. Similar ideas have also been applied to a different class of splines used in computer animation [Kochanek83], [Kochanek84].

3. Interactive Exploration of Modelling Parameters

An interactive system that illustrates most of the issues discussed in the previous section has been implemented on an Evans & Sutherland Multi Picture System (MPS). The system provides real-time feedback showing the effect of changes to control vertices, knots, shape parameters and end conditions. 2-D curves constructed from B-splines or Beta-splines can be constructed and manipulated. A full 3-D surface version is being built. The 2-D system was implemented in two stages.

Implementation on the MPS had to await the installation of basic device drivers to accommodate the MPS under the Unix operating system. A driver for the Evans & Sutherland PS-2 obtained from the University of California at San Francisco was modified for this purpose [Hayes83]. While this conversion was in progress, a prototype for the spline package was implemented using software already available on our Adage/Ikonas frame buffer. The second version of the system carried over many of the ideas from the prototype, but incorporated significant improvements to the user interface that were difficult to achieve on the raster system.

The Raster Prototype

The interactive spline package divides the frame buffer into two logical pieces, a segmented display list processor and a static frame buffer that supports interactive menus through colour lookup table techniques. Multiple bit planes were used with some bit planes reserved for scan conversion of segmented display lists and other bit planes reserved for menus.

The segmented display list processor was an earlier project used to provide basic graphics support on the frame buffer [Breslin82]. It provides a low-level capability not unlike the hardware features present in calligraphic displays such as the MPS. User programs define segments (sequences of display primitives such as *move*, *draw* and *include subsegment*) which are down-loaded to a resident bit-slice microprocessor. During each refresh cycle (the normal 30 hz video readout) the auto-clear feature of the frame buffer zeros the bit planes containing the scan-converted *dynamic segments*. Synchronized with vertical retrace, the bit-slice microprocessor traverses its display list (stored in local static RAM) and regenerates the display, updating any segments that have changed.

The package is limited in the number of vectors it can draw between vertical retrace and the time the video readout accesses the frame buffer during the next display cycle. Written directly in microassembler, it achieves fairly efficient drawing times. The overhead maintaining the segmented display list is kept low by limiting the operations to basic primitives. 2-D translations are supported, but full transformations are not and no clipping is performed. The segmented display list maintains *static segments* for the curve, its control polygon, and the individual control vertices. It also maintains a dynamic segment for the screen *tracker* associated with the graphics tablet and puck (the primary input device).

Because regeneration time is critical, the static segments are only updated when necessary. Whenever a control vertex is added, deleted or moved the appropriate segment is modified and redisplayed. The curve itself is only updated under user command because the package is not capable of maintaining a flicker-free display for the number of vectors necessary to render a smooth curve.

The dynamic tracker segment is constantly updated to insure prompt lexical feedback to the user. The tracker is used to implement a *locator* for positioning control vertices, a *pick* for selecting control vertices for deletion, and as a *button* (or *choice*) for controlling mode and command selection. The latter is a menu-based scheme in which only those menu items currently selectable appear on the screen. The item selected by the tracker is highlighted (additional lexical feedback) and changes colour when it is confirmed by a button push on the tablet's puck (providing *closure* for the selection task).

The menu package was originally implemented as part of an interactive editor for Benesh dance notation [Singh82], [Singh83]. It was built on top of the segmented display list processor and basic system routines for communicating with the Ikonas frame buffer. The menus are drawn once, during the initialization phase of the program, and colour lookup table techniques are used to "pop up" menu items as they are needed.

Each menu item is assigned a distinct colour number. Menu items not in use have their lookup table entries loaded with the background colour. As they become selectable the entries are changed to a visible colour. When the tracker points to a menu item its lookup table entry is changed again to perform the highlighting, and when confirmed it changes a third time to indicate that a selection has been made. An advantage of this approach is that there is a complete traceback of the user's choices visible on the screen. Inactive menu items are returned to the background colour.

After its development for the dance editor the menu package was used to build a prototype VLSI layout system (later moved to Orcatech 3000 workstations) and the spline editor. Similar techniques have been used in other systems implemented within the Computer Graphics Laboratory and constitute a major area of our research into effective interaction techniques.

The raster prototype allows the user to construct interpolating B-splines through the addition, deletion and modification of control vertices. Its primary drawback is that the really interesting changes, those that result from moving control vertices, require too much recomputation to be performed within a single vertical retrace. Elaborate boxing tests could be used to reduce the amount of computation, but self-intersecting curves and other pathological cases would still be difficult to handle efficiently with a general algorithm. Our plan from the start had been to use a line-drawing system capable of real-time transformations to provide a view of the curve changing in response to control vertices being dragged to new positions.

The Calligraphic Version

The Evans & Sutherland Multi Picture System contains a very powerful display processor capable of transforming a segmented display list at close to real-time rates. A double buffering scheme whereby the previously transformed display list is used to refresh the screen during the next transformation cycle guarantees a flicker-free display even when the transformations cannot be performed in a single refresh cycle. The system uses a high performance monochrome display (4096 addressability) that provides very fast line and character drawing.

The prototype package written for the Ikonas frame buffer was used as a model for the MPS package, but most of the program was written from scratch because the system was re-designed to take advantage of the unique capabilities of the MPS. The scope of the package was also broadened to include Beta-splines with user specification of control vertices, their multiplicities, global and local shape parameters, and a variety of endpoint conditions.

The MPS package employs a static menu system. In the Ikonas package many of the menu items trigger updates to the display. This is performed automatically in the MPS version so those menu items are no longer needed. Many new menu items have been added that set *modes*. These are more natural to display permanently, so the user sees the state of the system.

The screen is divided into a number of separate *windows*. Some contain menus, grouped according to function. One window contains the primary display showing the control vertices, the control polygon and the curve defined by the control vertices. Two auxiliary windows show the x and y coordinates of the control vertices to facilitate independent control over the two axes (implicitly giving the knot parametrization). A fourth window displays the basis functions and a fifth window is used to set shape parameters.

All of this occupies too much area to be effectively displayed. The solution we chose was to allocate the majority of the screen area to one of the windows (the default large window is the primary display showing the curve), relegating the other windows to much smaller areas. These provide a quick overview of the information, but sometimes not enough detail for fine-tuning. The user can "swap" any two windows simply by selecting the appropriate menu item and then pointing at the windows to be swapped. If the large format is still not enough, any window can be zoomed using another menu item followed by a center and scaling selection.

Display parameters selecting the control vertices, the control polygon and the curve itself are implemented as menu items that *toggle*. This allows the user to tailor the display to his needs. Parameters such as endpoint conditions and global vs. local shape parameters are also implemented as toggles. Their status is always available in the menu area because the current setting is highlighted using the brightness control available on the MPS.

Single, double, and triple end conditions may be selected which implicitly change the multiplicity of the first and last control vertices. Multiplicities of one, two or three can be set explicitly for any control vertex. The curves can be either open (distinct endpoints) or closed (the curve wraps onto itself).

Shape is controlled globally using two sliders for the bias and tension parameters, or locally by first picking a control vertex and then the two parameter values using the sliders. The sliders are implemented using the graphics tablet as a *valuators*. As the bias and tension parameters are changed, the curve changes automatically. Effects such as kinks are readily visible using these controls.

One aspect of the MPS implementation that is not very satisfactory is the lexical feedback. The primary input device is the tablet and stylus attached to the MPS. One bottleneck is communication with the host. To provide effective tracking the host must interrogate the MPS. This is multiplexed with display updates using the same I/O facilities. A bit embarrassingly, the MPS package sometimes is more sluggish than the Ikonas version.

Some of this is no doubt remediable with better software, especially the use of multitask programs that isolate the input and output functions into more natural modules. This organization has worked well on other systems, most notably an interactive Paint system [Beach82], [Booth84], [Plebon82]. The Evans & Sutherland PS-300 system takes a step in this direction by providing a 68000 microprocessor to orchestrate the input devices, although the initial versions offered little opportunity for application programs to control critical aspects of the user interface. This is an area that we feel needs the most attention in our present implementation. Our current driver does not take advantage of all of the features available on the MPS because it was originally programmed for a PS-2.

Additional Features

The extension to 3-D is the next obvious step. The present version of the system handles only 2-D curves. Curves in 3-D would be easy to add to the display, although the specification of control vertices in 3-D presents some interesting problems for the user interface. Full 3-D surfaces are the ultimate goal. Some problems are anticipated for elaborate surfaces, both because the number of vectors is large (a potential flicker and update problem) and because the image becomes "busy" and difficult to understand, especially if the control grid is superimposed on the surface.

Our current efforts are directed toward tuning the user interface and providing a complete set of examples to illustrate the mathematical theory. The system has been used in a graduate seminar as an aid to understanding the properties of B-spline and Beta-spline curves and surfaces.

4. Rendering Algorithms

The images produced by our modelling system are restricted to 2-D curves. Other work within the Computer Graphics Laboratory has concentrated on rendering algorithms for 3-D surfaces. One early project was a system for defining surfaces of rotation and extrusion that were rendered using realistic shading models [Sherwood83]. Our latest work in this area uses ray-tracing algorithms employing subdivision of the knot sequence [Sweeney83].

The rendering algorithms accept structured scene files composed of polygons, spheres, cylinders and freeform surfaces. Multiple light sources may be specified and each object may receive a texture map from an existing image. Optional properties include Cook-Torrance shading and reflection-refraction calculation.

5. Future Plans

We see the primary purpose of the current system as an aid to research and teaching. The balance between theoretical considerations (properties of B-splines and Beta-splines) and the practical considerations (user interface design and rendering algorithms) provides a number of challenges for the system designer.

On the theoretical side, we have yet to implement nonuniform knot spacing, the discrete version of the varying shape parameters, and some of the end conditions such as position, derivative and interpolation. For the user interface we need to make better use of the refresh and device controller in the MPS to improve interaction times. Many tasks now performed by the host could be handled by the MPS.

The 3-D implementation is likely to be quite different from the 2-D package. The issues involved with surfaces are somewhat different from those for curves, even though the mathematics is very similar. The issue of local control becomes a bit more difficult because the Cartesian product construction for patches means that introducing multiple control vertices or multiple knots has the undesired side effect of increasing the number of patches in a more global sense than for the simple case of curves. We are investigating subdivision techniques that work locally within the control grid to overcome this. We anticipate that these issues will overshadow considerations such as end conditions, which do not have nearly as interesting properties for surfaces as for curves.

The effective use of colour is something we have not fully explored in this context, although we are pursuing a number of projects based on the use of colour in computer graphics [Goetz82] [Schwarz84]. Work is also being done on more efficient rendering algorithms for raster displays. We are attaching a special purpose *geometry processor* (a custom microprocessor designed and implemented at Tektronix Laboratories [Bates82]) to our Ikonas frame buffer to provide a real-time transformation capability for raster images.

All of this work uses the frame buffer. Many of the critical inner loops run in microcode using a compiler for a subset of the "C" programming language written for the bitslice processor on the Ikonas [Gurd83].

Our current implementations have taken into account the display technology (raster or calligraphic) available. Newer systems such as the Silicon Graphics Iris workstations or workstations developed by Orcatech offer an attractive marriage of raster and calligraphic capabilities. We plan to explore this in greater detail as such systems become available.

Real-time display of full shaded surfaces is still in the future, but simple z-buffer visible surface calculation with faceted shading from a single light source is possible now. Incorporating the full flexibility of general spline techniques into this environment requires a complete understanding of the issues we are addressing.

Acknowledgements

The work reported here is the result of a team effort involving all of the members of the Computer Graphics Laboratory. Michael Sweeney is responsible for many of the rendering algorithms that have been implemented. The references cited here refer only to work done within the Computer Graphics Laboratory. A more complete survey of the laboratory's efforts is available in a techni-

cal report [Bartels83c]. References to the literature appear in the cited reports.

This project has been conducted with the collaboration of Prof. Brian Barsky who has provided many valuable insights. Primary funding was provided by the Natural Sciences and Engineering Research Council of Canada through Strategic Equipment and Operating Grants. The Province of Ontario provided equipment funds under its BILD Program. Additional hardware and operating support was provided by the University of Waterloo.

References

[Barsky82]

Brian A. Barsky and John C. Beatty, Varying the Betas in Beta-splines, Technical Report CS-82-49, Department of Computer Science, University of Waterloo (1982).

[Barsky83a]

Brian A. Barsky and John C. Beatty, Local control of bias and tension in Beta-splines, *Transactions on Graphics 2:2* April, 1983) pp. 27-52.

[Barsky83b]

Brian A. Barsky and John C. Beatty, Controlling the shape of parametric B-spline and Beta-spline curves, *Graphics Interface '83* (May, 1983) pp. 223-232.

[Bartels83a]

Richard H. Bartels, Splines in interactive computer graphics, *Proceedings of the 1983 Dundee Conference on Numerical Analysis* (1983).

[Bartels83b]

Richard H. Bartels, John C. Beatty and Brian A. Barsky, An introduction to the use of splines in computer graphics, Technical Report CS-83-09, Department of Computer Science, University of Waterloo (1983).

[Bartels83c]

Richard H. Bartels, John C. Beatty, Kellogg S. Booth and Daniel E. Field, Computer Graphics Laboratory Fall 1983 Review, Technical Report CS-83-33, Department of Computer Science, University of Waterloo (1983).

[Bates82]

Roger Bates, Jay Beck, John C. Beatty, Kellogg S. Booth, Terry Laskodi, Larry H. Matthies, Ed Reuss and Marc Wells, A high-performance raster display system, *Proceedings Graphics Interface '82* (May, 1982) pp. 355-364.

[Beach82]

Richard J. Beach, John C. Beatty, Kellogg S. Booth, Darlene A. Plebon and Eugene L. Fiume, The message is the medium: Multiprocess structuring of an interactive paint program, *Computer Graphics 16:3* (July, 1982) pp. 277-287.

[Booth84]

Kellogg S. Booth, W. Morven Gentleman and Jonathan Schaeffer, Anthropomorphic programming, Technical Report CS-82-47, Department of Computer Science, University of Waterloo (1984).

[Breslin82]

Paul H. Breslin and John C. Beatty, A powerful interface to a high-performance raster graphics system, Technical Report CS-82-45, Department of Computer Science, University of Waterloo (1982).

[Goetz82]

Susan M. Goetz-Obermeyer, John C. Beatty and Deryl J. Rasquinha, Colour principles and experience in computer graphics, *Proceedings Graphics Interface '82* (1982) pp. 313-322.

[Gurd83]

R. Preston Gurd, Experience developing microcode using high level language, *Proceedings of the 1983 SIGMICRO Conference* (1983).

[Hayes83]

Carol J. Hayes, System support for the Evans & Sutherland Multi Picture System, master's thesis, Department of Computer Science, University of Waterloo (1983).

[Kochanek82]

Doris H. Kochanek, Richard H. Bartels and Kellogg S. Booth, A computer system for smooth keyframe animation, Technical Report CS-82-42, Department of Computer Science, University of Waterloo (1982).

[Kochanek84]

Doris H. Kochanek and Richard H. Bartels, Interpolating splines for keyframe animation, this volume.

[Plebon82]

Darlene A. Plebon and Kellogg S. Booth, Picture creation systems, Technical Report CS-82-46, Department of Computer Science, University of Waterloo (1982).

[Schwarz84]

Michael W. Schwarz, John C. Beatty, William B. Cowan and Jane F. Gentleman, Towards an effective user interface for interactive colour manipulation, this volume.

[Sherwood83]

Geoff C. Sherwood, An object maker system, master's essay, Department of Computer Science, University of Waterloo (1983).

[Singh82]

Baldev Singh, John C. Beatty, Kellogg S. Booth and Rhonda Ryman, A graphics editor for Benesh Movement Notation, Technical Report CS-82-41, Department of Computer Science, University of Waterloo (1982).

[Singh83]

Baldev Singh, John C. Beatty, Kellogg S. Booth and Rhonda Ryman, A graphics editor for Benesh Movement Notation, *Computer Graphics 17:3* (July, 1983) pp. 51-62.

[Sweeney83]

Michael A. J. Sweeney and Richard H. Bartels, Ray tracing free form spline surfaces, submitted for publication.