Applications of Bezier Clipping Method and Their Java Applets

Tomoyuki Nishita
Faculty of Engineering, Fukuyama University,
985 Higashimura-cho, Fukuyama, 729-0292 Japan
E-mail: nis@eml.hiroshima-u.ac.jp

Abstract
Displaying objects with high accuracy is important in CAGD and for the synthesis of photo-realistic images. The representation of free-form surfaces can be classified into two: parametric surfaces such as Bezier patches, and implicit surfaces like metaballs. We discuss display methods for both Bezier patches and metaballs by using Bezier Clipping. Traditionally, polygonal approximation methods have been employed to display parametric surfaces. This paper introduces various display methods for Bezier patches without polygonal approximation. Bezier Clipping can be also applied to the following: 1) curve/curve intersection, 2) curve/surface intersection, 3) scan conversion of curved regions such as outline fonts, 4) various lighting simulations such as curved light sources and radiosity method. In order to show the effectiveness of Bezier Clipping widely by using the Internet, we have coded some of them(e.,g.,curve/curve intersection, metaballs) in Java language. The Bezier Clipping is very effective for displaying metaballs and metacircles (2D version of metaballs) and for the application of metaballs, we demonstrate realistic rendering of clouds, snow, smoke, and water droplets.

1. Introduction

Effective rendering methods for curved surfaces are discussed here. The representation of free-form surfaces can be classified into two categories: parametric surfaces and implicit surfaces. For the former, Bezier patches, B-spline patches, and NURBS are used. For the latter, algebraic surfaces and a set of density functions such as metaballs (or blobs) are used. This paper discusses the idea of Bezier clipping and its application to various rendering techniques. In order to show the effectiveness of Bezier Clipping widely by using the Internet, we have coded some of them in Java language.

Bezier clipping can be applied to hidden line/surface removal of Bezier patches. Bezier clipping can be also applied to raytracing of metaballs. The Bezier clipping technique can be applied not only to hidden line/surface removal but also to various shading effects.

2. Basic Idea of Bezier Clipping

Bezier clipping is an iterative method which takes advantages of the convex hull property of Bezier curves, and iteratively clips away regions of the curve which don't intersect the line. Thus we can refer to this method as an *interval*

Newton method. Bezier clipping converges more robustly with the polynomial's solution than does Newton's method. This method was first developed for raytracing Bezier patches[Nishi90].

The advantages of this technique are as follows:

(1) Applicable to a high order of polynomial (and rational functions)
(2) Robust
(3) No initial guess necessary
(4) All solutions within specified range
(5) Minimum/maximum root available if necessary
(6) Quick test for non-intersection
(7) High-degree Bezier function/curves solved by iterations using only linear equations (i.e., Bezier clipping uses only linear equations in each iteration).

The Newton method is often used for numerical analysis, but it requires a suitable initial guess, and it is difficult to be sure of finding all solutions**.** Bezier clipping overcomes these problems.

2.1 Applications of Bezier Clipping

The following are applications of Bezier clipping.

(1) Root finder for polynomials
(2) Basic geometric problems: curve/curve intersection[Seder90], curve /surface or surface/surface intersection[Seder91]
(3) Hidden surface removal for parametric surfaces: raytracing[Nishi90], scanline algorithm[Nishi91a], hidden line algorithm[Nishi92b]
(4) Hidden surface removal for metaballs[Nishi94a]
(5) Shading models: cylindrical light sources[Nishi92a], curved light sources and radiosity[Nishi94b], optical effects on curved surfaces such as caustics[Nishi94b] or water drops[Kaneda96], natural phenomena such as clouds[Nishi96]
(6) 2-D computer graphics: scan conversion of curved regions, outline fonts [Nishi91b], brush strokes, watercolor painting[Nishi93a], morphing [Nishi93b], and metacircles(2D version of metaballs).

As described above, Bezier clipping can be applied to many fields. This paper will focus on 3-D rendering, the details of (5) and (6) are omitted.

2.2 Solving to Polynomials

Polynomials can be converted to a Bezier curve. For example, a degree three polynomial can be converted to the cubic Bezier function. Fig. 1(a) shows the polynomial ($f(x) = 24x^3-42x^2+2x+2$) converted to the following a cubic Bezier function;

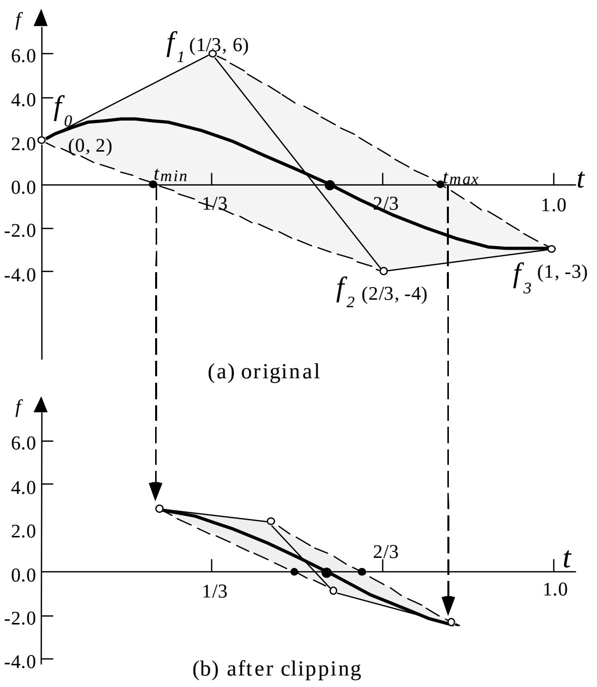


Fig.1 Solving to polynomial by Bezier clipping

$$f(t)=\sum_{k=0}^{3} f_k B_k^3(t) \tag{1}$$

where (k/3,f_k) (k=0,..,3) are control points of the Bezier function, and B_k^3 is the Bernstein function**.** The shaded region is the convex hull of the curve. The root of the curve always exists within the intersection between the axis and the convex hull; that is, the interval $t_{\min}$ and $t_{\max}$. By clipping the curve at $t_{\min}$ and $t_{\max}$, we can get a new curve with thinner convex hull as shown in Fig. 1(b). As the remaining part of the curve approaches a straight line, the intersection interval between the convex hull and the *t* axis rapidly narrows at the next step. By repeating this process we can find the root.

Iteration terminates when the intersection interval between the convex hull and the t axis is smaller than the given tolerance. In the first step, in the example of Fig. 1, the interval is 0.55, but in the third iteration, the interval is only 0.0003. After three iterations we can find the intersection point.

If the intersection between the convex hull and the axis is relatively large, there is a possibility of multiple roots. In that case, the curve is subdivided at the mid point into two curves. And Bezier clipping can then be applied to each curve.

Fig.2 shows the Java Applet for solving to polynominal (degree 6 in this case). We can show how effective the Bezier clipping method is for interactive systems through the Internet. In this applet, we can hear the word “get” when the roots are found. This sound help us make an attractive system.

2.3 Curve-Line Intersection

Fig.2 shows a line and a cubic Bezier curve. The Bezier curve with control points $P_k(x_k,y_k)$ is expressed by the following equation.

$$x(t) = \sum_{k=0}^{3} x_k B_k^3(t), \quad y(t) = \sum_{k=0}^{3} y_k B_k^3(t) \tag{2}$$

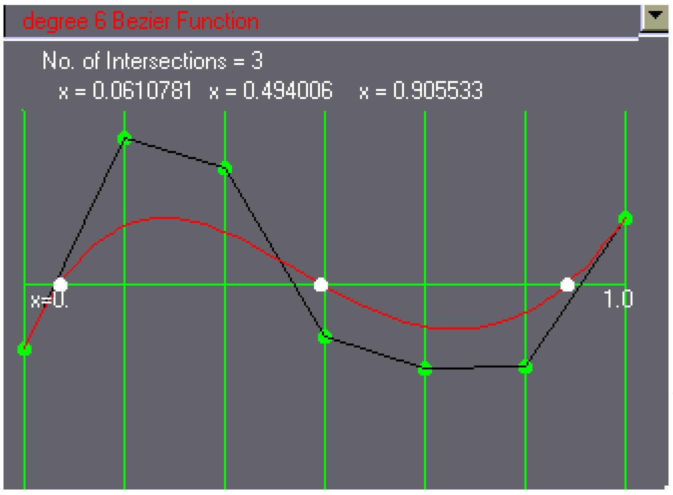
And the distance from any point (x,y) to the line is expressed by

$$d(x,y) = ax + by + c \tag{3}$$

By substituting *x* and *y* of the curve equation to the line equation, we can get the following Bezier function.

$$d(t) = \sum_{k=0}^{3} f_k B_k^3(t)$$
$$f_k = ax_k + by_k + c \tag{4}$$

where f_k is equivalent to the distance between the control point P_k and the line. Equation (4) is called distance function. (a,b) is the unit normal of line ($a^2+b^2=1$). Fig.1(a) shows the distance function expressed by the Bezier function. Parameter *t* at the intersection with the *t*-axis gives us the intersection between the line and the curve. Bezier clipping solves this intersection.


degree 6 Bezier Function

No. of Intersections = 3

x = 0.0610781 x = 0.494006 x = 0.905533

Fig.2 Java Applet for root finder of polynominal

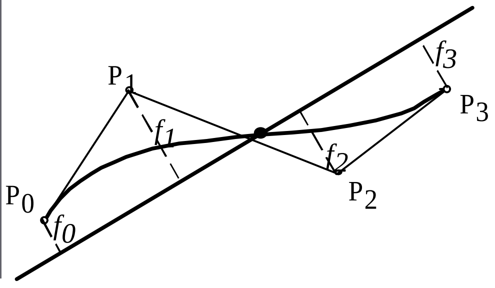
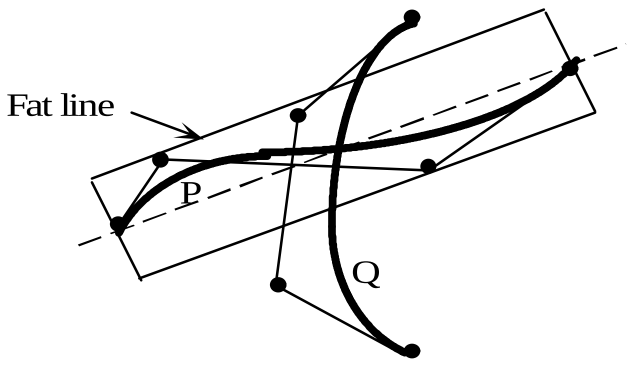


Fig. 3: Curve/line intersection.

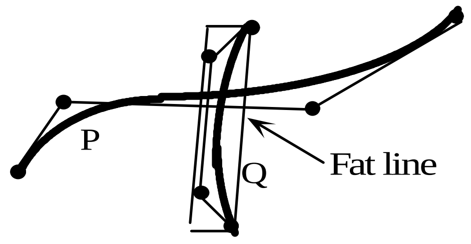
2.4 Curve-Curve Intersection

For curve/curve intersection test, we can introduce the idea of *FatLine*, which is a bounding box of the curve[Seder90]. Let's consider curve P and curve Q shown in Fig. 3. Curve Q is clipped with the FatLine of curve P. This gives us the small curve of P. Curve P can be clipped with the FatLine of curve P. By repeating this process, we can find the intersection point. See reference [Nishi92] for details.

Fig.5 shows the Java Applet for curve/curve intersection (degree 3 and 6 Bezier curves); three intersection points in this case.

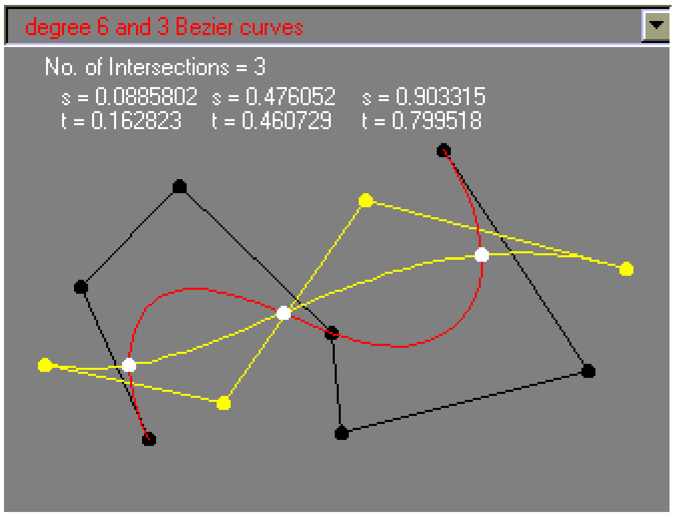


(a) clip Q by Fatline of P



(b) clip P by Fatline of Q

Fig.4 Curve/curve intersection test by using Fatline.



degree 6 and 3 Bezier curves

```
No. of Intersections = 3
  s = 0.0885802  s = 0.476052   s = 0.903315
  t = 0.162823   t = 0.460729   t = 0.799518
```

Fig.5 Java Applet for curve/curve intersection.

3. Display of Bezier Patches

In this section, we discuss a hidden surface removal method for Bezier patches.

Let's consider previous work on hidden surface removal of parametric surfaces. Solutions to the ray/patch intersection problem can be categorized as being based on subdivision or numerical techniques. Whitted[Whitt80] first developed the subdivision method. Kajiya's algorithm[Kajiya82] reduces the problem of intersecting a bicubic patch with a ray into one of finding the real root of a degree 18 polynomial. Our method[Nishi90] belongs to the subdivision method. After our paper was published, Fournier[Fourn94] used Chevyshev basis functions to speed up the ray/patch intersection test. The properties of Chevyshev polynomials result in the computation of better and tighter enclosing boxes. Kim[Kim95] has expanded our method. He reduced the amount of computation as much as possible by trying to find only the nearest point instead of computing them all. He built a BSP tree for each original patch in the preprocessing stage by doing adaptive subdivision over the surface. This binary tree allows us to find which part of the subdivided patch is likely to contain the nearest intersection from the viewpoint.

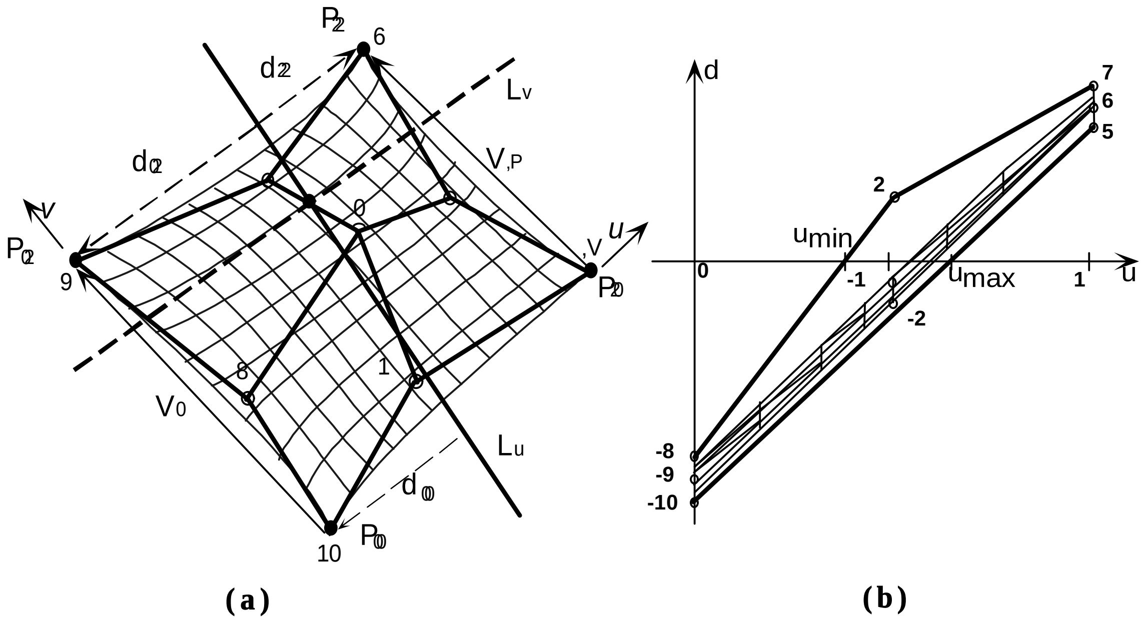


Fig.6 Ray/surface intersection.

Raytracing means to find *(u,v)* parameters from *(x,y)* coordinates on the screen. The viewing ray is the line intersecting two planes. After transforming the Bezier patch to be ray passing through the origin, the two planes become the lines, L_u and L_v (see Fig. 6(a)), passing through the ray (i.e., origin); the line equation passing through the origin is expressed by

$$L_u(x,y) = a_u x + b_u y\,. \tag{5}$$

The projected cubic Bezier patch is expressed by

$$x(u,v) = \frac{\sum_{i=0}^{3}\sum_{j=0}^{3} w_{ij} x_{ij} B_i^3(u) B_j^3(v)}{\sum_{i=0}^{3}\sum_{j=0}^{3} w_{ij} B_i^3(u) B_j^3(v)}, \quad y(u,v) = \frac{\sum_{i=0}^{3}\sum_{j=0}^{3} w_{ij} y_{ij} B_i^3(u) B_j^3(v)}{\sum_{i=0}^{3}\sum_{j=0}^{3} w_{ij} B_i^3(u) B_j^3(v)} \tag{6}$$

where $x_{ij} = X_{ij} / Z_{ij}$, $y_{ij} = Y_{ij} / Z_{ij}$, $w_{ij} = W_{ij} / Z_{ij}$, and (x_{ij},y_{ij}) is the projected control point of $P_{ij}(X_{ij},Y_{ij},Z_{ij})$; W_{ij} is weight for the control point.

By substituting *x* and *y* equations in the line equation, we get the following equation.

$$d_u(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} d_{ij} B_i^3(u) B_j^3(v) \tag{7}$$

$$d_{ij} = a_u x_{ij} + b_u y_{ij}\,.$$

Fig.6(a) shows the control point distances d_{ij} (d_{ij} for each control point is displayed in the figure). The function *d* can be represented as an explicit surface patch whose control points (u_{ij}, v_{ij}, d_{ij}); u_{ij}=i/3, v_{ij}=j/3. Even though *d* is function of (*u*,*v*), Figure 6(b) is a side view of the *d*(*u*,*v*) patch, the convex hull of the projected control points bounds the projection of the d patch. We can find the range having intersections (see [$u_{\min}$, $u_{\max}$] in Fig. 6(b)) by this figure. This process of identifying values $u_{\min}$ and $u_{\max}$ which bound the solution set, and then subdividing off the regions $u<u_{\min}$ and $u > u_{\max}$. In a similar manner, we define the process of Bezier clipping in parameter v. Our ray-patch intersection algorithm consists of alternately performing Bezier clipping in u and v. By repeating this process, we can get the small patch which is the intersection point.

Fig.7 shows examples of Bezier patches. (a) is raytracing, (b) is an example of radiosity using scanline algorithm [Nishi93d].

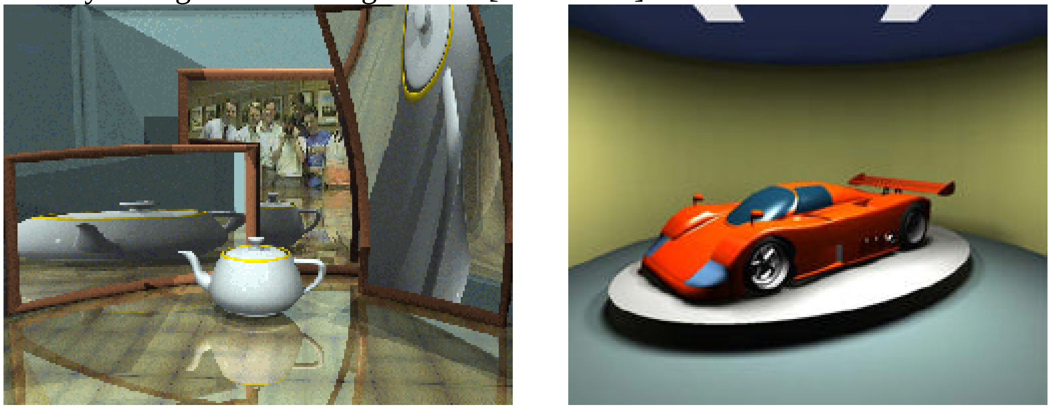

(a) raytracing (b) scanline algorithm with radiosity

Fig.7 Examples of Bezier patches

4. Displaying Metaballs

The features of metaballs are as follows: (1) the required data for metaballs is typically at least two to three orders of magnitude smaller than that modeled with polygons, (2) metaballs are suitable for use in the CSG model, (3) they are suitable for the representation of deformable objects, making them useful for animation. (4) they are well suited for modeling of human bodies, animals, organic models, and liquids. Because of such a usefulness, many commercial software packages implement metaball modeling techniques. The metaball technique has become an indispensable technique in 3-D graphics software. This modeling technique was first developed by Blinn[Blinn80] who called it *blobs*. In Japan, Nishimura et al.[Nishim85] developed it independently, and called it *metaballs*.

5.1 Field function

In the metaball technique, a free-form surface is defined as an isosurface (equi-potential surface) of a field function. The field value at any point is defined by distances from the specified points in space. We used the degree six field function proposed by Wyvill[Wyvill86]. If two balls are placed at the same location, it has twice the volume of the isosurface for a single ball. Thus, for geometric modeling, degree six polynomial function is useful expressed by

$$f_i(r) = -\frac{4}{9}(\frac{r}{R_i})^6 + \frac{17}{9}(\frac{r}{R_i})^4 - \frac{22}{9}(\frac{r}{R_i})^2 + 1 \quad (8)$$

where R_i is the radius of metaball *i* and r is the distance from a point to the center $P_i(x_i, y_i, z_i)$.

For *n* metaballs, the shape of the curved surface is defined by the points satisfying **t**he following equation.

$$f(x,y,z) = \sum_{i=1}^{n} q_i f_i - T = 0 \quad (9)$$

where *T* is a threshold, q_i the density values at the center of metaball *i*.

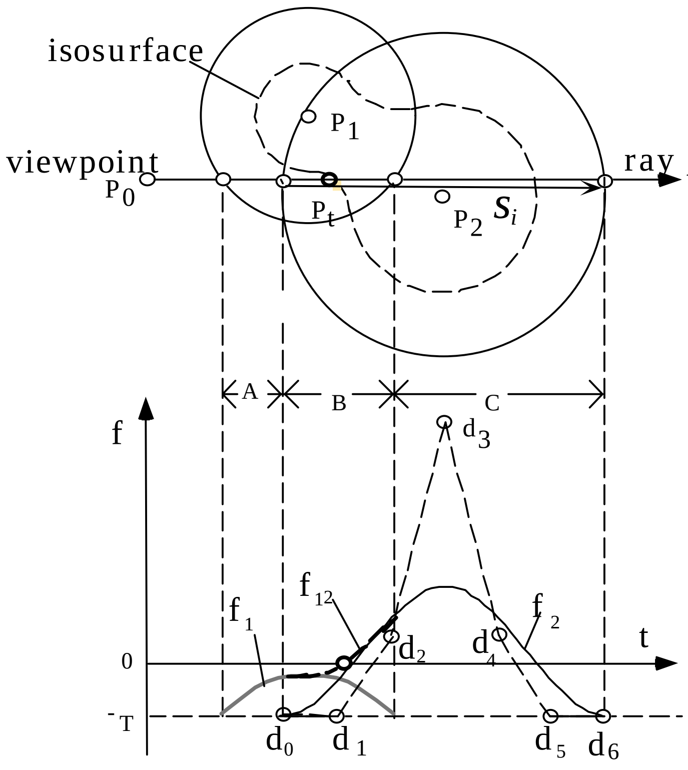


Fig.8 Density distribution on the ray.

4.2 Intersection Test between Ray and Metaballs

The main task for rendering metaballs is intersection tests between rays and isosurfaces. In our algorithm[Nishi94a], the field function on the ray is expressed by Bezier functions, so the root of this function is effectively and precisely solved by Bezier clipping.

Let's discuss the intersection test between a ray and multiple metaballs. Fig.8 shows a ray and an isosurface defined by two balls and shows the density distribution on the ray. By using parameter s_i (0<s<1) on the intersected interval with ball *i* (see Fig.8), the density is expressed by

$$d(s_i) = \sum_{k=0}^{6} d_k B_k^6(s_i) \quad (10)$$

where $d_0 = d_1 = d_5 = d_6 = 0$, $d_2 = d_4 = \frac{16}{27}a_i^2$, $d_3 = 8\frac{(8a_i + 5)a_i^2}{45}$, and a_i is (intersected length/R_i)^2 ($0 < a_i < 1$).

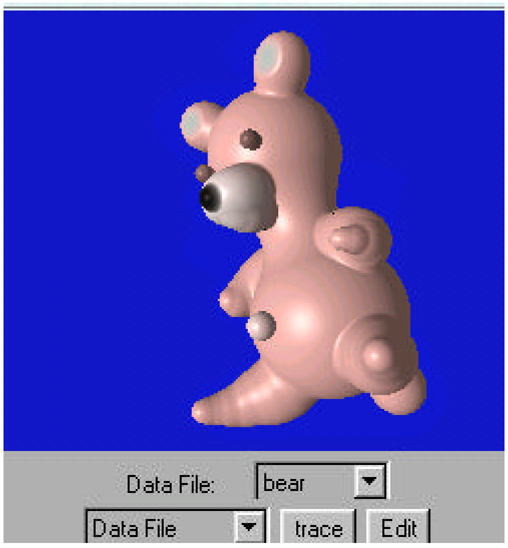


Fig.9 Java Applet for metaballs

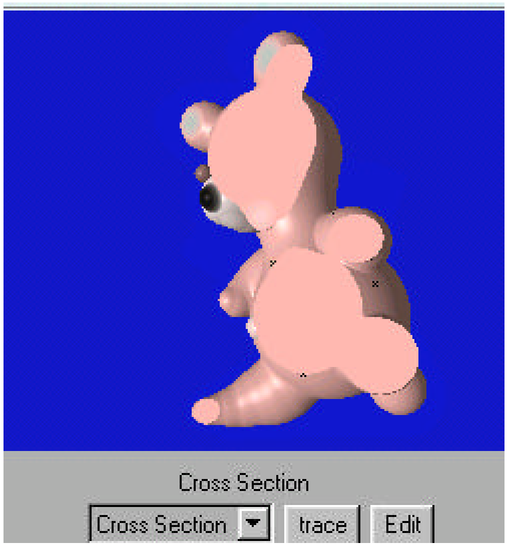
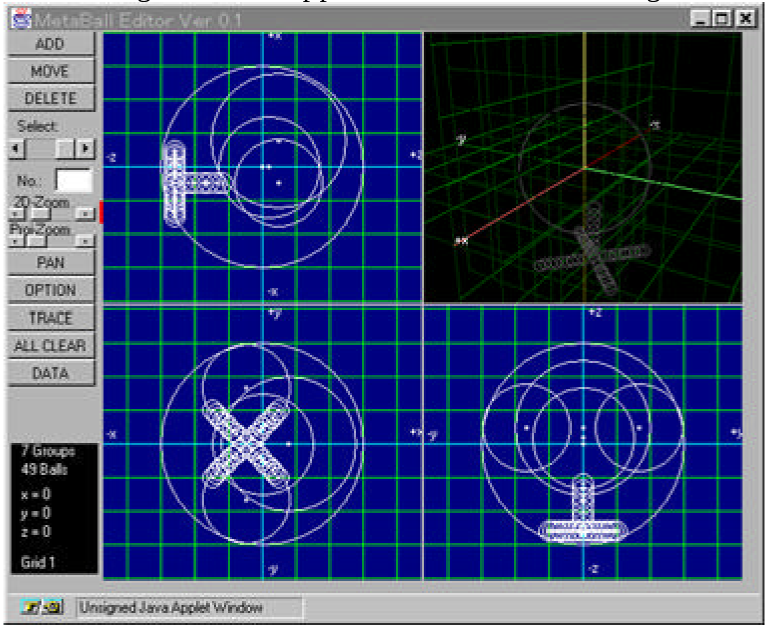
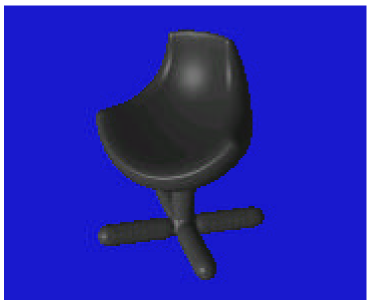


Fig.10 A cross section of metaballs.



(a)



(b)

Fig.11 Java Applet for metaball editor.

As shown in Fig. 8, Bezier curves, f_1 and f_2, are clipped by the interval to be tested (i.e., section B in the figure), then both of the clipped curves are composited. This composting of the curves is very simple. It is performed by simply adding each control point d_{ki} belonging to f_1 and f_2; After composting the curves, the new curve f_{12} is also expressed by a degree six Bezier curve, then the root (e.g., P_t in Fig. 8) is found by using Bezier Clipping.

Fig.9 shows the Java Applet for rendering metaballs. In this applet, we can add some balls interactively, and can get a cross section of the object by clicking just three points on it(see Fig.(b)). Fig.11(a) shows Java Applet for editing metaballs. Fig.(b) shows the result of the editing. In this case balls with negative density are used to obtain holes.

Fig. 12 (a) shows killer whales modeled by metaballs. This is one frame from a HDTV size animation. This figure shows the optical effects in water such as caustics on the whales and shafts of light[Nishi94b].

Fig. 13(a)-(b) show examples of two types of free-form surfaces in the scene. Fig. 13(a) shows Bezier surfaces (car and tree) and metaballs (clouds and a frog). Both types of surfaces can be displayed by a single program. The metaball technique is also useful for displaying translucent objects such as clouds/smoke. Clouds are defined by density fields, which are modeled by the metaball technique. That is, the surface of a cloud is defined by the isosurfaces of potential fields defined by the metaballs. Multiple scattering is taken into account for the clouds[Nishi96a]. The intersections of the isosurface (i.e., cloud surface) with the viewing ray are calculated by raytracing based on Bezier clipping. Fig.(b) shows teapot with water droplets; the teapot is modeled by 32 Bezier patches, and water droplets are defined by metaballs.

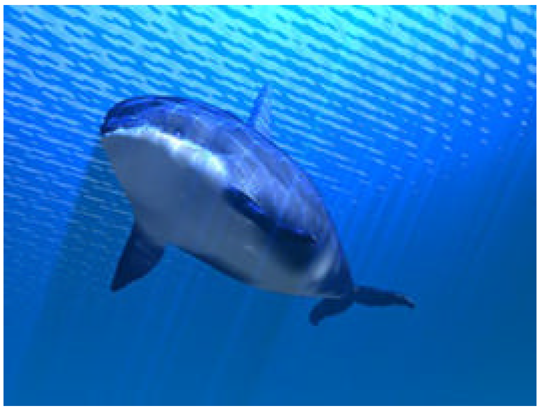
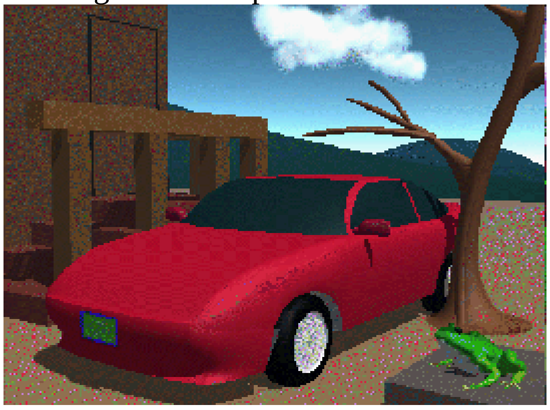
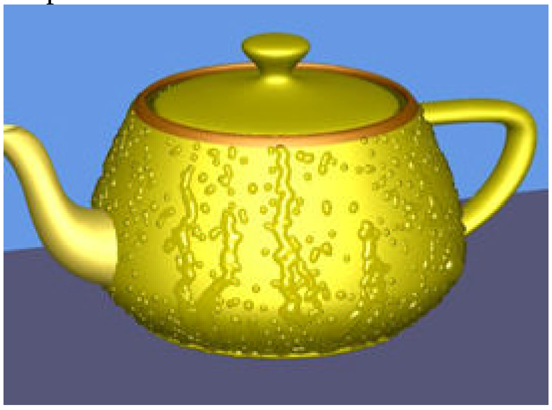


Fig.12 Examples of metaballs with optical effects within water.



(a)



(b)

Fig.13 Examples of raytraced scenes containing Bezier patches and metaballs

5. Displaying Metacircles

Ranjan et al [Ranjan96] used a representation of objects as a union of circles to define the distance between two objects and to base a method to interpolate between the two. They used circles because circles can be simplified to obtain smaller data sets. In their method, however, the boundaries of objects are not smooth because of C^0 continuity due to sets of arcs. Chung also used circles to define the images[Chung97]. To overcome this problem, we employed meta-circles which are a 2-D version of metaballs. We can get smooth curved boundaries because of the density field within the circles. Our purpose is to realize an interactive system using the Internet.

Fig.14 shows the fusion of two metacircles. In order to scan the metacircles, we need a scanline/iso-curve intersection test. In this case, we can employ the Bezier clipping method.

Fig.15 shows an example of Chinese calligraphy defined by metacicles. Fig.16 shows a horse again defined by metacicles. We can send the images by using only a small amount of data, i.e. coordinates of the center and its density.

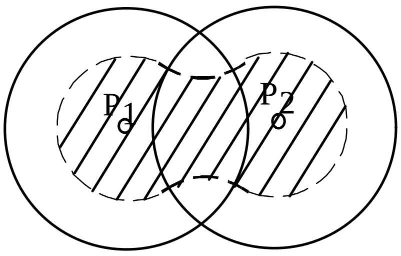


Fig.14 Fusion of metacicles

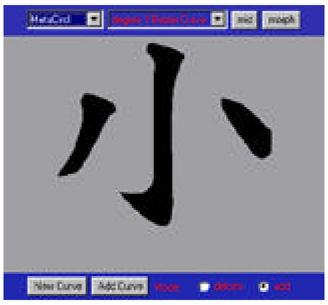
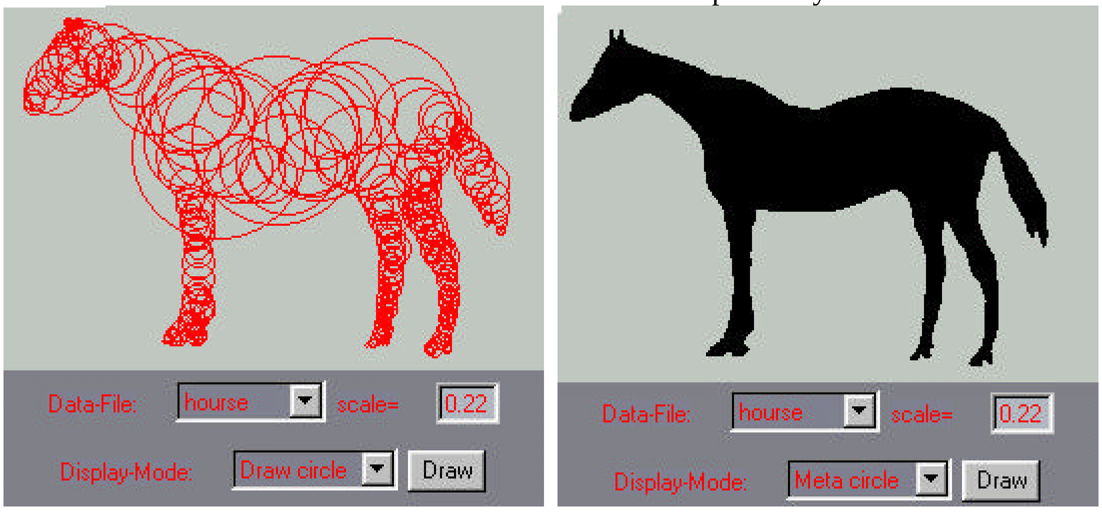


Fig.15 Java Applet for Chinese calligraphy composed by metacicles



Data-File: hourse scale= 0.22
Display-Mode: Draw circle

Data-File: hourse scale= 0.22
Display-Mode: Meta circle

(a) metacircles (b) binary image from metacircles

Fig.16 Example of metacircles.

6. Conclusion

We have introduced a display system for Bezier surfaces and metaballs using *Bezier Clipping.* The Bezier Clipping is a very powerful solver for geometric modeling and shading models. As shown in the examples, the system described here gives us photo-realistic images.

The advantages of the methods described here are as follows:

(1) Both of parametric and implicit surface can be displayed with high accuracy (i.e., without polygonization).

(2) Various shading effects for parametric surfaces can be simulated: cylindrical/curved light sources, radiosity.

(3) In the systems, parametric patches and metaballs can be displayed by a single program.

(4) Because of the effectiveness of Bezier clipping, we can develop an interactive system through the Internet.

Please refer the following URL for interactive applications of Bezier clipping.
http://www.eml.hiroshima-u.ac.jp/~nis/javaexampl/demoEng.html

Acknowledgment

I would like to acknowledge Prof. Sederberg of Brigham Young University for his contribution to our discussion on Bezier clipping.

References

[Blinn80] J.F.Blinn, "A Generalization of Algebraic Surface Drawing," *ACM Transaction on Graphics*, Vol.2, No.3 (1980) pp.235-256.

[Chung 97] J.Chung, N.Ohnishi, "Chain of Circles for Matcing and Recognition of Planer Shapes," IDY97-52,1997, pp.23-30.

[Fournier94] A. Fournier, J.Buchanan, "Chebyshev Polynomials for Boxing and Intersections of Parametric Curves and Surfaces," *Proc. of EUROGRAPHICS'94* (1994) pp.127-142.

[Kajiya82] Kajiya, J., "Ray Tracing Parametric Patches," *Computer Graphics*, Vol.16, No.3 (1982) pp.245-254.

[Kaneda96] K.Kaneda, Y.Zuyama, H.Yamashita, T.Nishita, "Animation of Water Droplet Flow on Curved Surfaces," *Proc. of Pacific'96* (1996)

[Kim95] J.Kim, D.Park, "Efficient Ray Tracing Trimmed Rational Surface Patches," *Proc. of Pacific Grapphics'95* (1995) pp.209-221.

[Lane80] J.Lane, L.Carpenter, T.Whitted, "Scan line Methods for Displaying Parametrically Defined Surfaces," *CACM*, Vol. 23, No.1 (1980) pp.23-34.

[Nishim85] H. Nishimura, M.Hirai, T.Kawai, T.Kawata, I.Shirakawa, K.Omura, "Object Modeling by Distribution Function and a Method of Image generation,", *Journal of papers given by at the Electronics Communication Conference'85* J68-D(4) pp.718-725 (in Japanese).

[Nishi90] T.Nishita, T.W.Sederberg, M.Kakimoto, "Ray Tracing Rational Trimmed Surface Patches," *Computer Graphics*, Vol.24, No.4 (1990), pp.337-345.

[Nishi91a] T.Nishita, K.Kaneda, E.Nakamae, "A Scanline Algorithm for Displaying Trimmed Surfaces by using Bezier Clipping," *The Visual Computer*, Vol.7, No.5 (1991) pp.269-258.

[Nishi91b] T.Nishita, S.Takita, E.Nakamae, "Scan Conversion of Regions Bounded by

Bezier Curves," *Proc. of CG & CAD* (1991) pp.198-204.

[Nishi92a] T.Nishita, S.Takita, E.Nakamae, “Shading Model of Parallel Cylindrical Light Sources," *CG International'92* (1992) pp.429-445.

[Nishi92b] T.Nishita, S.Takita, E.Nakamae, “Hidden Curve Elimination of Trimmed Surfaces Using Bezier Clipping," *CG International'92* (1992) pp.595-619.

[Nishi93a] T.Nishita, S.Takita, E.Nakamae, “A Display Algorithm of Brush Strokes using Bezier Functions," *CG International'93* (1993) pp.244-257.

[Nishi93b] T.Nishita, K.Fujii, E.Nakamae, “Metamorphosis using Bezier Clipping," *Pacific Graphics'93* (1993) pp.162-173.

[Nishi93c] T.Nishita, T. Shirai, K.Tadamura, E. Nakamae, ”Display of The Earth Taking into Account Atmospheric Scattering,"*Proc. of SIGGRAPH'93* (1993) pp.175-182.

[Nishi93d] T.Nishita, E.Nakamae, “A New Radiosity Approach Using Area Sampling for Parametric Patches," *Proc. of EUROGRAPHICS'93* pp.385-393.

[Nishi94a] T.Nishita, E.Nakamae, ”A Method for Displaying Metaballs by using Bezier Clipping," *Proc. of EUROGRAPHICS '94*, Vol.13, No.3 (1994) c271-280.

[Nishi94b] T.Nishita, E.Nakamae, “Method of Displaying Optical Effects within Water using Accumulation Buffer," *Proc. of SIGGRAPH'94* (1994) pp.373-379.

[Nishi96] T.Nishita, Y.Dobashi, E.Nakamae, “Display of Clouds Taking into Account Multiple Anisotropic Scattering and Sky Light," *Proc. of SIGGRAPH'96* (1996)

[Ranjian96] V.Ranjian, A. Fournier, ”Matching and Interpolation of Shapes using Unions of Circles”, EUROGRAPHICS’96, Vol.15, No.3, 1996, pp.129-142.

[Seder90] T.Sederberg, T.Nishita, “Curve Intersection using Bezier Clipping," *CAD*, Vol.22, No.9 (1990) pp.337-345.

[Seder91] T.Sederberg, T.Nishita, “Geometric Hermite Approximation of Surface Patch Intersection Curves," *CAGD*, Vol.8, No.2 (1991) pp.97-114.

[Watt90] M.Watt, “Light-Water Interaction using Backward Beam Tracing," *Computer Graphics*, Vol.24, No.4 (1990) pp.377-376.

[Whitt80] T.Whitted, “An Improved Illumination Model for Shaded Display,” *CACM*, 23, 6 (1980) pp.96-102.

[Wyvil86a] B.Wyvill, C.McPheeters, G.Wyvill, “Data Structure for Soft Objects," *The Visual Computer*, 2 (1986) pp.227-234.

[Wyvil86b] B.Wyvill, C.McPheeters, G.Wyvill, ”Animating Soft Objects," The *Visual Computer*, 2 (1986) pp.235-242.