Animation of Algebraic Surfaces

Dietmar Saupe and Matthias Ruhl

Universität Freiburg, Institut für Informatik, Am Flughafen 17, 79110 Freiburg, Germany

Summary. This contribution is a practise-and-experience report on the visualization of parameter dependent algebraic surfaces. An important example of a deformation of algebraic surfaces was proposed by Kummer in the last century. The techniques in computer graphics algorithms, software and hardware for the animated real-time display of many types of surfaces have been established in recent years. We present an application of this technology and a raytracer to the Kummer family and others. The purpose of this report is to demonstrate the feasibility of such studies by combining several pieces of readily available software and off-the-shelf hardware with only a minimal investment of extra programming. Raytracing produces high quality renderings, however, the method requires much processing time. Faster alternatives are methods yielding polygonalizations or using physically-based approaches. These provide less quality but allow near real-time user interaction. We present some examples for them.

1. Introduction

Algebraic surfaces and their deformations have been studied for more than one hundred years. Visualization of these surfaces has always been regarded important and traditionally plaster models were used for this purpose. Recently, of course, computer graphical studies have been carried out. In [8], e.g., Hanrahan investigates the raycasting approach to the rendering of algebraic surfaces. The central computational task is twofold. First, one has to efficiently convert the equations of the surface and a given ray into a single equation, i.e., a polynomial in one variable. Secondly, a numerical procedure must be employed to compute the smallest positive root of the polynomial. Hanrahan chose a method developed by Collins and Loos, which is based on Descartes rule of signs. In [13] the approach was tested with other root finding methods.

An important example of a deformation of algebraic surfaces was proposed by Kummer in the last century. It is given by a parametrized family of

fourth-order polynomials in affine coordinates x, y and z. In this deformation a double sphere is transformed into Steiner's roman surface and then into a tetrahedron. Although individual surfaces from this family had been visualized before, only just recently a sequence of (raycast) images illustrating the entire deformation was given by Barth and Endraß in [1]. In this work we present corresponding computer animations. It is the purpose of this work to demonstrate a feasible approach to rendering animations of mathematical objects suitable not only for the computer graphics specialists but also for students and researchers in mathematics without such background. Thus, we show how to combine commercial and public domain software with common off-the-shelf hardware to arrive at the desired product.

The animation is based on the original method of Hanrahan and supported by the public domain raytracer *rayshade* of Kolb. In this way we can preview the entire animation at full resolution and video frame rate, thus, eliminating the time intensive and error prone frame-by-frame recording. There are several more advantages of this all-digital approach to computer animation as compared to conventional video taping, video frame store and video laser disk: individual frames can easily be retouched, playback of arbitrarily rearranged frames is possible, scene duration is limited only by the available disk space, and animations can be transmitted over the net.

For the visualization of animated implicit surfaces there exist at least three principally different approaches: direct raytracing of the implicitly defined object, polygonalization with subsequent standard rendering of the polygonal model, and particle-based methods employing the physically-based modelling paradigm. The three methods are ordered here with respect to decreasing computational complexity as well as decreasing image quality. While the raytracing approach yields the best approximations and renderings we have that the particle-based method delivers a speed that allows near real-time performance. In this paper we briefly outline the methods and provide a couple of examples.

2. Raytracing algebraic surfaces

Raytracing is a powerful method for generating photo-realistic imagery. In traditional rendering methods primitive objects such as triangles or surface patches are projected onto a view plane and hidden surfaces need to be removed by special algorithms such as the z-buffer method. In contrast raytracing proceeds in the inverse direction. Rays are shot from a virtual camera position through pixels of a virtual screen into the 3D-scene where they may hit surfaces. The first such intersection point of the ray with a surface provides the solution of the visibility problem. With the surface normal vector and the vectors to light sources a simple illumination model yields the color to be placed at the corresponding pixel. From the intersection points more rays can be shot to the light sources to determine if the point is in the shadow of some other part of the scene. Moreover, reflected and transmitted rays can be considered yielding simulations of reflecting or translucent materials. For a broad introduction see, e.g., the book [9].

The crucial part of any raytracing method is the ray-surface intersection calculation. In the case of an algebraic surface, which is given as the zero set of a degree n polynomial $f(x,y,z)$ in three variables x, y, and z consider a ray parametrized as

$$x(t) = x_1 + x_2 t, \quad y(t) = y_1 + y_2 t, \quad z(t) = z_1 + z_2 t, \quad t \geq 0.$$

Then the ray-surface intersections are obtained as the solutions of

$$F(t) := f(x(t), y(t), z(t)) = 0.$$

After the substitution of the ray equations the result $F(t)$ is a degree n polynomial in one variable. Such substitutions — carried out by hand — are very cumbersome and error prone for all but the very simplest algebraic surfaces. However, they are easily performed using symbolic calculation programs such as MAPLE, Mathematica, or Reduce. These systems are also capable of outputting corresponding computer code. Care must be taken to optimize the resulting formulas in order to avoid unnecessary multiple evaluations of common expressions. The symbolic differentiation is also useful, namely for the generation of formulas and code for the partial derivatives of the polynomial

$f(x, y, z)$, which provide the surface normal necessary for computer graphics illumination models. In [8] a custom made LISP program was used to derive appropriate C programs.

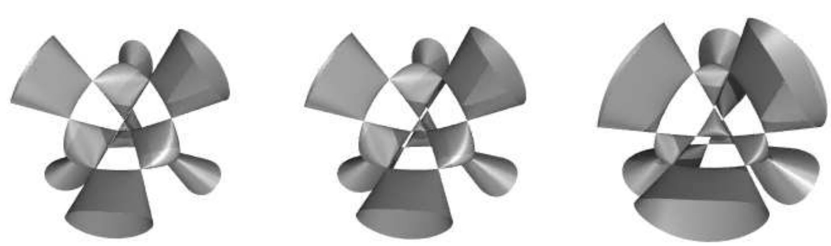
While symbolic manipulation produces the coefficients of the polynomials $F(t)$, numerical procedures are required to solve the algebraic equation $F(t) = 0$. Since only the first intersection of a ray with the surface is of interest one must be able to efficiently and accurately compute the *smallest real positive root of degree* n *polynomials* $F(t)$. There are many methods available, for example:

1. direct methods for polynomials of degree 4 or less [2],
2. Sturm sequences [14],
3. methods based on differentiation [4],
4. root bracketing with bisection based on Descarte's rule of signs [5],
5. methods based on polynomials in Bernstein form [11],
6. global (complex) methods [10].

A comparative analysis of these methods with respect to the special problem of finding the smallest positive real zeros of polynomials has not yet been carried out to the knowledge of the authors.

There are many raytracing programs available. We have used the program *rayshade* of Craig Kolb which is in the public domain. Although general algebraic surfaces are not directly supported by *rayshade*, this program is ideally suited for our task since it allows extensible geometric primitives. It is relatively easy to write an algebraic surface module consisting of a gradient calculation routine and a ray-surface intersection procedure which incorporates clipping volumes (half spaces and balls). For degree 4 surfaces we used the direct root finding methods. For higher degree surfaces the polynomial root finder internal to *rayshade* was used. It is based on Newton's method and does not guarantee the correctness of the solution. In order to provide correctness, one of the algorithms from the list above may be implemented.

An animation of a parameter dependent algebraic surface may contain several thousand frames. Since each frame can be computed independently from all others the entire computational task can be carried out conveniently on several workstations in parallel. The rendered frames could be recorded frame by frame on a video tape. However, this requires long recording times;



only a few frames per minute are possible. Today there exists hardware technology that enables an all-digital approach to the viewing of the animation that does not require an external storage medium. Individual frames can be JPEG-compressed and stored on a hard disk. The compressed files can then be loaded in real-time, i.e., at a frame rate of 25 Hz, from the disk, decompressed and written to the frame buffer for immediate viewing. Currently this requires compression/decompression hardware; the COSMO Compress board from Silicon Graphics is an example. As mentioned in the introduction this approach has many benefits as compared to traditional techniques: individual frames can easily be retouched, playback of arbitrarily rearranged frames is possible, scene duration is limited only by the available disk space, and animations can be transmitted over the net. Therefore, we think that this solution is ideal for research and education. It is simple so that nonspecialists and students may use the technique after only very limited training. The only deficiency is a slight degradation in image quality due to the lossy compression involved. In our animations we could not see such artifacts and, moreover, as computer disks are becoming bigger and faster this handicap will surely disappear in the future.

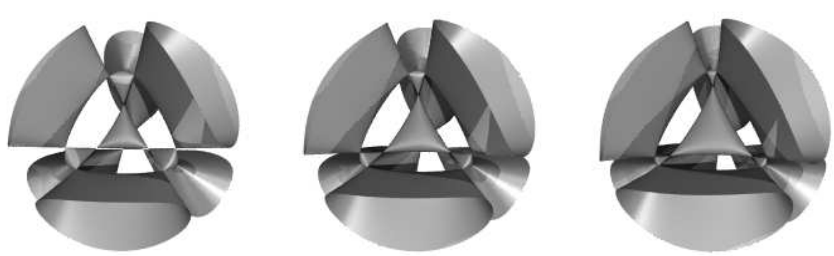
As an example we present a subset of the animation of the parametrized family of Kummer surfaces at the bottom of the pages and in the color section (Figures 4.1, 4.2). The equation is

$$(3-\nu^2)(x^2+y^2+z^2-\nu^2)^2-(3\nu^2-1)pqrs=0$$

where $\nu \in \mathbb{R}$ is the parameter and

$$\begin{aligned} p &= 1-z-x\sqrt{2}, \\ q &= 1-z+x\sqrt{2}, \\ r &= 1+z+y\sqrt{2}, \\ s &= 1+z-y\sqrt{2}. \end{aligned}$$

The surface is unbounded for $\nu^2 \geq 1$ and clipped at a sphere whose radius is chosen so that the displayed surface contains all of the singular points (except the singularity at infinity).



3. Polygonalization and physically-based interactive animation

The raytracing approach to computer animation of implicit surfaces — although good for high quality renderings — lacks interactive control which is necessary to choose proper camera paths, parameter dynamics and so on for an animation of some previously unknown surfaces under investigation. In this section we briefly sketch two alternative approaches to rendering algebraic surfaces that may provide the necessary speed for interactive control. The first one is polygonalization of implicit surfaces. In this more traditional computer graphics technique a collection of approximating polygons is generated which are well suited for rendering by means of common graphics hardware. There exists a large body of literature on this topic, see, for example, [3]. In our implementation [12] we use the marching cubes algorithm to quickly derive a first polygonalization. To iron out local insufficiencies the vertices can be automatically moved around by means of a spring-mass model with surface constraints yielding a better triangulation in a few time steps, see figure 3.1. In this application, running under Open Inventor on an SGI, a suitable view of a surface can be selected under interactive control after which a corresponding parameter input file for the rayshade program can automatically be produced. Or the raytracer can directly be invoked to render the interactively defined scene.

A more recent technique considered here for interactive control of implicit surfaces is the physically-based dynamic sampling method developed by Witkin and Heckbert [15] which we apply for algebraic surfaces. The method uses particle systems. A simple constraint equation locks a set of particles onto the surface while the particles and the surface are allowed to move. Local repulsion is used to make the points, called *floaters*, spread evenly across the surface. By varying the radius of repulsion adaptively while fissioning and killing particles based on the local density, good sampling distributions can be achieved rapidly. The work [15] focussed more on *modelling* of implicit surfaces while we are more interested in a priori defined parameter-dependent surfaces. Moreover, our surfaces have singularities and are unbounded (or clipped to a finite volume resulting in a boundary curve) in contrast to those in [15] which are single connected components without singularities or bound-

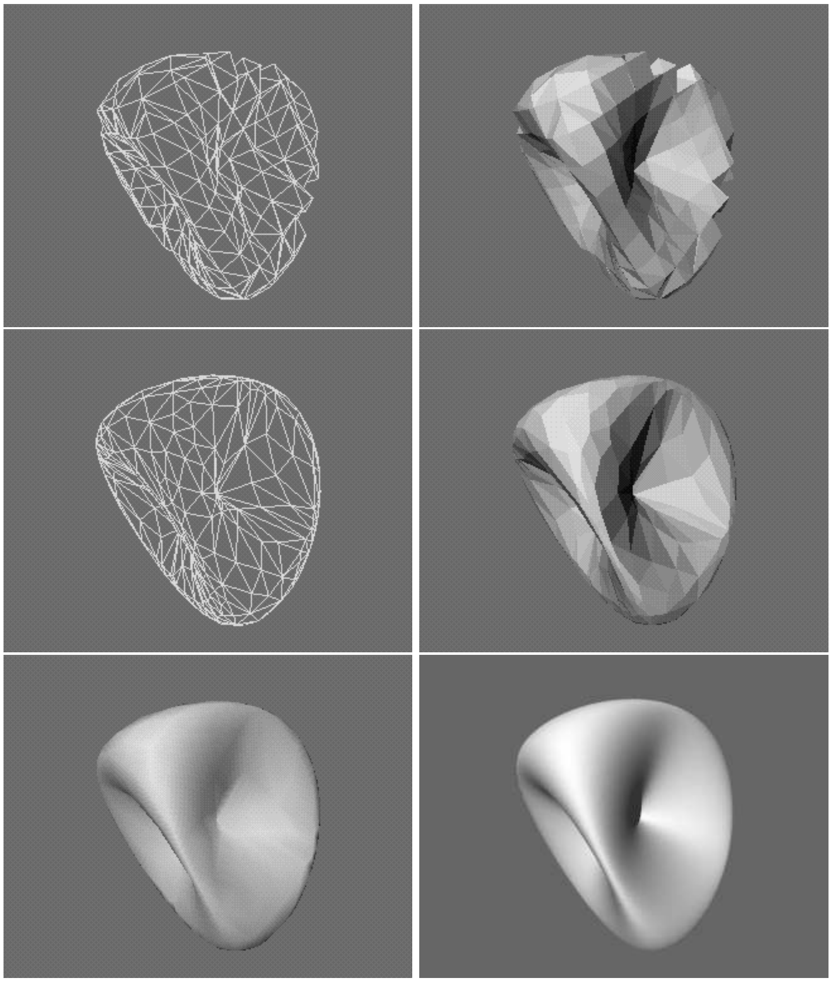
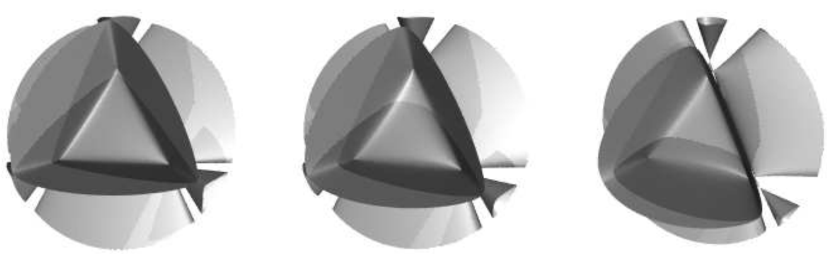


Fig. 3.1. Steiner surface polygonized, improved and shaded. The marching cubes algorithm yields the top row images, after relaxation of the mesh vertices the improved center row representations are obtained. The bottom row shows the result Phong shaded (left) and raytraced (right) (© A. Rösch).



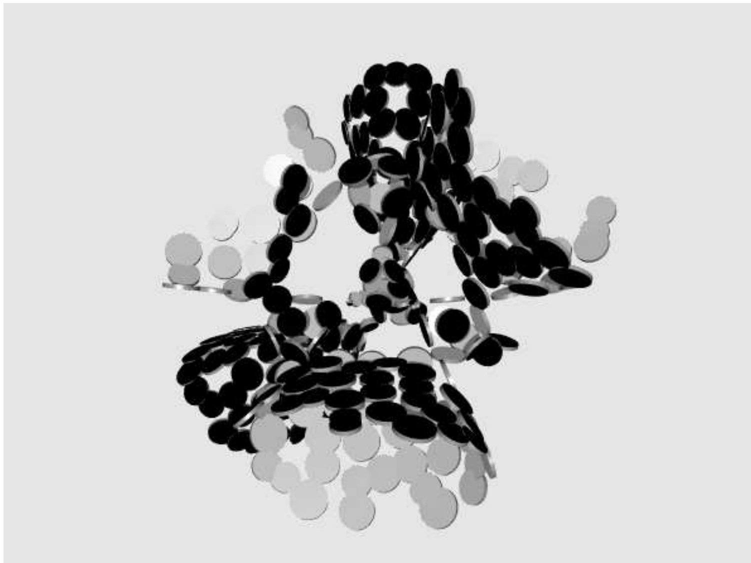


Fig. 3.2. Kummer surface at $\nu^2 = 1.5$ with a few particles not yet at equilibrium.

ary. To cope with these difficulties we modify the physical model underlying the mechanism of floaters by introducing boundary effects and adapt the radii of repulsion to local surface properties as well as to the particle density. With these modifications we can achieve near real-time sampling and rendering of deforming implicit algebraic surfaces. Details will be presented in a forthcoming paper. Figures 3.2, 3.3 and the color section (Figure 4.3) show some examples. These renderings show the floaters as shaded disks with normal vectors obtained from the gradients of the surface's defining function F. The radii of the disks are taken proportional to the particles' repulsion radii. Particles viewed from the 'back side' are shown in light gray. The program starts by randomly inserting a number of particles into the scene which quickly settle on the surface, then spread over the surface and start fissioning yielding an overall acceptable density.

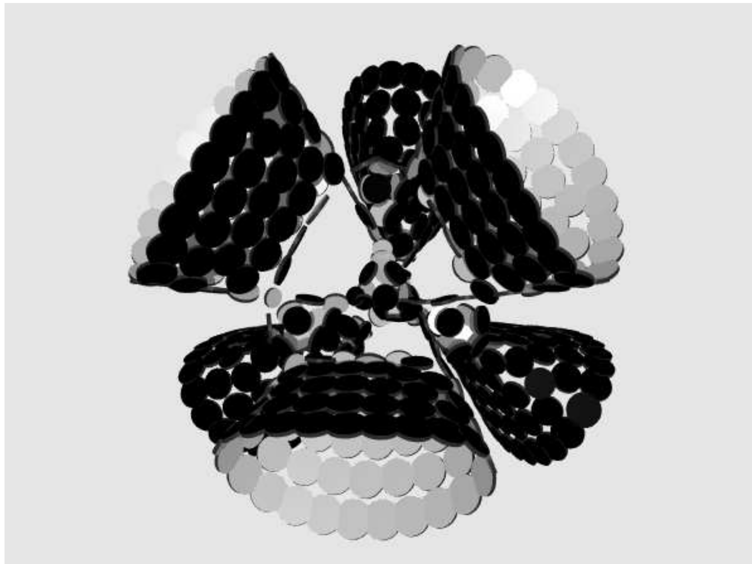
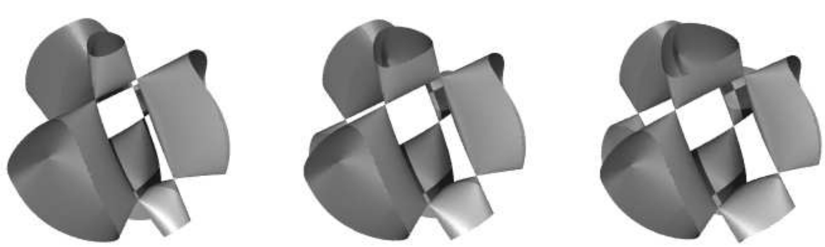


Fig. 3.3. Kummer surface at $\nu^2 = 1.5$ with more particles and at equilibrium.

4. Conclusion

In Section 2. we have shown the feasibility of raytraced animations by combining several pieces of readily available commercial and public domain software and off-the-shelf hardware with only a minimal investment of extra programming. Naturally, the achieved quality cannot be as perfect as obtained with customized algorithms such as in [6] or [13]. Artifacts can be seen, e.g., as creases inside the glass Klein bottle in the color section. They are due to insufficiencies in the polynomial root finders we used. In Section 3. we have shown how to overcome the limitations in the time complexity of the raytracing method by using polygonalization or physically-based methods, thereby providing tools for interactive investigation of algebraic surfaces. Of course, this comes at the cost of increased programming efforts and a degradation in image quality.

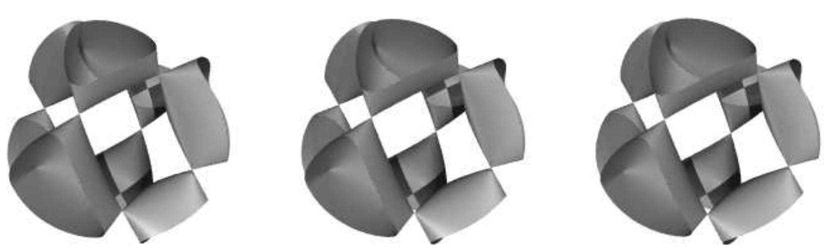


Acknowledgment

We thank Angela Rösch for providing the images in figure 3.1.

References

1. W. P. Barth and S. Endrass, *Deformation, a series of computer pictures,* Internal Report, Mathematisches Institut, Universität Erlangen, 1995.
2. I. N. Bronstein and K. A. Semendjajew, *Taschenbuch der Mathematik,* Harri Deutsch, Zürich, 1973.
3. J. Bloomenthal, *Polygonalization of implicit surfaces,* Computer-Aided Geometric Design 5,4 (1988), 341–355.
4. G. E. Collins and R. Loos, *Polynomial real root isolation by differentiation,* SYMSAC 1976, 15–25.
5. G. E. Collins and R. Loos, *Real zeros of polynomials,* Computing Suppl. 4 (1982), 83–94.
6. S. Endrass, *Flächen mit vielen Doppelpunkten,* DMV-Mitteilungen 4/95 (1995), 17–20.
7. G. Fischer, *Flächen · Körper · Licht, Mathematische Modelle,* Vieweg, Braunschweig, 1990.
8. P. Hanrahan, *Ray tracing algebraic surfaces,* Comp. Graph. 17 (1983), 83–90.
9. A. Glassner (ed.), *An Introduction to Ray Tracing,* Acad. Press, London, 1989.
10. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C,* Cambridge University Press, Cambridge, 1992.
11. A. Rockwood, K. Heaton, and T. Davis, *Real-time rendering of trimmed surfaces,* Computer Graphics 23,3 (1989) 107–116.
12. A. Rösch, *Interaktive Visualisierung implizit definierter Flächen,* Diplomarbeit, Universität Freiburg und Universität Erlangen, 1996.
13. O. Stelzner, *Visualisierung algebraischer Flächen mit Raytracing-Verfahren,* Diplomarbeit, Fachbereich Mathematik, Universität Bremen, 1990.
14. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis,* Springer-Verlag, New York, 1980.
15. A. Witkin and P. Heckbert, *Using particles to sample and control implicit surfaces,* SIGGRAPH'94, Comp. Graph. Ann. Conf. Ser. (1994), 269–277.



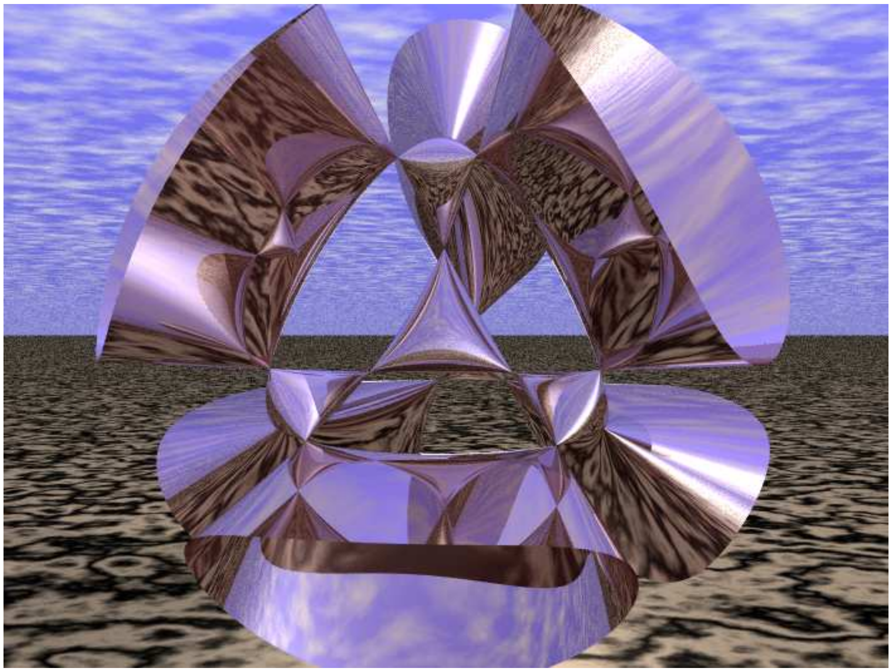


Fig. 4.1. Raytraced Kummer surface

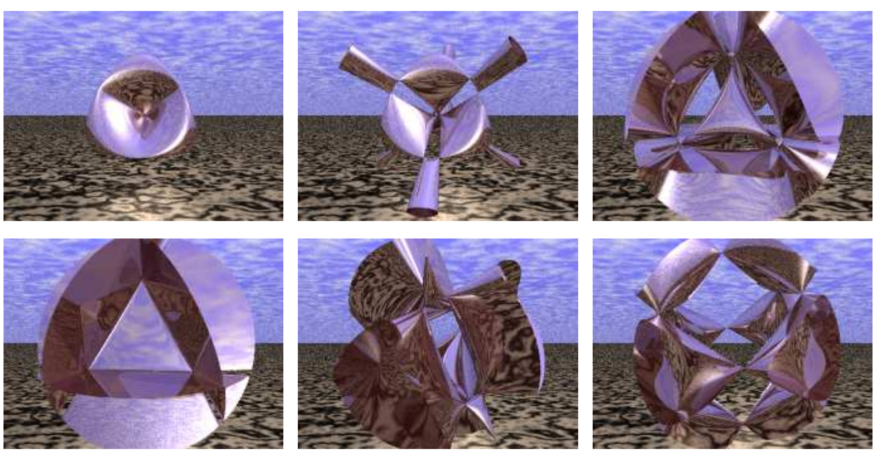


Fig. 4.2. Six frames from the animation

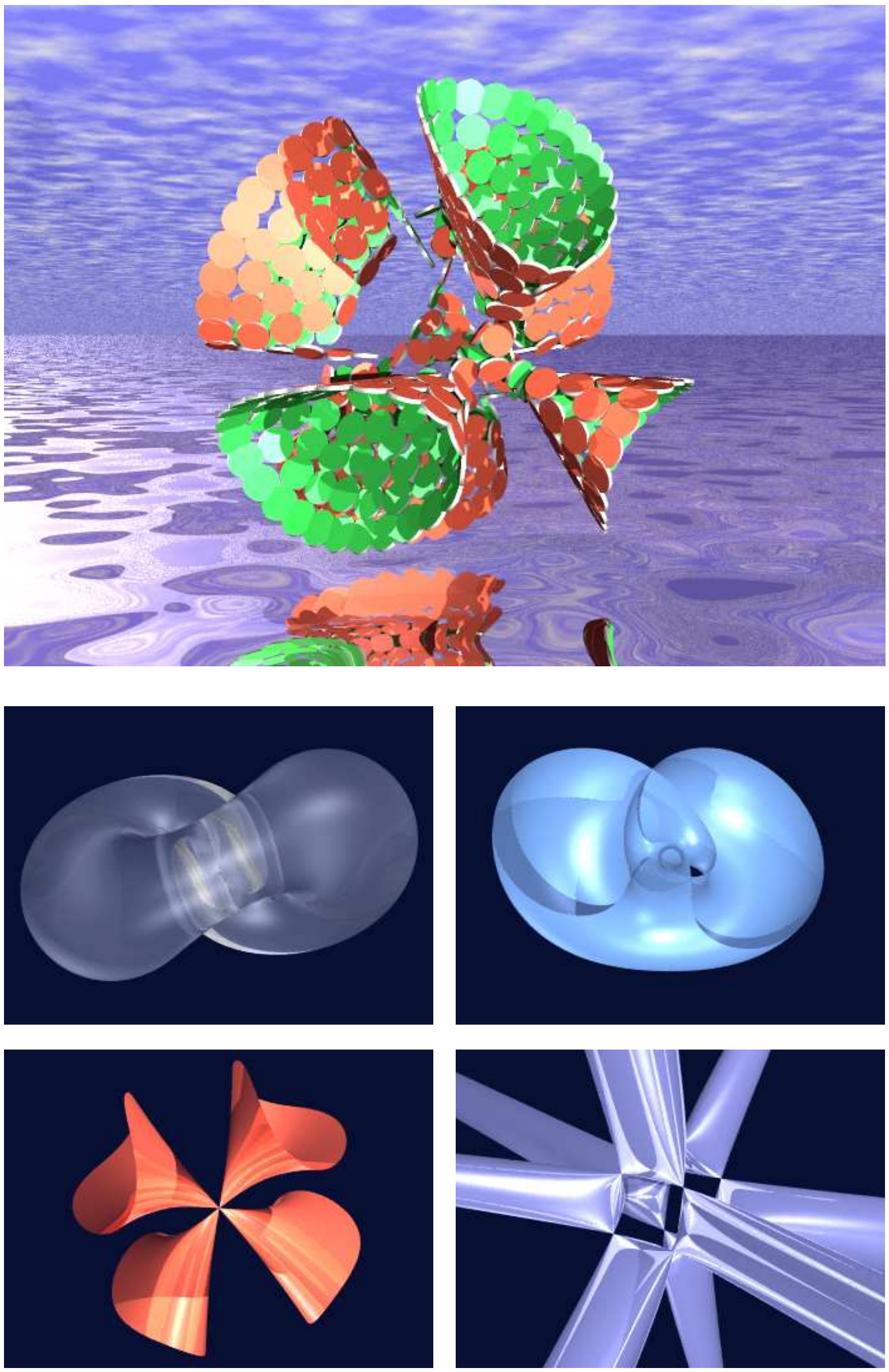


Fig. 4.3. *Top:* Raytraced image of floaters approximating a Kummer surface, *Center:* Klein Bottle, *Bottom:* Degree four surfaces