Techniques for Visualizing Fermat's Last Theorem: A Case Study

A.J. Hanson P.A. Heng B.C. Kaplan
Department of Computer Science and the Center for Innovative Computer Applications
Indiana University, Bloomington, Indiana 47405

Abstract

We describe some mathematical approaches and computer graphics techniques for illustrating concepts related to Fermat's Last Theorem. We present a selection of visualization methods, and describe their interactions with the available software systems and with the mathematical subject matter.

1 Introduction

Special techniques are required to effectively exploit computer graphics for the visualization of concepts that are important in abstract mathematics. In this paper, we describe observations made in the process of creating a three minute computer animated videotape dealing with some elementary aspects of Fermat's Last Theorem, a famous unsolved problem in number theory.

Our approach to the representation of the different concepts presented in the video was influenced by many factors: the available hardware, real and perceived constraints of the available software, constraints imposed by the video medium, and a number of peculiarities and features of the mathematical domain itself. In the following sections, we describe the mathematical concepts that we tried to depict, our experiences with the software systems that played a part in our efforts, some specific successful visualization techniques, and some unexpected mathematical insights that we noticed.

2 Mathematical Ideas of Fermat's Last Theorem

In the margin of his copy of *Arithmetica* by the Greek number theorist Diophantus of Alexandria, the 17th century mathematician Pierre de Fermat stated that

$$x^n + y^n = z^n \tag{1}$$

cannot be solved with non-zero integers (x, y, z) *for any integer power* $n > 2$,

and then wrote

I have discovered a truly marvelous proof that this margin is too small to contain.

No one has yet succeeded in proving or disproving Fermat's declaration, although it is known to be true for all integers n below an astronomical number. Fermat himself never referred to this "marvelous proof" in any of his other writings, so no one knows whether he had a proof or not. For a very readable account of the early history of the theorem, see Edwards [3]; a more mathematical survey of the theorem's status (no longer quite up to date) is provided by Ribenboim [5].

In our video, we touched on only a handful of the vast variety of mathematical ideas that are potentially relevant to the Fermat theorem. We naturally chose those that were both relatively understandable to a scientific but nonmathematical audience, and adaptable to visually interesting graphical displays.

Explicit parametric solutions of the equation. We now show how to obtain a parametric solution for Eq. (1) in a particular local coordinate system. First, we observe that the expressions

$$\begin{aligned} u_1 &= \frac{1}{2}(\exp(a+ib)+\exp(-a-ib)) \\ &= \cos b \cosh a + i \sin b \sinh a \qquad (2) \\ u_2 &= \frac{1}{2i}(\exp(a+ib)-\exp(-a-ib)) \\ &= \sin b \cosh a - i \cos b \sinh a, \qquad (3) \end{aligned}$$

where $0 \le b < 2\pi$ and a ranges over all real values, behave like complex extensions of $\cos(b)$ and $\sin(b)$, that is

$$(u_1)^2 + (u_2)^2 = 1. \qquad (4)$$

Next, we define

$$\begin{aligned} z_1 = \frac{x}{z} &= s_1(u_1)^{2/n} \qquad (5) \\ z_2 = \frac{y}{z} &= s_2(u_2)^{2/n}, \qquad (6) \end{aligned}$$

where s_1 and s_2 are nth roots of unity of the form

$$s(k,n) = \exp(2\pi i k/n) \qquad (7)$$

for integers $0 \le k \le (n-1)$. Rewriting Eq. (1) as

$$(z_1)^n + (z_2)^n = 1, \qquad (8)$$

and substituting the right hand sides of Eqs. (5,6) for z_1, z_2, we see that Eq. (4) implies that Eq. (8) is identically satisfied. Since there are n possible distinct values for each of the phases s_1 and s_2, the entire finite surface is obtained by patching together the n^2 different quadrilateral regions found by evaluating the pair (z_1, z_2) for $0 \le b \le (\pi/2)$ and some practical range of a.

This technique is well known in computer graphics in the real domain (see below); we are unaware of whether or not this approach to representing the complex surfaces has been noted in the mathematical literature.

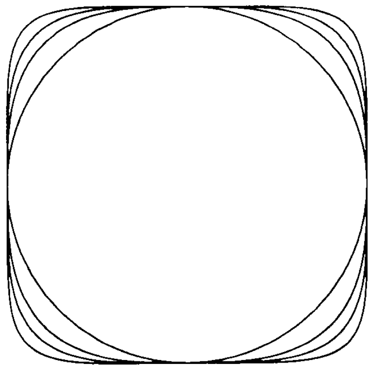


Figure 1: Superquadric 2D shapes for several values of n.

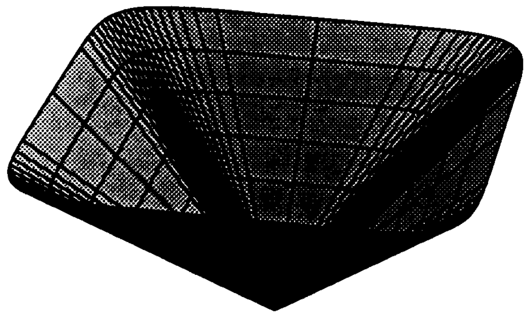


Figure 2: The 3D Fermat equation for $n = 7$; constant z cross sections of this cone-like surface are 2D superquadrics.

Superquadrics and cones. When $a = 0$ and $0 \le b \le (\pi/2)$, in Eqs. (2,3), the family of real 2D solutions to $|z_1|^2 + |z_2|^2 = 1$ with $s_1 = \pm 1$, $s_2 = \pm 1$ are known as *superquadrics* in the computer graphics literature [1]; we plot the resulting curves in Figure 1 for various values of n.

The full 3D form of Eq. (1) is cone-like, interpolating between a true cone for $n = 2$ and an inverted, square-based pyramid as $n \to \infty$. The planar cross sections of the 3D equations are of course 2D superquadrics; Figure 2 shows the 3D shape for $n = 7$.

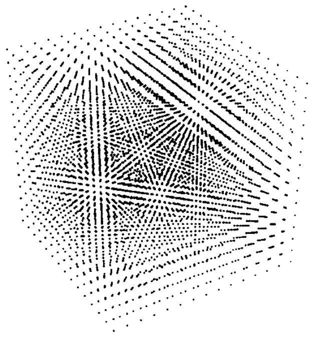


Figure 3: A perspective view of the cubic integer lattice.

Integer crossings of the curve. The fundamental statement of Fermat's last theorem translates graphically into the following observation: if we vary the superquadric exponent n continuously with $n > 2$, so that the curve with a fixed integer value of z sweeps a portion of the xy plane, the curve of necessity passes through many integer pairs (x, y), but *none* of these intersections occurs for integer n. We also see that for any z, there is always a particular value of n, $n = \log 2/(\log z - \log(z-1))$ (corresponding to the point $(z-1, z-1, z)$), beyond which no more integers can be intercepted. The 3D space of integers for which intersections of the curve might be sought can be viewed as a cubic lattice, shown in Figure 3.

Inverting the deformation transform. In computer graphics, Barr [2] introduced a technique for ray tracing deformed superquadrics by inverting the original deformation transformation and tracing the deformed ray. We may perform an analogous transformation on the integer grid,

$$x' = z\left|\frac{x}{z}\right|^{(n/2)}, \; y' = z\left|\frac{y}{z}\right|^{(n/2)}. \qquad (9)$$

We then see the *integers* sliding into the center of the plane as $n \to \infty$, as illustrated in Figure 4.

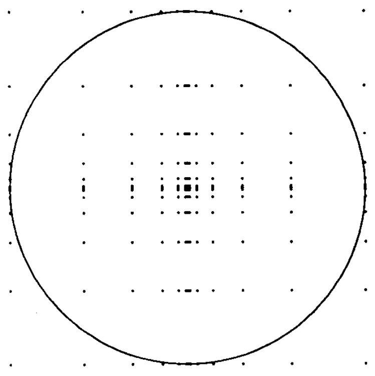


Figure 4: The deformed integer grid at $z = 7$ corresponding to the deformed conical curve of Figure 2.

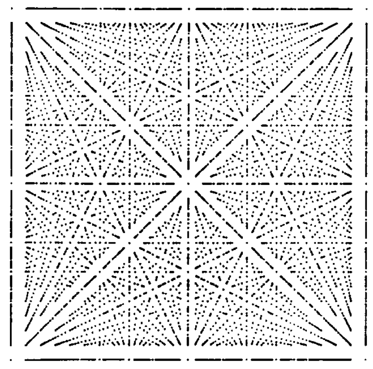


Figure 5: A portion of the grid of rational points in the unit square for denominators up to 24.

Projecting the homogeneous equation to the unit square. For many mathematical purposes, it is preferable to look at Fermat's equation not as a homogeneous equation for a cone in 3-space, but as the 2D inhomogeneous equation found by dividing both sides by z^n. In this framework, the curve deforms from the unit circle to the unit square as n increases, and our 3D integer lattice is replaced by the set of rational points in the unit square. We show the set of such points with denominators up to 24 in Figure 5.

Extending the real curve to a surface in two complex dimensions. The full complex equations (5,6) produce a surface of one complex dimension in the space CP2 of two complex dimensions (that is, a surface of two real dimensions embedded in a real four dimensional space). To display this surface, we therefore had to choose particular ways of projecting the object from four dimensions into three, and then into two dimensions for computer graphics display. We accomplished this by transforming the two imaginary components of z_1 and z_2 into a single three-dimensional value z; the full $4D \Rightarrow 3D$ projection then became

$$\begin{aligned} x &= Re(z_1) \\ y &= Re(z_2) \\ z &= \cos\alpha\, Im(z_1) + \sin\alpha\, Im(z_2). \end{aligned} \tag{10}$$

The finite part of the resulting surface for $n = 3$ looks like Figure 6a. Typically, we chose a different color for each value of the phases s_1, s_2 in Eq. (7), so that the surface appeared as a patchwork quilt of n^2 distinct square pieces sewn together to make the full 2D surface. The fixed points of the cyclic group were then distinguishable as the centers of "pie charts" with n wedges meeting at a single point on the surface.

Global topology and transforming to the generalized Riemann sphere. For each integer power n, the complete Fermat surface is a closed manifold with the topology of a 2-sphere having g handles attached. g is known as the *genus* of the surface and, according to the genus formula [4], has the value

$$g = \frac{(n-1)(n-2)}{2}. \tag{11}$$

The Euler characteristic of the surface is related to the genus by $\chi = 2 - 2g$.

To depict these closed surfaces for computer graphics purposes, we perform the following transformation:

$$\begin{aligned} &u_1 = x_1/\phi,\ u_2 = y_1/\phi, \\ &u_3 = x_2/\phi,\ u_4 = y_2/\phi, \\ &u_0 = x_0/\phi, \end{aligned} \tag{12}$$

where $x_1 = Re(z_1)$, $y_1 = Im(z_1), x_2 = Re(z_2)$, $y_2 = Im(z_2)$, $x_0 = ((x_1)^2 + (y_1)^2 + (x_2)^2 + (y_2)^2)/2r$, and $\phi = 1 + x_0/2r$. Since

$$(u_1)^2 + (u_2)^2 + (u_3)^2 + (u_4)^2 + (u_0)^2 = 1,$$

Eq. (12) parameterizes a 4-sphere embedded in a 5-dimensional space. This transformation is the analog of the transformation from CP1 to the Riemann sphere S^2, except that CP2 is *not* precisely equivalent to S^4, while CP1 *is* equivalent to S^2. An example of the result of the projection of this transformation to 3D from 5D is shown in Figure 6b. We have considered several other approaches, some more topologically consistent than this one, for representing and depicting the surface within a finite volume; we hope to investigate these alternatives later.

The surfaces that we deal with intersect in extremely complex ways in the 3D projection, but they are in fact nonintersecting in 4D. Thus in 4D it is relatively simple to get a complete topological description in terms of a unique and nonredundant set of vertices, edges, and faces giving an Euler characteristic $\chi = V - E + F = 2 - 2g$ agreeing with Eq. (11).

3 Constraints and Features of the Software Environment

In a project of this type, the result and the techniques used can clearly be driven as much by the software capabilities (and the ability of the designers to exploit documented and undocumented features of the software systems) as by the methods one would ideally like to use for visualizing the problem.

The whole visualization process that was undertaken in the production of our video can be divided into three parts. The first part was the development of the script and explicit algorithms for generating the desired mathematical objects; these elements were laid out in a *Mathematica* notebook (running on a Macintosh IIcx) that served as a storyboard and a rough draft of the animation sequences. The second part dealt with animating integer grids and was done by using Wavefront Technology's design and rendering package. The third part, which focused on animating surfaces in CP2, was carried out using an object-oriented graphics library based on Stardent Computer's Doré — *Dynamic Object Rendering Environment*. All rendering and animation were done on a Stardent Titan interfaced to a SONY VO-9850 single-frame animation video recorder using a Lyon Lamb MiniVas controller, with controlling software provided by Wavefront and Stardent.

3.1 Experience with Mathematica

The first task in creating the animation was to develop a storyboard showing approximate timing and the expected content of typical frames from each scene. We used Wolfram Research's *Mathematica* system on the Macintosh for this purpose, as well as for studying various visual approaches to the material. Equations needed by the animators were checked in the storyboard, which then served both as a source of explicit algorithms and as a rough draft for the animation. We were very impressed by *Mathematica*'s ability to handle almost anything we needed — we found it to be a very powerful tool for our purposes, with only a handful of drawbacks.

Strengths. Among the particular strengths of *Mathematica* were:

- Complex arithmetic is handled transparently. This was very useful given our emphasis on experimenting with the Fermat equations in the complex domain.
- The parametric plotting packages both in 2D and 3D were extremely flexible and easy to use. This made small experiments in visualizing aspects of the equations very convenient. Figures 1, 2, 3, 4, and 5 were all generated and printed by *Mathematica*.
- Numerical computation is handled very carefully in *Mathematica*. Thus some of the problems we had when computing surfaces using standard programming languages could be circumvented in *Mathematica*, at the price of reduced speed.

Drawbacks. Among the drawbacks we experienced in our 8 megabyte Macintosh IIcx environment were the following:

- While graphics animation is supported in *Mathematica*, there is no way to specify a *sequence of times* for a set of graphics frames. This prevented us from being able to test the exact timing of the rough draft storyboard specified by a set of representative still frames.
- Memory limitations caused unending problems and loss of time; virtual memory support and graceful recovery from memory overflows would have been invaluable. We were prevented from discovering certain basic properties of the equations in *Mathematica* because we never had enough memory to finish; the discoveries were made instead after much programming effort in the Doré graphics package described below. We had originally expected to be able to carry out all such investigations of mathematical properties directly within *Mathematica*.
- The speed of 3D graphics is painfully slow as well as being very memory intensive even for very simple objects. It would have been very useful to have some sort of support for high-speed 3D graphics once the polygon tables had been reduced to pure numbers; in principle, *Mathematica* should be able to know that a polygon table has been evaluated numerically, thus enabling high-speed 3D graphics routines to be used on the numerical results.

3.2 Experience with Wavefront

One member of the team was fresh from an intensive training course given by Wavefront Technologies. Thus we felt confident using

Wavefront for certain types of rendering tasks that were familiar from the course and the documentation. However, we should note that other, undocumented, Wavefront features are covered in a more advanced course we could not commit resources to attend.

Wavefront was designed for commercial animation, not for visualization, so naturally Wavefront handles those aspects of animation that are used in commercial applications (i.e., camera movement, special effects, etc.) better than it does mathematical objects. Wavefront has excellent tools for creating visual excitement in an animation, something that Doré cannot easily match. In particular, the user interface for the Wavefront choreography program *Preview* is one of the best available.

Tasks Attempted. Among the specific tasks attempted with Wavefront for the video were the following:

- Provide visual support while the introductory narration explains the history and definition of the theorem.
- Give the viewer a feeling for the Fermat equation by conducting a visual tour around and through the surface as n changes from 2 to infinity. (See Figure 2.)
- Show visually how, even though the graph goes through integer points for various values of n, this *never* occurs when n is an integer > 2.
- Show how an inverse transformation of the integer grid can be used as an alternate means of viewing the graph of the equation as n varies. (See Figure 4.)
- Show how a camera motion transforms the 3D integer lattice into the rational points of the 2D unit square. (See Figure 5.)
- Tie all parts of the animation together using the Wavefront *Compositor* to create titles, dissolves, and other special effects. We found this essential as a substitute for traditional video special effects equipment which was not available to us.

Advantages. We found that good visual effects could be achieved using our knowledge of Wavefront for the following sorts of images:

- Simple graphs not involving complicated mathematics.
- Linear and spline movements between camera positions.
- Visually exciting camera moves to help the viewer get a real "feel" for the geometry of the surfaces.
- Compositing dissolves from one scene to another.
- Special effects such as transparency, textures, and changes in camera projection types from perspective to orthographic.

Constraints. However, for the particular mathematical domain at hand, the methods available to us in Wavefront presented the following difficulties:

- We would have found a "compiled-in" representation for spheres, cones, splines, and other smooth, nonpolygonal surfaces to be extremely useful.
- We needed automatic retessellation of objects depending on the camera distance. Figure 7a shows an anomaly resulting from the need to use one fixed sphere tessellation to represent mathematical points.
- Mathematical objects such as points, lines, and planes could not be represented in Wavefront as idealized graphical icons whose size did not change with distance from the camera.
- Wavefront's built-in interpolation methods did not support many of the specialized interpolations among key frames that we

needed (e.g., perspective transformations of the camera focal length tied in tandem with an overall scene scaling that kept one object the same size in the view field).

- Since Wavefront deals with "object files" that are polygon lists, simple mathematical objects, such as a $20 \times 20 \times 20$ grid of spheres, had to be represented as astronomically large object files. (See, e.g., Figure 3.)

3.3 Experience with Doré

We quickly found that Doré was a much more appropriate tool to use than Wavefront for examining our complex parametric surfaces in CP2. However, we also found that Doré imposed its own undesirable constraints, many discovered while doing video recording.

Tasks Attempted. We implemented the following kinds of animation with Doré in order to visualize the complex surfaces:

- The easiest animation to implement was the "cutaway" view. This was accomplished simply by moving the front clip plane back and forth. More complex styles of cutaway would clearly have been useful.

- Surface sweeping was another good way of visualizing these mathematical objects. By gradually increasing the a bounds in the parametric equations (2,3), we could see the surface grow from thin slices to its full topology. Changing the zoom factor and viewpoint while the object grew made the growing process even clearer. However, this technique was time-consuming because the vertices had to be recomputed at each step.

- Another interesting parameter animation was what we called "r-animation." By varying the value of r in the transformation (12) taking the equation to a 4-sphere of radius r, we could see the surface evolve from an open surface to a compact one. This animation was much better from some viewing angles than others; we found these angles by rotating the surfaces interactively. Interactively testing out various ideas was a critical part of our approach with Doré.

Constraints. We encountered a number of difficulties peculiar to the Doré graphics library. Among these were:

- **Dissolves.** During video recording, we had no natural utility function available in Doré to do dissolves between two different scenes. The only simple way to do this was to fade the first scene to a black screen, switch to a blacked-out second scene, and gradually increase the light intensity back to the normal value. A tedious alternative was to do X-window dumps of both Doré scenes, convert the file formats, and use the Wavefront *Compositor*.

- **Transparency.** Transparency worked unexpectedly well with the dynamic renderer in Doré. However, there was no way to change smoothly between an opaque surface and a transparent surface. In Doré, a surface with transparent intensity equal to 1 is totally transparent, but a surface with transparent intensity equal to 0 is not totally opaque as we expected. It already looks rather transparent.

- **Single scenes.** The standard user interface provided with the Titan demonstration package includes many nice commands, buttons and knobs for interactively manipulating scenes in a single window. Support for multiple windows in the user interface would have been useful.

- **Lighting.** The effectiveness of particular views was strongly influenced by the lighting choices; unfortunately, the default user interface does not allow the lighting positions to be changed interactively. This capability would have given much needed flexibility.

- **Animation.** The default user-interface controls for orientation choices are awkward to use in an exploratory manner, and there is no support for interpolation or splining among orientations or viewpoints. To accomplish these effects, we had to add several commands by hand to the user interface code, including a command to read other commands from a file for scripting. This ultimately allowed us to write a single script for the entire video recording run. Improving the animation facilities would be very desirable.

- **Patch glitches.** When the integer n is greater than 8, there are gaps in between patches; this appears to be due to rounding error in computing the vertices of those patches, as well as some "butterfly twists" of square patches where the surface is rapidly varying. This phenomenon is illustrated in Figure 7b. The gaps are reduced by substantially increasing the patch grid size, but this is very time-consuming. As a result, we used $n = 6$ as the most complicated case in our video recording.

Advantages of Doré

- Doré programming itself is object-based, so it was quite easy to build an object-oriented graphics library based on it. One of the classes that we used for this project was an analytical surface object designed for representing parametric equations. There are methods associated with this class of object for setting various parameters on the surface; diffuse color, transparency, bounds for parameters in the equation, mesh size, and many other features can all be set or changed by sending a message to the object.

- The library has classes of objects such as buttons and sliders that can easily be added to manipulate the graphics. We used many such controls to *interactively* determine the parameter settings that were used in the final animation script. The same controls can be used for interactive visualization as well.

- The dynamic renderer in Doré is quite fast. Consider an image made up of 36 patches, each of which contains more than 100 polygons and is rendered with features such as Gouraud shading, transparency, and specular highlighting, together with frequent sin, cos, $\tan^{-1}$, cosh, sinh, and exponential function calls. It took approximately 8 seconds to generate such an image using only one Stardent Titan processor. When the values of n and the mesh detail are small, we can have almost real time animation. Furthermore, we can use options such as backface culling, flat shading, wireframe representation and multiple processors to make the program run even faster.

4 Useful Visualization Methods

The following techniques are among those that we used in the videotape or found very helpful for visualizing mathematical data:

1. **Cutaway surfaces.** Cutaways of self-intersecting or centrally detailed structures can be accomplished almost trivially by adjusting the position of the front clip plane. Arbitrary "negative" cutaway volumes would give more flexibility. Another useful cutaway variant not used in our video is to divide the surface into ribbons and cut out alternate ribbons, so that one can see the interior through the missing strips.

2. **Surface evolution.** By animating the bounds of the parameters used in the parametric equation, we can see how the surface evolves.

3. **Transparency.** The interior properties of the object can be seen using transparency, whose effectiveness is greatly enhanced by

judicious choice of view angles and rotational motion.

4. **Rotations.** Rotating the mathematical object in 3D as well as in 4D (or higher dimensions if appropriate) is very effective in generating additional spatial intuition about the object due to the extra information carried by motion parallax cues.

5. **Focal length.** Varying 3D or 4D (or higher dimensional) focal length parameters generates a hierarchy of foreshortening cues that help the viewer to develop a feeling for the relative spatial position of objects that have been projected from a high dimensional space down to the computer graphics screen.

6. **Model parameters.** Varying the model parameters is useful to understand such phenomena as the limiting behavior of the mathematical object. For Fermat's equation, for example, we can watch the true cone change to a pyramid as $n \rightarrow \infty$.

7. **Compactification.** For surfaces whose defining equations pass through the manifold at infinity, a view of the global topology can be achieved by mappings that bring all the points, including those at infinity, within a compact sphere.

5 Mathematical Serendipity

While carrying out this project, we encountered a handful of pleasant surprises when the computer graphics techniques and the mathematical subject matter joined together in unexpected ways. Among these were the following:

1. The projection from the 3D grid of integers used to test Eq. (1) to the 2D grid of rational numbers is accomplished by a simple viewpoint change! That is, certain properties of projective geometry in mathematics can be visualized as a graphical operation consisting of a simultaneous camera motion and a focal length change.

2. When finite size spheres were used to represent grid points, the perspectively projected image of the rational numbers just described has the following property: spheres with larger distances in the denominators of their projective transformations are displayed smaller than spheres closer to the camera. Thus, families of equivalent ratios (e.g., (3,4), (6,8), (9,12)) are shadowed by the nearest point of the family, i.e., the one with the smallest denominator.

3. We had to display complex surfaces made of dozens of patches with intricate local topology, multiple intersections, and multiple local common points. We color-coded the patches by their complex phase for visual interest. We then noticed the accidental side-effect that we could easily see when corners of many distinct square patches shared a common point — a fixed point of the action of the cyclic group.

4. The topology of the surface as a collection of patches was never computed explicitly; from a computer graphics standpoint, we just added each patch to the graphics context as an independent object. The "sewing together" of the patches to represent a complete analytical surface was accomplished automatically by the graphics package without our having to work it out in detail.

Acknowledgements

Among the many people who contributed to this effort we would like to thank Dennis Gannon, Rick McMullen, Eric Ost, and the staff of CICA (the Indiana University Center for Innovative Computer Applications). Special thanks go also to John Ewing and Chuck Livingston of the Indiana University Department of Mathematics, and to Amy Thomas of WTIU.

References

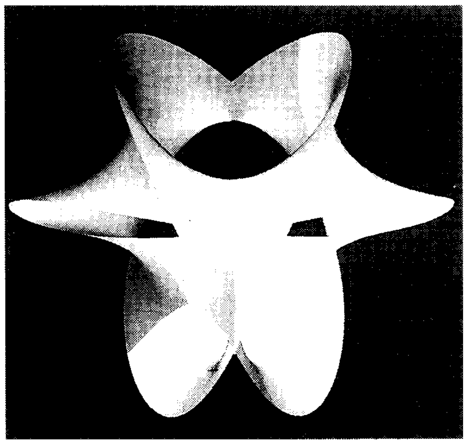
[1] A.H. Barr, "Superquadrics and Angle-Preserving Transformations," *IEEE Computer Graphics and Applications* **1**, 1981, 11–23.

[2] A.H. Barr, "Global and Local Deformations of Solid Primitives," *Computer Graphics* **18**, 1984, 21–30.

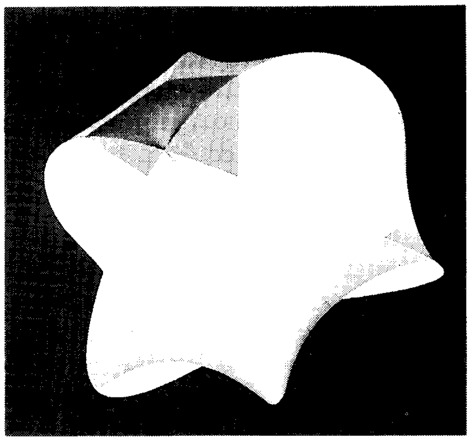
[3] H.M. Edwards, "Fermat's Last Theorem," Scientific American (October, 1978).

[4] P. Griffiths and J. Harris, *Principles of Algebraic Geometry* (Wiley, 1978).

[5] P. Ribenboim, *13 Lectures on Fermat's Last Theorem* (Springer-Verlag, New York, 1979).

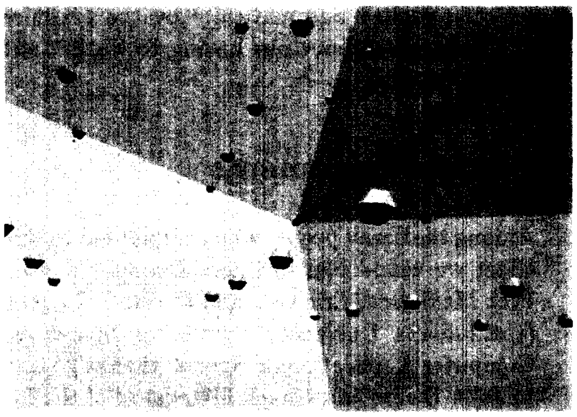


(Color Plate 47, page 467)

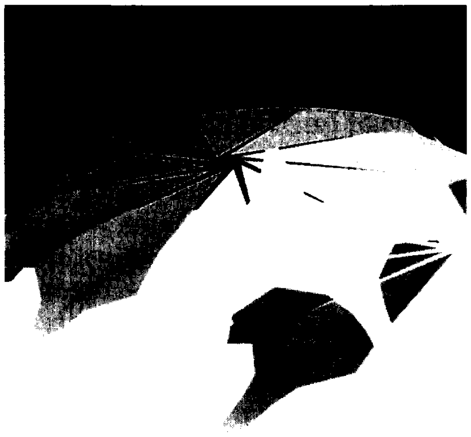


(Color Plate 48, page 467)

Figure 6: (a) One view of the surface obtained by projecting the Fermat equation for $n = 3$ from 4D to 3D. (b) The $n = 3$ surface obtained by projecting the entire infinite surface into a compact four-sphere, and then projecting that to 3D.



(Color Plate 49, page 468)



(Color Plate 50, page 468)

Figure 7: (a) Coarsely tessellated spheres, meant to represent integer lattice points, do not have the right effect if they are too close to the camera. (b) Rounding error and butterfly-twisted square patches cause gaps in the patchwork approach to the surface for large exponents n.