Rapid Raster Ellipsoid Shading

Don Herbison-Evans, donherbisonevans@yahoo.com

Technical Report 139 (1978), Basser Department of Computer Science, University of Sydney, Australia
(updated 14 October 2003)

ABSTRACT

The problems of coloring figures composed of 3-dimensional ellipsoids are discussed.

The problem of identifying the ellipsoid that colours a particular pixel is considered first. The use of osculating rectangular boxes (boxing) and spheres (bubbling) to reduce the number of quadratic equations that have to be solved are described, and their limitations evaluated.

In the calculation of the colour of a pixel, the use of the y component (vertical) rather than the z component (longitudinal) of the surface normal is advocated as giving fast pictures that look natural despite the absence of shadows. The use of moduli rather than squared components avoids slow square roots and gives pictures that are perfectly acceptable.

INTRODUCTION

In order to speed up the display of figures composed of ellipsoids on a shaded raster display, two problems need to be considered:

a. the identification, for each pixel, of the ellipsoid that colours that pixel;
b. the shade of colour that is assigned to the pixel, given the ellipsoid that colours it.

The calculations appropriate to to these problems can be viewed as occurring in a triply nested loop, with counters running through :

- x locations (where the x axis s oriented horizontally)
- y locations (where the y axis is orieted vertically)
- all the relevant ellipsoids.

In general, these loops may be nested in any order.

If frame and z buffers are available, it is natural to run through the ellipsoids in the outermost loop. Then, for a particular ellipsoid, the (x,y) locations are scanned, and where a point on the ellipsoid projects onto a pixel, the z depth of the ellipsoid surface at that point computed. If this z depth is less than the value in the z buffer at that pixel, the new value is written in, and the colour computed and put into the corresponding frame buffer pixel.

ELLIPSOID BOXING

It is wasteful to scan the whole (x,y) viewplane for every ellipsoid if each on average only covers a small proportion of the total picture area. In this case, it is useful to compute beforehand the minimum and maximum extents in x and y of the outline of each ellipsoid, and then only to scan over that subarea when painting the ellipsoid into the buffers (boxing).

If the semiaxis lengths of a particular ellipsoid are a_1, a_2, and a_3 (vector **a**), and the direction cosines of the major axes are represented by the matrix **R**, then the osculating rectangular box which just contains the ellipsoid has sides of length $2b_1$, $2b_2$, and $2b_3$, where :

$$b_i^2 = \text{sum}_j (r_{ij} * a_j^2)$$

If the centre of the ellipsoid is at vector position **c**, then :

$$((x_{min})_i , (x_{max})_i) = (c_i - b_i , c_i + b_i) \qquad i = 1 , 2 , 3$$

These six values are also useful if there is no frame or z buffer available. In this case, x and y must be scanned in the outer loops, and for each pixel: all the ellipsoids scanned as the innermost loop to see which has the lowest z depth there, and what colour is to be used for the pixel if it is the nearest so far.

If y is the outermost loop, then the y components of the box for each ellipsoid may be used to make an active list of ellipsoids for that scanline, which are the only ones that need be tested at the various x positions. Also buffers for frame and z are only required for the pixels in the line instead of for the whole picture.

The calculation of the z depth for a given (x,y) position and given ellipsoid involves the solution of a quadratic equation, which includes forming the discriminant (a biquadratic in x and y, typically requiring five multiplications) and, if this is positive, finding its square root.These operations may be slow compared with array accesss and value comparisons, depending on the hardware available. If so, time may be saved by prefixing by the tests :

$$x_{min} < x \qquad \text{and} \qquad x < x_{max}$$

This is worthwhile if

$$p*d > p*f + e*d$$

where

e = area of osculating rectangle around an ellipsoid
p = total picture area
f = average time to do the tests
d = time to solve the quadratic equation for z

This may be written as :

$$e/p + f/d < 1$$

i.e the sum of the fractional time speed up and the fractional area covered is less than one.

If the time to do the tests is significant, it can be minimised by doing the tests in such an order that most ellipsoids are rejected as early as possible for most pictures. This can be achieved approximately by finding and using the average centre x position of the active ellipsoids. If, for example, it is less than the centre of the scanline, it is likely that it will be faster to do the tests against x_{max} before those against x_{min}, as most of the pixels will be to the right of most of the ellipsoids.

It may also be useful to z boxing: if the z depth for some ellipsoid, A, that projects onto a given pixel is smaller than the z_{min} of another ellipsoid, B, then it is a waste of time computing the z depth of B at that pixel. This test can be made most effective by testing the ellipsoids in increasing order of z_{min}, i.e. painting from front to back: once one has been selected, the ones behind need not be examined. This z boxing and painting front to back are useful for speeding up calculations whether or not frame and z buffers are available.

ELLIPSOID BUBBLING

An alternative technique to boxing is to test each pixel against the osculating sphere for each ellipsoid. This sphere is centred on the centre of the ellipsoid, and has a radius, r, equal to the value of the largest semi-axis of the ellipsoid. Note that this test has no need for any square roots, by testing squared values :

$$r^2 > (x-c_1)^2 + (y-c_2)^2 + (z-c_3)^2$$

where x and y refer to the current pixel, and z is the smallest z depth found so far for that pixel.

This test may be slower than than the average time of the set of boxing tests as it involves three multiplications. Also it may not be so restrictive, and filter out fewer non-colouring ellipsoids. The volume of the osculating box can only be less than that of the osculating sphere if :

$$2b_1 * 2b_2 * 2b_3 < 4 * pi * r^3/3$$

or alternatively :

$$s_1 * s_2 / s_3^2 < pi/6$$

where **s** are the semi-axis lengths in ascending order :

$$s_1 < s_2 < s_3 \qquad (\text{ note that } s_3 = r \text{ })$$

COLORING

Given the ellipsoid that colours a given pixel, the next problem is to calculate the colour there. A common way to shade a curved surface in computer graphics is to use the value of the z component (away from the observer) of the surface normal. This method has the advantage that no shadow calculations are required. However, this leads to rather stark pictures, such as would be seen by a miner with a lamp on his hat.

A more natural shading is obtained if the y component of the surface normal is used. The absence of shadows with this nominally vertical illumination is seldom disturbing to the human eye.

shaded using z (away)
component of surface normal

shaded using y (vertical)
component of surface normal

If the ellipsoid is represented by the the quadratic form :

$$\mathbf{x}^{+}.\mathbf{M}.\mathbf{x} = 1$$

where

$$\mathbf{M} = \mathbf{R}^{+}.\mathbf{D}.\mathbf{R}$$

$$\mathbf{D} = \begin{pmatrix} 1/a_x^2 & 0 & 0 \\ 0 & 1/a_y^2 & 0 \\ 0 & 0 & 1/a_z^2 \end{pmatrix}$$

R is the matrix of semi-axis direction cosines,

and

+ superscript means transpose,

then the gradient is :

$$\mathbf{g} = 2.\mathbf{M}.\mathbf{x}$$

This may be normalised to give direction cosines :

$$\mathbf{n} = \mathbf{g} / \mathrm{sqrt}(\mathbf{g}^{+}.\mathbf{g})$$

The square root can be avoided by using an approximation to the gradient, **m**, computed using moduli :

$$\mathbf{m} = \mathbf{g} / (|g_1| + |g_2| + |g_3|)$$

This gives incorrect shading if the illumination is truly vertical and the surface is an isotropic scatterer. However, in normal life, these conditions are seldom realised, and shading using this algorithm for the y component of **m** is perfectly acceptable to the human eye.

shading using the modulus
rather than the square root

ACKNOWLEDGEMENTS

This work was made possible by a grant for study leave from the University of Sydney, and the hospitality and help of Norman Badler and the staff of the Computer and Information Science Department of the Moore School of Electrical Engineering, University of Pennsylvania, and facilities of the School of Computing Sciences of the University of Technology, Sydney.