NUDES 2: A NUMERIC UTILITY DISPLAYING
ELLIPSOID SOLIDS

Version 2.

D. Herbison-Evans


December 1977.

BASSER DEPARTMENT OF COMPUTER SCIENCE

THE UNIVERSITY OF SYDNEY

N.S.W. AUSTRALIA 2006
NUDES 2: A NUMERIC UTILITY DISPLAYING
ELLIPSOID SOLIDS

Version 2.

D. Herbison-Evans

December 1977.

BASSER DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF SYDNEY
NSW 2006
AUSTRALIA
NUDES 2: A Numeric Utility Displaying Ellipsoid Solids,

Version 2.

Abstract

D. Herbison-Evans

A system is described for producing 16 mm animated films of moving humanoid figurines and other figures composed of concatenated articulated interpenetrating ellipsoids. The method of Cohen is used for drawing the elliptical outlines of the ellipsoids. This uses a $2 \times 2$ matrix for each ellipse which plots it in terms of an angular parameter running from $0$ to $2\pi$.

A detailed description is given of an elegant hidden line algorithm for partially interpenetrating ellipsoids. This consists in solving the quartic equations resulting from the simultaneous solution of pairs ellipses, viz. (a) the outline of the one being drawn and (b) that of one potentially obscuring it. This is done in terms of Cohen's parameter for the drawn outline, resulting in an ordered list of hidden arcs of that outline. The visible outlines can then be generated to any required fidelity. The time for the entire process is dominated by the time required to solve, for $n$ ellipsoids, the $n(n-1)$ quartic equations, and is nearly independent of the number of straight chords generated and used to represent each outline.

Extensions to this system are described to allow interpenetrating ellipsoids to be drawn realistically and to allow for windowing to the viewing area. Where two ellipsoids interpenetrate, it is desirable to draw the outline of each not up to the outer edge of the other but to the points where it disappears into the other. These can be found by the simultaneous solution of the ellipse equations of (a) the outline being drawn and (b) the projection onto the viewing plane of the ellipse of intersection of the obscuring ellipsoid and the plane of the outline of the drawn ellipsoid. This doubles the number of quartic equations to be solved overall, but leads to much more realistic drawings.

The windowing to the viewing area is done, in the philosophy of the totally elliptical system, by defining a viewing ellipse. Parts of objects outside this ellipse need not be drawn. This then adds one more quartic equation to be solved for each ellipsoid in the scene.

A method of reducing significantly the number of quartics to be solved is described. Each pair of ellipsoids is filtered by testing for total inclusion and also for non-intersection of projected outlines by using the projected separation of centres versus their combined maximum and minimum semiaxis lengths.
NUDES 2: A Numeric Utility Displaying Ellipsoid Solids

Version 2.

D. Herbison-Evans

Introduction

This system is designed for producing 16 mm animated movies of figures composed of concatenated, articulated, interpenetrating ellipsoids. It was motivated by a request from a choreographer, the late Ms Phillipa Cullen, to show dances that had been previously coded in choreographic notation.

A number of earlier computer animation systems have included three dimensional ellipsoids as one of a number of basic shapes available (Goldstein 1971, Fleck et al 1975, Herbison-Evans 1974). These systems have either avoided the hidden line problem or else solved the full hidden surface problem. The complexity of this is somewhat forbidding (e.g. Levin, 1976).

In the present work, these calculations have been minimised by restricting the final visual representation to outline drawings rather than to shaded areas, by not drawing the lines of intersection of ellipsoids (Weiss, 1966), and by only allowing figures to be composed of ellipsoids (rather than in combination with polyhedra, cylinders, etc.)

The latter restriction limits application of the NUDES 2 system to figures which have natural convex 3D curvature. Thus it is very efficient for representing a human form e.g. using 20 ellipsoids, each being drawn using 100 chords, needs the solution of at most $2 \times 20^2$ (=800) quartic polynomials. A polyhedral representation giving this fidelity would need approximately $\frac{1}{2} \times 20 \times 100^2 = 10^5$ faces. An unsophisticated hidden line calculation on this polyhedral form would then need $\approx 10^{10}$ interface comparisons.

Conversely, the ellipsoidal model is inefficient for representing objects with flat faces. Thus it would take a row of 100 ellipsoids to represent the edge of a cube to the fidelity described above, and hence 60,000 ellipsoids would be needed to represent a single cube.

Thus it behoves the user to use a model to suit the figures he wishes to represent.
Input Language

The NUDES 2 input language is keyword driven, uses space as the main delimiter, and is generally free format. The syntax is described in Appendix II. It contains a description of the figures and their required motions.

The description of the figures is in the form of declaration, and gives names to the figures, the number of ellipsoids in each figure, and the names of each ellipsoid. Each ellipsoid is then defined and its three semi-axis lengths specified. Each articulation joint between pairs of ellipsoids is then named, and specified in terms of the ellipsoids which it joins, and the co-ordinates of the joint relative to the centres of each of these ellipsoids.

The motions required of the figures are described in terms of a number of primitives. Figures or the whole scene may be moved, scaled or rotated, or the joints of the figures may be bent. Each action can be specified as relative or absolute. It is specified over a set of notional frame numbers of the final movie. Each action can either be repeated for every frame in the set, or else distributed linearly over these frames, or else distributed quadratically. In the last case, it can be specified as accelerating and decelerating, i.e. starting and finishing at rest, or else accelerating from rest to some speed, or else decelerating from some speed to rest.

Additional commands allow the figures to be set so that their lowest point just touches the ground (i.e. the plane y = 0), also for the number of displayed frames to be increased or reduced by some integral factor, and also to allow a set of frames to be duplicated elsewhere.

Drawing the Elliptical Outlines.

The method Cohen (Cohen,1970) has been implemented to approximate the outline of an ellipsoid as a series of chords. The end of each chord is generated as the product of a matrix and the end of the previous chord,
relative to the ellipsoid centre.

\[
\begin{pmatrix}
u_{i+1} \\
v_{i+1}
\end{pmatrix} =
\begin{pmatrix}
m_{11} & m_{12} \\
m_{21} & m_{22}
\end{pmatrix}
\begin{pmatrix}
u_i \\
v_i
\end{pmatrix}
\]

\[
\begin{pmatrix}
x_i \\
y_i
\end{pmatrix} =
\begin{pmatrix}
u_i \\
v_i
\end{pmatrix} +
\begin{pmatrix}
c_x \\
c_y
\end{pmatrix}
\]

where

\((u_i, v_i)\) are co-ordinates of the \(i^{th}\) vertex relative to the ellipse centre,

\((x_i, y_i)\) are co-ordinates of the \(i^{th}\) vertex relative to the scene co-ordinate system,

\((c_x, c_y)\) are the co-ordinates of the centre of the ellipse relative to the scene system.

Thus the position of each chord vertex is obtained by 2 multiplications and 4 additions from the previous vertex. This process also has the advantage of automatically putting a higher density of points along those portions of the curve with higher curvature.

The elements of Cohen's matrix are related to the semi-minor and -major axis lengths of the ellipse, \(s_x, s_y\), and the angle, \(\phi\), between the minor axis of the ellipse and the scene x-axis:-

\[
m_{11} = (\sin\phi \cos\phi)(s_y/s_x - s_x/s_y) \sin\theta + \cos\theta
\]

\[
m_{12} = (\cos^2\phi \sin^2\phi + s_y/s_x - s_x/s_y) \sin\theta
\]

\[
m_{21} = -(\sin^2\phi \cos^2\phi - s_y/s_x - s_x/s_y) \sin\theta
\]

\[
m_{22} = (\sin\phi \cos\phi)(s_x/s_y - s_y/s_x) \sin\theta + \cos\theta
\]

where \(\theta = 2\pi/k\)

\(k = \) number of chords around ellipse.

If the ellipse is expressed in the form

\[a_1 x^2 + a_2 y^2 + a_3 xy + a_4 x + a_5 y + a_6 = 0\]

then

\[s_x = 2/(a_1 + a_2 + \sqrt{(a_1-a_2)^2 + a_3^2})\]

\[s_y = 2/(a_1 + a_2 - \sqrt{(a_1-a_2)^2 + a_3^2})\]

and

\[\tan\phi = 2(1/s_x^2 - a_1)/a_3\]
If the ellipsoid is expressed in co-ordinates with an origin at its own centre, but with co-ordinate axes parallel to the co-ordinate axes of the whole scene, as:

$$e_{11}x^2 + e_{22}y^2 + e_{33}z^2 + 2e_{12}xy + 2e_{13}xz + 2e_{23}yz = 1$$

and its centre is at \((c_x, c_y, c_z)\), then

$$a_1 = e_{11} - \frac{e_{13}^2}{e_{33}}$$

$$a_2 = e_{22} - \frac{e_{23}^2}{e_{33}}$$

$$a_3 = 2(e_{12} - \frac{e_{13} e_{23}}{e_{33}})$$

$$a_4 = -2a_1 c_x - a_3 c_y$$

$$a_5 = -2a_2 c_y - a_3 c_x$$

$$a_6 = a_1 c_x^2 + a_2 c_y^2 + a_3 c_z^2 - 1$$

Before drawing the outline of each ellipsoid, the intersection points of that outline are calculated for the ellipses corresponding to

(a) the viewing window ellipse

(b) the outlines of all the other ellipsoids in the picture

(c) the projections onto the xy plane of the ellipse of the intersection of all the other ellipsoids with the plane of the outline of the drawn ellipsoid.
Viewing Window

Windowing is done in order to eliminate unnecessary calculation of profiles, and to eliminate problems of wrap-around and screen over- or under-flow. On most display devices, the visible area is rectangular. The NUDES 2 system, in keeping with the elliptical philosophy of the system, has an elliptical viewing window.

If the display device is rectangular and strictly limited, then the elliptical window must be defined to lie within it, just touching it at the midpoint of each side. The loss is a small part of each corner, and is usually barely noticed.

If the display device has invisible display area around a rectangular visible area, then the viewing ellipse can be defined just to touch the visibility rectangle at its corners. There is then no loss of picture, but some unnecessary calculation of outlines. This is offset by the simplification of system from using ellipses throughout.
Intersections of Ellipses

Given the equations of two ellipses, then their points of intersection are the solutions to both of these equations simultaneously. If one of the equations is that of the outline of an ellipsoid which is to be drawn, then the intersection points are required in terms of Cohen's parameter, \( \theta \), for this outline. To obtain these, the equations must be expressed in a coordinate system centred on this ellipse and aligned parallel to its major and minor axes. Then the equations for \( x \) and \( y \) of a point on this ellipse in Cohen's formulation can be substituted into the equation of the ellipse which is intersecting it. This equation is in terms of \( \sin \theta \) and \( \cos \theta \), and these may be substituted by the appropriate formula in terms of \( t \) (i.e., \( \tan(\theta/2) \)) which gives the quartic:

\[
b_4 t^4 + b_3 t^3 + b_2 t^2 + b_1 t + b_0 = 0
\]

where

\[
b_4 = a_1 s_x^2 - a_4 s_x + a_6
\]

\[
b_3 = 2(a_5 s_y - a_3 s_x)
\]

\[
b_2 = 2(a_6 + 2a_2 s_y^2 - a_1 s_x^2)
\]

\[
b_1 = 2s_y(a_3 s_x + a_5)
\]

\[
b_0 = a_1 s_x^2 + a_4 s_x + a_6
\]

This quartic may be solved by the usual methods (e.g., Barnard & Child, 1936). The roots should lie in the range \(-\pi\) to \(\pi\). These are translated to lie in the range 0 to \(2\pi\), and put into a list in ascending order.
Interpenetrating Ellipsoids

The elliptical outline of the ellipsoid to be drawn lies in a plane. If this plane is expressed as

\[ p_x x + p_y y + p_z z = 0 \]

then

\[ p_x = e_{13}/e_{33} \]
\[ p_y = e_{23}/e_{33} \]
\[ p_z = -p_x c_x - p_y c_y - c_z \]

This plane will intersect some other ellipsoid, say with \( j \)th coefficients \( e_{j11} \) to \( e_{j33} \) and centre \( (c_{jx}, c_{jy}, c_{jz}) \), in another planar ellipse. If the projection of this ellipse is expressed in the form

\[ a_{j1} x^2 + a_{j2} y^2 + a_{j3} x y + a_{j4} x + a_{j5} y + a_{j6} = 0 \]

then

\[ a_{j1} = p_x^2 e_{j33} - 2 p_x e_{j13} + e_{j11} \]
\[ a_{j2} = p_y^2 e_{j33} - 2 p_y e_{j23} + e_{j22} \]
\[ a_{j3} = 2(p_x p_y e_{j33} - p_x e_{j23} - p_y e_{j13} + e_{j12}) \]
\[ a_{j4} = 2 p_x (c_{jx} e_{j13} - p_c e_{j33} + c_{jy} e_{j23} + c_{jz} e_{j33}) \]
\[ -2(p_c e_{j13} + c_{jx} e_{j11} + c_{jy} e_{j12} + c_{jz} e_{j13}) \]
\[ a_{j5} = 2 p_y (c_{jx} e_{j13} - p_c e_{j33} + c_{jy} e_{j23} + c_{jz} e_{j33}) \]
\[ -2(p_c e_{j23} + c_{jx} e_{j12} + c_{jy} e_{j22} + c_{jz} e_{j23}) \]
\[ a_{j6} = p_c^2 e_{j33} - 2 p_c (c_{jx} e_{j13} + c_{jy} e_{j23} + c_{jz} e_{j33}) \]
\[ + c_{jx}^2 e_{j11} + c_{jy}^2 e_{j22} + c_{jz}^2 e_{j33} + 2(c_{jx} c_{jy} e_{j12} + c_{jy} c_{jz} e_{j13} + c_{jz} c_{jx} e_{j13}) - 1.0 \]

The intersection points of this ellipse with the drawn outline are possible points of entry of the outline into the interior of the other ellipsoid. Such points are calculated in terms of Cohen's parameter for the drawn ellipse as before.
Shifting Ellipsoids.

In order to find the hidden arcs, one ellipse needs to be expressed in terms of co-ordinates centred on and aligned with the axes of another ellipse. This can be done in two stages:

(1) shifting the origin of an ellipse from \((0,0)\) to \((c_x, c_y)\):

\[
\begin{align*}
\alpha_6^1 &= \alpha_6 + \alpha_1 c_x^2 + \alpha_2 c_y^2 + \alpha_3 c_x c_y - \alpha_4 c_x - \alpha_5 c_y \\
\alpha_5^1 &= \alpha_5 - 2\alpha_2 c_y - \alpha_3 c_x \\
\alpha_4^1 &= \alpha_4 - 2\alpha_1 c_x - \alpha_3 c_y \\
\alpha_3^1 &= \alpha_3 \\
\alpha_2^1 &= \alpha_2 \\
\alpha_1^1 &= \alpha_1
\end{align*}
\]

(2) rotating the axes through an angle \(\phi\):

\[
\begin{align*}
\alpha_1^1 &= \alpha_1 \cos^2 \phi + \alpha_2 \sin^2 \phi - \alpha_2 \cos \phi \sin \phi \\
\alpha_2^1 &= \alpha_1 \sin^2 \phi + \alpha_2 \cos^2 \phi + \alpha_2 \cos \phi \sin \phi \\
\alpha_3^1 &= 2(\alpha_1 - \alpha_2) \sin \phi \cos \phi + \alpha_3 (\cos^2 \phi \sin^2 \phi) \\
\alpha_4^1 &= \alpha_4 \cos \phi - \alpha_5 \sin \phi \\
\alpha_5^1 &= \alpha_4 \sin \phi + \alpha_5 \cos \phi \\
\alpha_6^1 &= \alpha_6
\end{align*}
\]
Avoiding Intersection Calculations.

The solution of the quartic equation of the intersections of two ellipses is expensive, and avoided wherever possible.

Before finding the intersections of a possible obscuring ellipsoid with the outline of a drawn ellipsoid, a simple check is done to see if intersection is impossible. The projected separation of their centres is compared with the sum of the maximum semi-axis of the obscuring ellipsoid and the larger semi-axis of the drawn ellipse outline. If the latter sum is smaller, then they do not overlap.

If, after finding that they could overlap, the solution of the quartic gives no real roots, then the drawn outline is checked for total obscuration. If it is totally obscured, all further intersection and drawing calculations for that ellipsoid are skipped.

In checking for windowing, a preliminary test is made of the separation centres versus the difference in semi-axes to see if an elliptical outline is totally without or within the window.
Hidden Arcs

Having calculated the Cohen parameters of possible points where an arc of the ellipsoid being drawn may be obscured by the j'th other ellipsoid, the points 0 and 2π are added and then these are all sorted into ascending order. Each arc between two of these points is considered in turn, and at its midpoint, the z depth is compared of the outline and of the nearer point on the potentially obscuring ellipsoid. If the midpoint is obscured, then the whole arc is obscured, and it is melded into a list of hidden arcs of the outline.

The list of hidden arcs is kept as a stack of pairs of Cohen parameter values, each pair consisting of the start and end values of that hidden arc. The arcs are kept in ascending order. The melding of a new arc into this list is performed appropriately for the cases where the new arc has no overlaps with existing arcs, or overlaps a previous arc at either the lower or upper end, or else overlaps it completely.
Drawing Visible Arcs of Ellipses.

The system accepts, at run time, a specification of the number of chords to be drawn around each ellipsoidal outline, say \( k \). This is used to generate a normal Cohen matrix, using \( \theta = \frac{2\pi}{k} \), for each elliptical outline in turn of each frame in turn.

To draw a particular ellipse, the first point is taken on its semi-minor axis at

\[
\begin{pmatrix}
  x \\
  y
\end{pmatrix}
= \begin{pmatrix}
  c_x \\
  c_y
\end{pmatrix}
+ \begin{pmatrix}
  x \\
  y
\end{pmatrix}
\begin{pmatrix}
  s_x \cos \phi \\
  s_x \sin \phi
\end{pmatrix}
\]

The program then steps around the ellipse drawing chords until the next chord would include the beginning of the first invisible arc. The program then runs through the invisible arcs. The ellipse is drawn up to the start of the current invisible arc, and the notional pen moved invisibly to the end of that arc. The next invisible arc is then checked to see if that starts before the end of the current chord. If it does, that invisible arc is treated as before; otherwise the rest of the chord is drawn.

During the processing of a given hidden arc, the end of the current chord is updated to the next integral multiple of \( \frac{2\pi}{k} \) beyond the end of that arc. This saves unnecessary chord generation.
Current Implementation

The system is divided into three parts. The first program takes the NUDES 2 input commands for the figure definitions and then their motions, and from these produces a file of information on each frame of the resulting notional animated display. Each frame in this file has a list of the 3 dimensional sizes, positions, and rotation matrices of each of the ellipsoids.

The second program takes each frame in turn from this file, and calculates for each ellipsoid the projected position of its centre, the coefficients of Cohen's drawing matrix, and an ordered list of hidden arcs.

The third program takes this information, and draws the partial ellipses described there.

The first and second programs are implemented in ANSI standard Fortran on the Sydney University Computer Centre Cyber 72 and take about 3000 and 1000 Fortran statements (including documentation) respectively. The third program is implemented in Fortran on the Cyber to draw to the printer page (for rapid checking) and optionally to a Calcomp plotter. It is also being implemented on the 1MLAC PDS-4 display computer in PDS-4 assembler. This machine is connected to the Cyber by a 300 baud line, and also has a 16 mm stopped frame movie camera viewing a slave display screen under its control.
References


FLECK, J.T., BUTLER, F.E., and VOGEL, S.I., (1975), "An Improved Three Dimensional Computer Simulation of Vehicle Crash Victims".


APPENDIX 1.

Notation.

\[ x = (x, y, z) \]  
- co-ordinate system variables: \( x \): left to right,  
\( y \): down to up,  
\( z \): near to far.

\[ A = (a_1, a_2, a_3, a_4, a_5, a_6) \]  
- co-efficients of ellipse outline of ellipsoid,  
in form  
\[ a_1x^2 + a_2y^2 + a_3xy + a_4x + a_5y + a_6 = 0 \]

\[ B = (b_1, b_2, b_3, b_4, b_5) \]  
- co-efficients of quartic equation for tan \((\theta/2)\)

\[ C = (c_x, c_y, c_z) \]  
- Co-ordinates of centre of an ellipsoid.

\[ E = (e_{11}, \text{ to } e_{33}) \]  
- co-efficients of matrix of quadratic form of an ellipsoid (about its own centre, in form  
\[ e_{11}x^2 + e_{22}y^2 + e_{33}z^2 + 2e_{12}xy + 2e_{13}xz + 2e_{23}yz = 0 \]

\[ M = (m_{11}, m_{12}, m_{21}, m_{22}) \]  
- Cohen's matrix for obtaining the next point on the outline of an ellipse from a previous point.

\[ P = (p_x, p_y, p_c) \]  
- co-efficients of plane of outline of ellipsoid, in form  
\[ p_x x + p_y y + p_c x + z = 0 \]

\[ S = (s_x, s_y) \]  
- lengths of semi-axes in \( x \) and \( y \) directions respectively of an elliptical outline.

\( k \) = number of chords around ellipse outline.

\( \theta \) = Cohen's parameter for plotting an ellipse.

\( \phi \) = angle of semiminor axis of an ellipse to scene axis.

\( t \) = tan \((\theta/2)\) axis.
APPENDIX II    Syntax of NUDIES 2

< program > :: = < alldeclrs > < allmovnts >

< alldeclrs > :: = < alldeclrs > < declaration > | < declaration >
< declaration > :: = DEBUG < integer > |
                      SET < variablename > < value > |
                      FIGURE < figurename > < ellpsdcount > < ellpsdnamelist > |
                      ELLIPS < ellpsdname > < x > < y > < z > |
                      JOINT < jointname > < ellpsdname > < x > < y > < z > |
                      < ellpsdname > < x > < y > < z > |
                      COPY < figurename > < figurename >

< x > :: = < value > | < variablename >
< y > :: = < value > | < variablename >
< z > :: = < value > | < variablename >

<ellpsdcount > :: = < posinteger >
< value > :: = < integer > | <integer> |. < posinteger > | <posinteger>. <posinteger>
< integer > :: = <posinteger > | + <posinteger> | - < posinteger >
< posinteger > :: = < posinteger > < digit > | < digit >
< digit > :: = 0| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

< variablename > :: = < name >
< figurename > :: = < name >
< ellpsdnamelist > :: = <ellpsdnamelist> <ellpsdname> | <ellpsdname>
< ellpsdname > :: = < name >
< jointname > :: = < name >
< name > :: = "up to six characters"

< allmovnts > :: = < allmovnts > < movement > | < movement >
< movement > :: = REPEAT < framestop > < framestop > < action > |
                           LINEAR < framestart > < framestop > < action > |
                           QUADRA < framestart > < framestop > < action > |
                           ACCELE < framestart > < framestop > < action > |
                           DECELE < framestart > < framestop > < action > |
                           DUPLIC < framestart > < framestop > LIKE <prevstart> <prevstop>
                           SPEED < multiplier > |
                           STOP
<action> ::= GRNDAL |
              GRNDFG <filename> |
              GROALL <factor> |
              GROFG <filename> <factor> |
              RTALPT <angle> <axis> <x> <y> <z> |
              RTFSGPT <filename> <angle> <axis> <x> <y> <z> |
              RTALEL <ellipsdname> <angle> <axis> |
              RTFGEI <filename> <ellipsdname> <angle> <axis> |
              MVALBY <x> <y> <z> |
              MVFCBY <filename> <x> <y> <z> |
              MVALTO <ellipsdname> <x> <y> <z> |
              MVFCTO <filename> <ellipsdname> <x> <y> <z> |
              BENDBY <jointname> <movingellname> <angle> <axis> <refellname> |
              BENDTO <jointname> <movingellname> <refellname> <colatitude> <longitude> <twist> |
              DEBUG <level> |

<factor> ::= <value> | <variblename> |
<angle> ::= <value> | <variblename> |
<colatitude> ::= <angle> |
<longitude> ::= <angle> |
<twist> ::= <angle> |

</prevstart> ::= <value> | <variblename> |
</prevstop> ::= <value> | <variblename> |
</framestart> ::= <integer> |
</frametop> ::= <integer> |

<factor> ::= <value> | <variblename> |
<level> ::= <integer> |
<multiplier> ::= <integer> |

<movingellname> ::= <ellipsdname> |
<refellname> ::= <ellipsdname> |