Drawing Perspective Surfaces with Hidden Line Removal

*In honor of my teacher Dr. Olgierd Alf Biberstein **

Ignacio Vega-Páez[1], José Angel Ortega[2], Georgina G. Pulido[3]
ivega@g-ibp.com, oeha430210@hotmail.com and gpulido@att.net.mx

*2025 translation to English by Stacy David Thurston[4] ***

IBP-TR2008-08
July 2008, México, D.F

Abstract

An efficient computer algorithm is described for the perspective drawing of a wide class of surfaces. The class includes surfaces corresponding to single valued, continuous functions which are defined over rectangular domains. The algorithm automatically computes and eliminates *hidden lines*. The number of computations in the algorithm grows linearly with the number of sample points on the surface to be drawn. An analysis of the algorithm is presented, and extensions to certain multivalued functions are indicated.

The algorithm is implemented and tested on .Net 2.0 platform. Run times were found to be exceedingly efficient for visualization, where interaction online and viewpoint control, enables effective and rapid examination of a surfaces from many perspectives.

[1] Postgraduate Studies and Research in Mechanical and Electrical Engineering from the National Polytechnic Institute, Instituto Politécnico Nacional (SEPI ESIME IPN) and Information Technology Management from the International Business Partner Consulting Company (IBP -Consulting)

[2] Graduate Studies and Research in Mechanical and Electrical Engineering of the National Polytechnic Institute, Instituto Politécnico Nacional

[3] Department of Basic Sciences and Engineering of the Metropolitan Autonomous University, Universidad Autónoma Metropolitana (UAM).

[4] Was a computer advisor at the University of Manitoba, B.A. with minor in math

* Professor of Physics and Mathematics at the National Polytechnic Institute, Instituto Politécnico Nacional, Mexico City, from1970 until death in 1997.

** Original arXiv publication in Spanish: https://arxiv.org/abs/0809.4093, original license: Creative Commons(CC) 3.0, this translation is CC version 4.0.

Terms Key: Hidden line algorithm, Drawing surfaces, Computational Geometry, CAD/CAM, Computer Vision, Graphics.

Introduction

The general problem of efficiently and meaningfully displaying three-dimensional objects in two dimensions is central to computer graphics. In particular, when drawing perspective objects the *line hiding problem* of removing line segments not visible from the observer's point of view is one of the problems that has always been present. Over time, several algorithms dealing with this problem have appeared in publications. Relative to the number of points, some of these algorithms are prohibitively time-consuming or have very large memory requirements. Some have both of these undesirable characteristics.

This paper gives a detailed description of a very efficient algorithm for drawing in perspective an arbitrary surface corresponding to a continuous single-valued function defined on a rectangular domain, with hidden lines removed. The observation point from which the surface is viewed can be any point outside the surface. Some generalizations of the algorithm to non-rectangular domains and to multi-valued functions are mentioned.

This algorithm has been implemented on the .NET 2.0 platform with considerably shorter computation time than previous algorithms for drawing similar surfaces. A very useful feature of the application is that the program works interactively and allows the user to change the observation points with the *mouse* and immediately display the view of the surface under analysis.

II. Problem Formulation

Given a three-dimensional surface *S* defined on a rectangular domain *R*, and an observation point *V* outside *S* (figure 1), we may assume that *R* is centered at the origin *O* of the *x*-*y* plane and oriented so that its sides are parallel to the *x* axis and the *y* axis, respectively. This can be accomplished without loss of generality by convenient translations and rotations of both *R* and *V*.

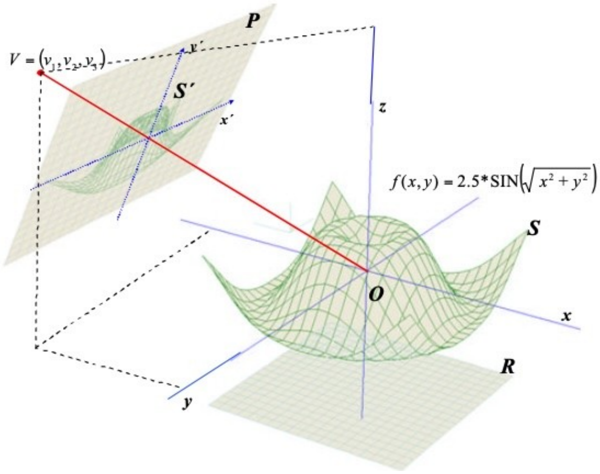


Figure 1: Perspective projection of surface S on plane P.

The plane *P*, containing the perspective image *S′* of *S*, is chosen to be perpendicular to the line joining *V* and the origin *O*. A rectangular coordinate system with *x′*-*y′* axes at *P* is chosen which preserves the original vertical direction of *S*. That is, the *y′* axis on *P* must be the projection of the original *z* axis. This last choice is necessary, as we shall see later.

Relative to the rectangular system on P, the perspective image on P of each point in S has a well-defined pair of coordinates. The proof of these coordinates is given in the original Spanish edition appendix(see page 1 for a link).

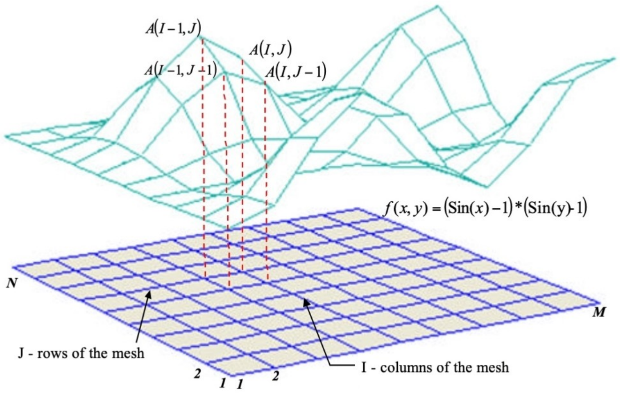


Figure 2: Surface patch of a subrectangle of the mesh.

The surface S is first quantized into an MxN rectangular mesh of points over the domain R. figure 2 shows the mesh lines in R, dividing R into subrectangles. The part of S defined over a subrectangle is shown. It will be called the *surface patch* of S. After quantization, only the four points of the patch defined over the four vertices of the subrectangle are known, and linear interpolation of the function between adjacent mesh points is assumed, that is, between adjacent points in the same row and adjacent points in the same column of the mesh. Thus, S need not be explicitly defined by a mathematical function; the data for S can be given by a rectangular array of points corresponding to a rectangular mesh of points over the domain. The behavior of S inside each subrectangle of the mesh is ignored. Thus, each surface patch of S will be represented by its four linearly interpolated edges in three-dimensional space. The image of these edges in the projection plane is a four-sided polygon (figure 3). For each patch of the surface S, only the visible line segments of these edges will be drawn.

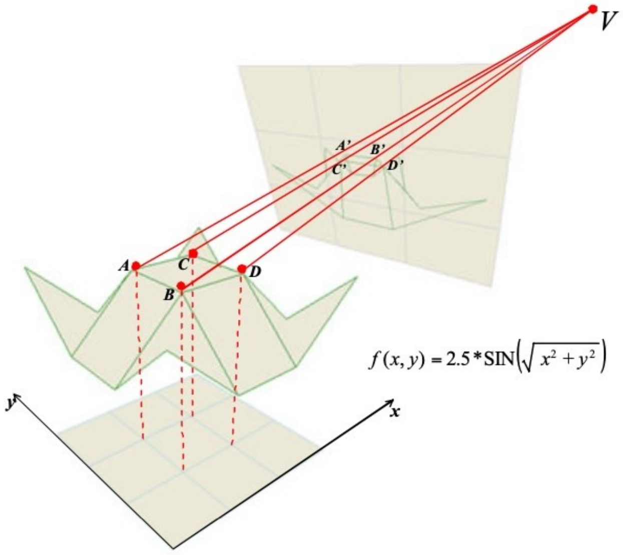


Figure 3: Projection of the linearly interpolated surface patch.

Intuitively, the surface should be thought of as an opaque elastic membrane stretched over a rigid frame consisting of all linearly interpolated edges of the surface patches. The problem then is to draw the surface in perspective by removing the line segments hidden by the opaque membrane from the observer's point of view.

Note, the vertical projection onto the *x-y* plane coincides with the mesh lines in *R*.

III. Basic Ideas of the Algorithm

The algorithm for determining the visibility of any given edge or segment is based on two ideas. The first consists in choosing a particular *ordering* of the patches on the surface *S* so that no patch appears earlier in the *order* and is hidden from the observation point or the patch appears later. The surface must be drawn one patch at a time according to the chosen ordering. Thus, a line segment on some edge of a patch under consideration is hidden from the observation point if, and only if, its image on the projection plane lies within a region of the plane already covered by the images of previously drawn patches.

This particular arrangement of the surface patches has the property that, for a surface *S* which corresponds to a single-valued continuous function on a rectangular domain, the region of the projection plane which is covered by the image emerging from *S,* will grow continuously with each patch that is drawn in the chosen order. And because the vertical direction of the surface is preserved, at each stage of the drawing of *S*, the *hidden* region of the plane can be precisely delimited between two continuous piecewise linear functions. Thus, determining the visibility of any edge of a surface patch is reduced to the problem of deciding how much of its image on the projection plane lies between these two functions. This delineation of the hidden region of the projection plane, after each patch is drawn, by means of the two continuous piecewise linear functions, constitutes the second idea of the algorithm. As can be seen in figure 4.

Comment: if the *y* axis on the projection plane is not chosen to be the perspective image of the original *z* axis, then the surface image will be *tilted* such that it will be impossible to delineate its boundary using only two functions.

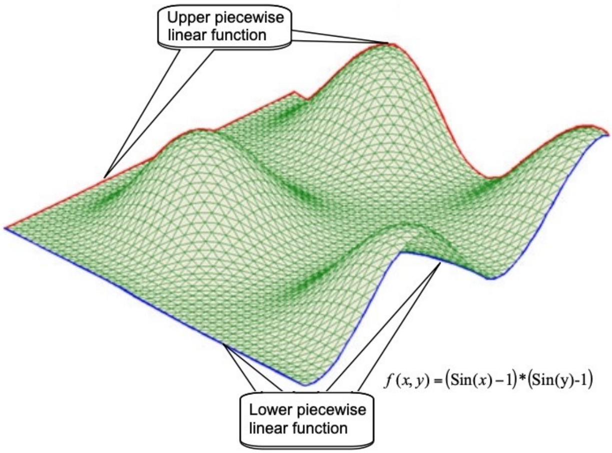

Upper piecewise linear function
$f(x, y) = (\mathrm{Sin}(x) - 1) * (\mathrm{Sin}(y) - 1)$
Lower piecewise linear function

Figure 4: Hidden regions of the plane are limited between two continuous piecewise linear functions.

IV. Ordering of the Surface Patches

It is possible to arrange the patches of the surface S such that the closer ones are not hidden by the more distant ones, depends on the following observation:

Suppose A and B are two arbitrary points on the surface such that A hides B from V. Geometrically, this means that A, B, and V lie on a straight line, and the distance from A to V is less than the distance from B to V. Then, it is easily shown that the vertical projections A_0, B_0, V_0 of the points A, B, V respectively, on the x-y plane, satisfy the relation as well. That is, they are collinear, and A_0 is closer than B_0 to V_0.

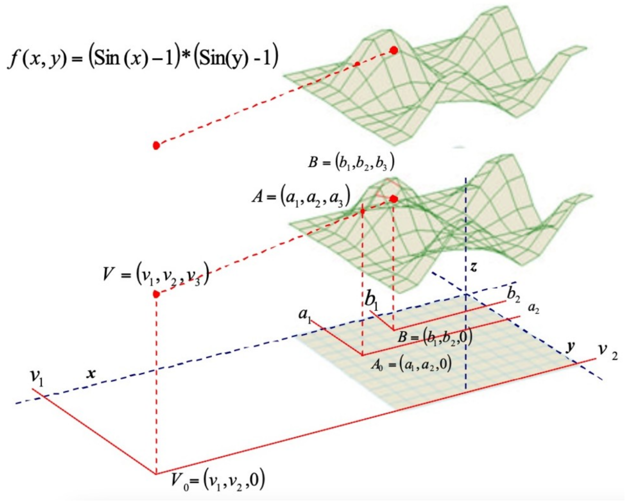


Figure 5: Points A and B on the surface with A hiding B.

This observation is the key to the ordering of the surface patches. First, consider the case where the vertical projection $V_0=(v_1,v_2,0)$ of the observation point onto the *x-y* plane is southwest of the domain, as shown in figure 5. [That is, $v_1<=x_1$, and $v_2<=x_2$, where $(x_1,y_2.0)$ is the lower left corner of the domain.] An immediate consequence of the above observation is that if $A=(a1,a2,a3)$ hides $B=(b_1,b_2,b_3)$ from $V=(v_1,v_2,v_3)$, then $a_1<b_1$, and $a_2<b_2$; i.e., A_0 must be further southwest than B_0 (a consequence of the southwest location of V_0). This suggests the following rule for determining the relative order of points in S, given their location southwest of V_0: Let *A* and *B* be any two points in *S*. If one of them, say *A*, is further southwest than the other (more precisely, if A_0 is further southwest than B_0), then *A* must precede *B*. On the other hand, *A* and *B* can be ordered independently of each other.

There are several ways of arranging the patches of the surface *S* so that this rule of relative point order is satisfied, so that early patches are not obscured by later patches—see Figures 6 and 7 which illustrate possible arrangements of the subrectangles of *R*. The induced ordering of the patches of *S* has this property; i.e., arranged row by row in figure 6, from left to right in each row, starting with the front row (relative to the southwest location of V_0). It is obvious that, under this arrangement of patches, no point in a closer patch will be obscured by a point in a later patch, since no point in a closer patch is further southwest than any point in a later patch.

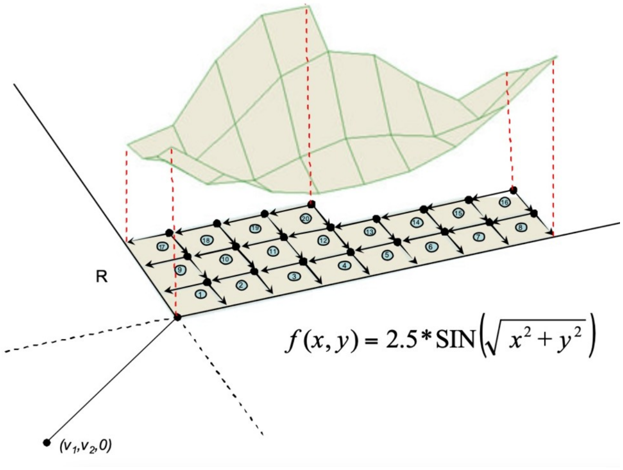


Figure 6: Frontal arrangement of the mesh patches.

For other *corner* locations of V_0, (i.e., northwest, northwest, and southeast of the domain) analogous classifications of the surface patches can be made. Rather than discuss convenient classifications of the surface patches for other locations of V_0, we proceed to describe the algorithm in detail for the special case where V_0, is southwest of the domain, and then show how the other cases can be reduced to cases involving only *corner* locations of V_0. Another ordering which preserves the property is shown in figure 7. The Cantor ordering which scans the domain *R* diagonally, starting with the corner nearest to V_0, that is, (1,1), (1,*N*), (*N*,1), or (*N*, *N*), guaranteeing that the nearest ones will be painted first and the later ones, later.

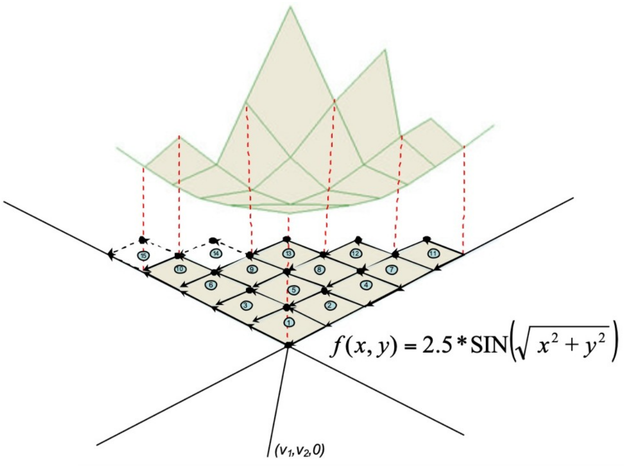


Figure 7: Frontal arrangement of the mesh patches.

V. Algorithm Details

Let *S* be a surface defined on a rectangular domain *R* of the *x*-*y* plane. We assume that *S* will be viewed in perspective from an observation point *V* whose vertical projection V_0 on the *x*-*y* plane is southwest of *R*. Given a rectangular *M*x*N* mesh of points on *R*, for *I*=1,2,...,*M* and *J*=1,2,...,*N*, let *A*(*I, J*) denote a *sample point* or simply point on *S* defined on the mesh *R* located at the *I*-th column and *J*-th row of the mesh. For *I*=2,...,*M* and *J*=2,...,*N*, let *S*(*I*, *J*) be the patch of the surface with vertices *A*(*I*, *J*), *A*(*I*,*J*-1), *A*(*I*-1,*J*), *A*(*I*-1,*J*-1). See Figure 2. Above considerations, the patches can be ordered as follows, and drawn according to this ordering, one by one:

S(2,2), *S*(3,2), ..., *S*(*M*,2),

S(2,3), *S*(3,3), ..., *S*(*M*,3),

.....

S(2,*N*), *S*(3,*N*), ..., *S*(*M*, *N*),

...

Now, consider the sequence of points *A*(*1*,*J*), *J*=1,...,*N*, defined along the first column of points of the mesh. The line segment joining these points in consecutive order will be called the *main left edge* of *S*. Similarly, the line segment joining the sequence of points *A*(*I*,1), *I*=1,...,*M* in consecutive order will be called the *main front or right edge* of *S*. As a consequence of the observation made at the beginning of Section IV, both edges are completely visible from *V*. Thus, we can start drawing the entire line segment of the main edges. Then, the drawing of each patch *S*(*I, J*), following the above order, can be done by adding only the visible segments of two of its four edges, namely its *back edge*, joining *A*(*I, J*) and *A*(*I*-1,*J*), and its *right edge*, joining *A*(*I, J*) and *A*(*I, J*-1), because the other two edges must have already been considered in relation to the previous patches or as one of the main edges.

Figure 8a has an example surface defined on a 5x3 point mesh. The 5x3 arrangement of dots shown on the surface results in two rows of patches: four in the front row and four in the back row. Figure 8b shows the conjunction of the main left edge and the main front edge of the surface.

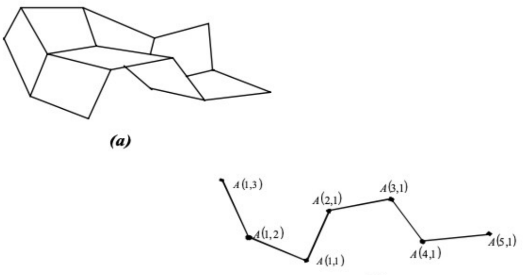


Figure 8: (a) Example surface. (b) Main left and front edges (b).

Figures 9a through 9h show the subsequent patches that are added as the surface is drawn. As each patch is added, L_1 denotes its *back* edge, and L_2 denotes its *right* edge. The dotted lines indicate hidden partial line segments that are not painted.

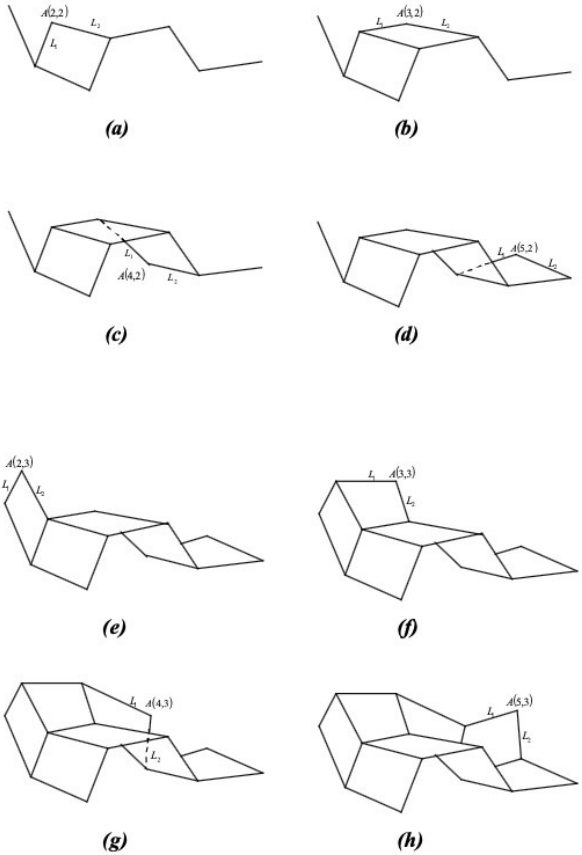


Figure 9: Sequence in which the patches of surface S are not painted.

As indicated earlier, determining the visibility of each new edge under consideration will involve two continuous piecewise linear functions. Intuitively, one of these functions, Max, will be used to define the upper perimeter curve of the region of the projection plane so far covered. The other function, Min, will be used to define the lower perimeter broken curve. At this point, it would be useful to identify the tracing device mesh with a rectangle in the projection plane that circumscribes the perspective image of the surface. This corresponds to framing and transforming the coordinates of the image points so that they fall within the dimensions of the tracing device mesh.

So to improve accuracy in delineating the hidden region of the projection plane, or equivalently, the hidden region of the tracing mesh, the number of points in each of the Max and Min functions is chosen to match the number of horizontal units in the tracing device mesh.

Before starting the layout, we artificially set

$\text{Max}(k) = \text{Min}(k) = -1$, for $k=1,\dots,Kx$

where Kx is the horizontal dimension of the layout mesh.

After each line segment is drawn, Max and/or Min will be modified to reflect the change in the shape of the image arising from S.

We start with the line segments on the leading edges of S.

Let A and B be any two consecutive *sample points* on the leading left edge or the leading front edge, that is, any two consecutive points in the following succession (see figure 2):

$A(1,N), A(1,N\text{-}1), \dots, A(1,1), A(2,1),\dots, A(M,1)$

Let Ax, Ay be the plot coordinates of A (scaled and translated), and Bx By be the plot coordinates of B. First, the line segment joining the points (Ax, Ay) and (Bx, By) is painted. Then Max and Min are redefined between Ax and Bx by setting

$\text{Max}(Ax)=\text{Min}(Ax)=Ay$

$\text{Max}(Bx)=\text{Min}(Bx)=By$

Linearly interpolating Max and Min between Ax and Bx. In figure 10, Kx and Ky refer to the dimensions of the layout mesh, in the horizontal and vertical directions, respectively.

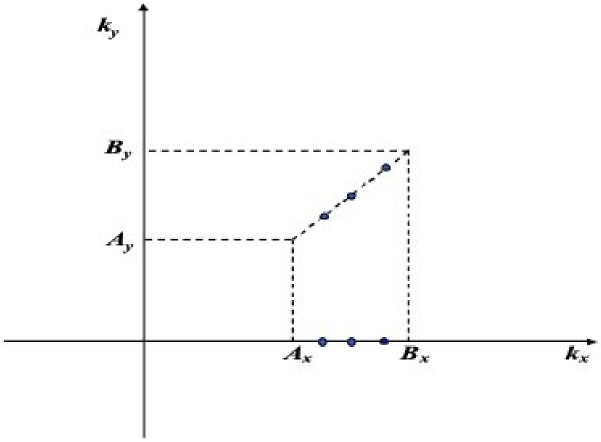


Figure 10: Linear interpolation of Max and Min.

After the above procedure is carried out for each pair of consecutive points in the array, the principal edges will be drawn completely, and their dashed line will be defined by Max and Min between the two end points. Note that the bounded region between Max and Min has zero area and reflects the fact that no patch has yet been drawn.

We now proceed to draw the surface patches. Each *S*(*I, J*) in the array, starting with *S*(2,2) will be processed uniformly as follows: Let *L1* denote the edge between *A*(*I, J*) and *A*(*I*-1,*J*). Let *L2* denote the edge between *A*(*I, J*) and *A*(I, *J*-1). For each of these edges, we must first determine its visible segments, then draw only those segments, and modify Max or Min to reflect the change in the shape of the hidden region (the entire edge may be hidden, of course).

First, Let *L*=*L1*, and *A*=*A*(*I, J*), *B*=*A*(*I*-1,*J*), be the endpoints of *L1*. The following criterion is used to determine the visibility of any point *C* on *L*: If *Cx*, *Cy* are the stroke coordinates of *C*, then

C is visible if and only if *Cy*=Max(*Cx*) or *Cy*=Min(*Cx*). This visibility criterion applies to each of the endpoints *A* and *B*. (Let *Ax*, *Ay* be the stroke coordinates of *A* and *Bx*, *By* those of *B*.) There are four possibilities:

A, B are both visible. We then assume that all points in *L* are visible. This assumption is made to reduce computation time. It may happen that some hidden line segments are drawn incorrectly if the surface has sharp *spikes* in the foreground. This problem can be overcome by starting with a finer mesh on *R*.

The line segment joining (*Ax*, *Ay*) and (*Bx*, *By*) is drawn. If *Ay*=Max(*Ax*), Max is linearly interpolated between *Ax* and *Bx* by setting Max(*Ax*,)=*Ay*, Max(*Bx*,)=*By*. Otherwise, we must have *Ay*=Min(*Ax*), and Min is similarly interpolated between *Ax* and *By*, with Min(*Ax*)=*Ay*, Min(*Bx*)=*By*.

(ii) *A* and *B* are both hidden. Then, for the same reasons as in case (i), we assume that all points in *L* are hidden and no line segment is drawn. Max and Min are left unchanged and we proceed to the next edge.

(iii) *A* is hidden and *B* is visible. A search is made along the line joining *A* and *B*, starting at *A*, for the first visible point *C* (discrete steps are taken in the horizontal direction of the tracing mesh). The segment joining *C* and *B* is assumed to be completely visible and is drawn. If *Cy*=Max(*Cx*), Max is shifted linearly between *Cx* and Bx, with Max(*Cx*)=*Cy*, Max(*Bx*)=*By*, otherwise Min is shifted between *Cx* and *Bx*.

(iv) *A* is visible and *B* is hidden. A search is made along the line joining *A* and *B* for the first visible point *C*, starting from the hidden point of *B*. The rest is analogous to case (iii).

For each patch of the surface *S*(*I, J*), this procedure is carried out with *L*=*L1* and *B*=*A*(*I*-1,*J*); then the procedure is repeated with *L*=*L2* and *B*=*A*(*I, J*-1). This completes the description of the algorithm for the special case where the vertical projection of the observation point V_0 onto the *x*-*y* plane is in the southwest of the domain *R*.

If V_0 is in any of the other *corner* regions of the *x*-*y* plane (indicated by *NW*, *NE*, and *SE* in figure 11a), we can rotate the data defining the surface (or the mathematical function or the rectangular array of sample points) by 90, 180, or 270 degrees, respectively, and apply the algorithm as described for V_0 to the *SW* region of the *x*-*y* plane. Another approach is to change the ordering of the surface patches and define new principal edges to satisfy the other locations of V_0.

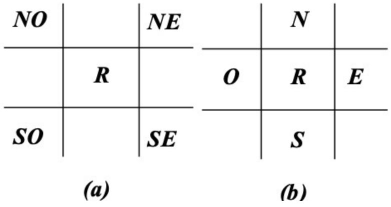


Figure 11: Domain partitioning for non-cornered V_0 locations.

If V_0 lies in any of the regions directly west, north, east, or south of *R* (figure 11b), we can partition *R* into two sub-rectangles *R1* and *R2* (figure 12b), and apply the algorithm to each of *R1*, *R2*, since V_0 is a *corner* location of each sub-rectangle. The two parts of the surface corresponding to *R1* and *R2* will have to be *glued together* along the partition line, and this can be done without *failure* by augmenting the sample of surface points with the set of points defined at the intersection of the partition line with the mesh lines on *R*.

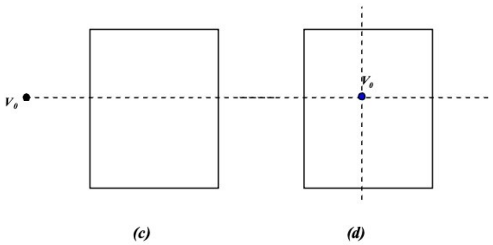


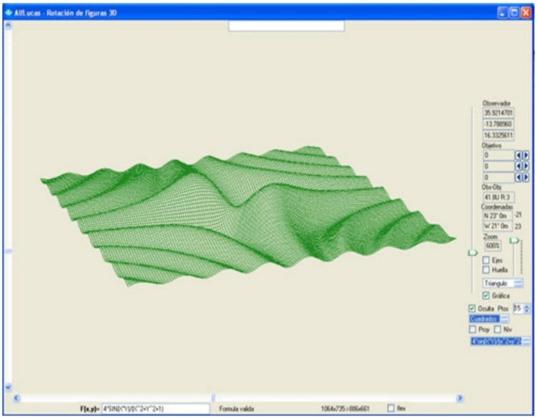
Figure 12: Possible locations of the observation point.

Finally, if V_0 is within the domain R, we can divide R into four sub-rectangles (figure 11d) and apply the algorithm to each of $R1$, $R2$, $R3$, $R4$, with V_0 located at the *corner.*

Comment: Note that under the assumption that the projection plane P must be perpendicular to the line joining V and the origin, if V_0 coincides with the origin, then the projection of the z-axis of V onto P is a point. In this case, the y'-axis at P must be chosen to be the image of some other line. See the Appendix.

Figures 1 through 7 were generated by the implementation on .NET Framework-compatible personal computers. And the surfaces in Figures 13 and 14 also show the working environment of the implementation, for these cases they were evaluated with a square mesh of 150 and 90 points, for a total of 22,500 and 180 points.

It is clear that using this algorithm, the computation time must vary linearly with the number of points, this is because each point is processed by evaluating its stroke coordinates IX, IY, and then the ordinate IY is compared with two values, Max(IX) and Min(IX).


AlfLucas - Rotación de figuras 3D

Observador: 35.9214701, -13.788960, 16.3325611
Objetivo: 0, 0, 0
Obs-Obj: 41.8U R:3
Coordenadas: N 23° 0m -21; W 21° 0m 23
Zoom: 600%
☐ Ejes
☐ Huella
Triangulo
☑ Gráfica
☑ Oculta Ptos 15
Cuadrados
☐ Proy ☐ Niv

F(x,y)= 4*SIN(X*Y)/(X^2+Y^2+1)
Formula valida
1064x735->886x661
☐ flex

Figure 13: Implementation work environment.

VI. Generalizations

6.1 Convex domains

The algorithm can be generalized to an arbitrary surface *S* defined by a continuous single-valued function over a convex domain *R*, by extending the definition of the function *f* artificially to a rectangular domain *R′* containing *R*. The ideas of the algorithm are then applied to the surface *S′* defined by the extended function *f* over *R′*, with the two following modifications.

First, the sample of points along the major edges of *S′* must be replaced by a subset of points defined at the intersection of the boundary of *R* with the mesh lines on *R′*. And, a special consideration must be given to drawing the lines connecting these points.

Second, a test must be made for the sample of points in *S′* to determine whether they belong to *S*, so that only line segments that are part of *S* are pinned. One way to accomplish this is to set $f(x, y)=z_0$ for all (x, y) in $R'-R$, where z_0 is a number outside the range of values of *f* over *R*. The convexity of the domain is necessary to bound the image that emerges at each stage of drawing between two functions.

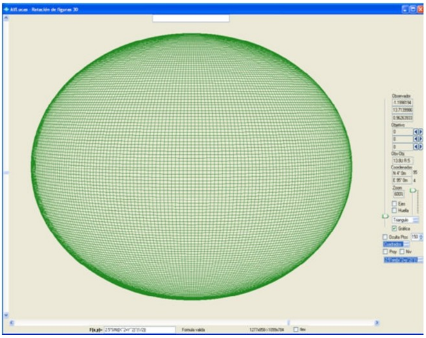


Figure 14: The sphere in the implementation environment with convex domain.

6.2 Multi-valued functions

The algorithm can also be extended to certain multi-valued functions. For example, if the surface is a sphere, it is first divided by a horizontal plane into two single-valued pieces, and if the observation point is before the horizontal plane, the algorithm is first applied to the upper hemisphere, and then, without resetting Max or Min, the algorithm is applied to the lower hemisphere.

References

[Beebe 79] Nelson H.F. Beebe, "A Users's Guide to <PLOT 79>", Departments of Physics and Chemistry, University of Utah, Salt Lake City, Utha, USA., 79.

[Bibertein 90]. Olgierd Alf Biberstein. Notas del curso Geometría Diferencial I y II. Escuela Superior de Física y Matemáticas del IPN, México, D.F. 1990-1991.

[Canto 93] Ángel Bernardo Canto Gómez, "Descripción de Funciones <PLOT 90>", Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla, Mex., Ago 93."

[Canto 94] Ángel Bernardo Canto Gómez y Harold V. McIntosh, "Paquete de Graficado <PLOT 90C> para PC Versión 1.0", Departamento de Aplicación de IBP-Memo 2008-08 20/21 Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla, Mex., Mar 94."

[Foley 81] J.D. Foley and A. VanDam, "Fundamentals of Interactive Computers Graphics," Addison-Wesley, Reading, Mass., 1981.

[Gil 84] Francisco Gil Z. y Harold V. McIntosh, "<PLOT 84>", Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla, Mex., Ago 86."

[Hernández 93] José Pedro Hernández Enríquez, "Descripción breve de las subrutinas degraficación de PLOT 84", Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla, Mex., Ago 93."

[McIntosh 75] Harold V. McIntosh, "<PLOT 75>", Instituto Nacional de Ingeniería Nuclear, Salazar, Edo. de Mex., México., May 75.

[McIntosh 89] Harold V. McIntosh, "Contours for <PLOT>", Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla, Mex., Mar 89.

[McIntosh 93a] Harold V. McIntosh, "GEOM, A program to Draw Molecules (and Eclipses)", Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla, Mex., May 93.

[McIntosh 93b] Harold V. McIntosh, "GEOM, for drawing SPHERES and things", Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla, Mex., Nov 93.

[Mortenson 88] M.E. Mortenson, "Computer Graphics" An Intruduction to the Mathematics and Geometry., Industrial Press, New Cork, 1988.

[Newman 76]. Newman W. and Sproull R. Principles of interactive computer graphics. Edit. McGraw-Hill New york, 1976.

[Ortega 88] José. A. Ortega, R. Magdaleno R., "Análisis y Diseño en Ingeniería Auxiliado por Computadora", I.P.N., Serie en Ciencias e Ingeniería, Vol. 10, 1988.

[Tamayo 92]. Tamayo Islas C. Notas del curso graficación e interfaces graficas. Depto de Ingeniería Eléctrica CINVESTAV, México, D.F. DIC 1992.

[Vega-Páez I. 95] Vega-Páez I y C. Hernández-Hernández, "PLOT 94 in ambiance X-Windows", Proceedings in Information Systems Analysis and Synthesis ISAS '95, 5th International Symposium on Systems Research, Informatics and Cybernetics, pp. 135-139 August 16-20, 95, Baden-Baden, Germany.