Chen Tet Khuan and Alias Abdul Rahman / Geoinformation Science Journal, Vol. 6, No. 1, pp. 84-92

0D Feature In 3D Planar Polygon Testing For 3D Spatial Analysis

Chen Tet Khuan and Alias Abdul-Rahman

Department of Geoinformatics,
Faculty of Geoinformation Science and Engineering,
Universiti Teknologi Malaysia,
81310 UTM Skudai, Malaysia

{kenchen, alias}@fksg.utm.my

Abstract

In this paper, we discuss the problem of 0D feature-in-3D planar polygon. It has been recognized that 3D union and 3D intersections are among important spatial operators in 3D GIS. A 0D feature-in-polygon test is one of the problems in 3D GIS. Though it is a classic problem, however, it has not been addressed appropriately in the literature. Moreover, the 0D feature-in-polygon query is rather complicated if it is implemented into 3D spatial information system. From the aspect of 3D spatial analysis, the general 0D feature-in-3D planar polygon problem should be formulated in a suitable way. Our basic idea is to solve a general 0D feature-in-3D planar polygon problem that includes all special conditions. The method addresses an essential mathematical algorithm that applicable for real objects and provides an approach for implementation in further 3D analytical operation, e.g. 3D union or 3D intersection for 3D GIS.

Key words: 0D feature, 3D planar polygon, and 3D GIS

1. Introduction

In GIS, a very natural problem that implements the field of computational geometry is the determination of 0D feature in polygon. By given a 0D feature, N and an arbitrary closed polygon P represented as an array of n points, $P_0, P_1, \ldots, P_{n-1}, P_n = P_0$, determine whether N is inside or outside the polygon P. In literature (Foley, et. al, (1990); Haines, (1994b); Harrington, (1983); Nievergelt and Hinrichs, (1993); O'Rourke, (1998); Sedgewick, (1998); Weiler, (1994); Woo et. al, (1997)), two categories of definitions can be found. The first one is the even-odd, in which a line is drawn from N to some other point S that is guaranteed to lie outside the polygon. If this line NS crosses the edges $e_i = P_i P_{i+1}$ of the polygon an odd number of times, the N is inside P, otherwise it is outside (see Figure 1a). This rule can easily be turned into an algorithm that loops over the edges of P, decides for each edge whether it crosses the line or not, and counts the crossings. We discuss these issues in detail in Section 4. The second one is based on the *winding number* of N with respect to P, which is the number of revolutions made around that point while traveling once along P. By definition, N will be inside the polygon, if the winding number is nonzero, as shown in Figure 1(b).

From these two methods, various implementations of these strategies exist like (Franklin, (2005); Haines, (1994a); Haines, (1994b); Mehlhorn and Näher, (1999); O'Rourke, (1998); Sedgewick, (1998); Stein, (1997); Theoharis and Böhm, (1999)), which differ in the way to compute the intersection between the line and an edge in order to determine whether the 0D feature is inside or outside the polygon. However, some of the

problems need to be investigated, since these determination strategies are rather fragmented, i.e. in term of applying into 3D planar polygon instead of 2D, or if y-intercept crosses the vertices or line of 3D polygon in certain special cases.

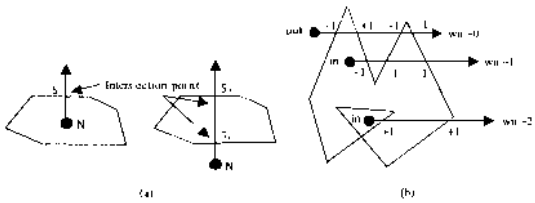


Figure 1: Determination of 0D feature in polygon, (a) *odd-even*, and (b) *winding methods*

In this paper, we focus on simple but complete strategy in determining the 0D feature inside or outside 3D planar polygon. Section 2 discusses the 3D plane equation and covers some examples in order to verify the mathematical function. Then, the determination of 0D feature *on/above/below* the 3D planar polygon is described in section 3. The determination will only be continued to next stage if this test is successfully passed, i.e. 0D feature is *on* the 3D planar polygon. The determination of 0D feature is located *inside* or *outside* 3D planar polygon will be discussed in detail in section 4. The paper presents the experiments on the algorithm in section 5 and finally, the concluding remarks of the work.

2. The 3D Plane Equation

There are many ways to represent a plane π. Some work in any dimension, and some work only in 3D. In any dimension, one can always specify 3 ***non-collinear*** points $P_0=(X_0,Y_0,Z_0)$, $P_1=(X_1,Y_1,Z_1)$, $P_2=(X_2,Y_2,Z_2)$ as the vertices of a triangle, the most primitive planar object. In 3D, this uniquely defines the plane of points (x,y,z) satisfying the equation:

$$\begin{vmatrix} X-X_0 & Y-Y_0 & Z-Z_0 \\ X_1-X_0 & Y_1-Y_0 & Z_1-Z_0 \\ X_2-X_0 & Y_2-Y_0 & Z_2-Z_0 \end{vmatrix} = 0 \quad \text{(Eq. 1)}$$

$$\mathbf{Ax + By + Cz + D} = 0 \quad \text{(Eq. 2)}$$

This determinant is satisfying general form of plane equation, with normal, Pn = (A, B, C), and (x,y,z) denotes any point on the plane

For any plane more than 4 vertices, plane equation is same as the equation of a triangle formed by any 3 points. As both polygon and triangle in Figure 2 have the same plane equation if and only if they are co-planar. In order to compute the equation of any polygon, which more than 3 vertices, any 3 vertices can be used to calculate the plane equation. Figure 3 denotes an example of calculating the plane equation from rectangular.

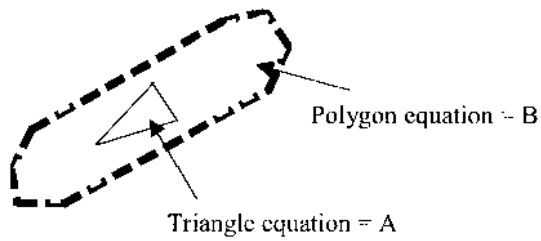


Figure 2: Both Triangle and polygon are co-planar

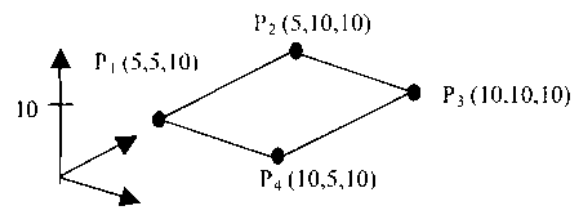


Figure 3: Rectangular

To calculate the plane equation of a polygon that more than 3 vertices, e.g. rectangular from Figure 3, only 3 vertices will be used. In the 1st case, P_1, P_2, and P_3 are used. The plane equation of the rectangular in Figure 2 is: **Given P_1 = (5,5,10), P_2 = (5,10,10), and P_3 = (10,10,10):**

$$\begin{vmatrix} X-X_1 & Y-Y_1 & Z-Z_1 \\ X_2-X_1 & Y_2-Y_1 & Z_2-Z_1 \\ X_3-X_1 & Y_3-Y_1 & Z_3-Z_1 \end{vmatrix} = 0$$

$$\Rightarrow \begin{vmatrix} X-(5) & Y-(5) & Z-(10) \\ (5)-(5) & (10)-(5) & (10)-(10) \\ (10)-(5) & (10)-(5) & (10)-(10) \end{vmatrix}$$

$$\Rightarrow (X-5)\bullet[0-0]-(Y-5)\bullet[0-0]+(Z-10)\bullet[0-25] = 0$$

$$\Rightarrow [0X-0Y-25Z+250]/-25 = 0/-25$$

(Simplify the equation by deviding a factor of (-25)

$$\Rightarrow 0X+0Y+Z-10 = 0$$

Therefore, P_n (A, B, C) = P_n (0, 0, 1), and D = -10.

In order to prove the plane equation is unique for any co-planar polygon, P_1, P_2, and P_4 are used. In the 2nd case, the plane equation is :

Given P_1 = (5,5,10),P_2=(5,10,10), P_4=(10,5,10);

$$\begin{vmatrix} X - X_1 & Y - Y_1 & Z - Z_1 \\ X_2 - X_1 & Y_2 - Y_1 & Z_2 - Z_1 \\ X_4 - X_1 & Y_4 - Y_1 & Z_4 - Z_1 \end{vmatrix} = 0$$

$$\Rightarrow \begin{vmatrix} X - (5) & Y - (5) & Z - (10) \\ (5) - (5) & (10) - (5) & (10) - (10) \\ (10) - (5) & (5) - (5) & (10) - (10) \end{vmatrix} = 0$$

$$\Rightarrow (X - 5)\bullet[0 - 0] - (Y - 5)\bullet[0 - 0] + (Z - 10)\bullet[0 - \ldots = 0$$

$$\Rightarrow [0X + 0Y - 25Z + 250] = -25 \quad = 0 - -25$$

These 2 cases prove that:

For any polygon that more than 3 vertices, the plane equation is unique and it can be computed using only 3 vertices.

3. Determination of 0D Feature on/above/below 3D Planar Polygon

To determine a 0D feature is inside a 3D planar polygon, one important condition must be fulfilled in the first place. The 0D feature must located on the 3D plane. In order to verify this condition, the plane equation from Eq. (2) is followed:

$Ax + By + Cz + D = 0$, (from Eq. 2)

with normal, P_n = (A, B, C), and 0D feature = (x,y,z)

Take note that the 3D polygon is created from a set of points. The sequence of this point set is important in determining a 0D feature is located *on*, *above* or *below* a 3D planar polygon. If the sequence of points is counter-clockwise, any point that fulfils $Ax + By + Cz + D > 0$ denotes that a 0D feature is *above* the 3D plane. Conversely, a 0D feature is located *below* the 3D plane when $Ax + By + Cz + D < 0$. A 0D feature is located *on* the 3D plane when $Ax + By + Cz + D = 0$. Figure 4 denotes the a 0D feature is located in two different cases.

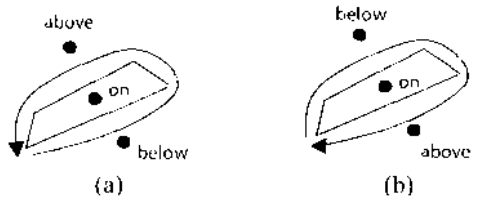


Figure 4: The point set sequence of a polygon: (a) counter-clockwise, and (b) clockwise

4. Determination of 0D Feature inside/outside 3D Planar Polygon

After a 0D feature is verified either *on/above/below* a 3D planar polygon, the second condition is followed. The 0D feature must within the boundary or interior of a 3D planar polygon. To determine whether a 0D feature is inside or outside a 3D polygon, the XY plane is tested in the first place. If a 0D feature is located at the interior of 3D polygon, the YZ plane test will be ignored. This is because only 0D feature that is co-planar with the 3D polygon will be tested. If the 0D feature is located above or below the 3D planar polygon, it is definitely outside the polygon. Conversely, if the 0D feature is located at the border of 3D polygon, the YZ plane test will be carried out.

4.1 The XY plane test

In most of the common test for detecting point inside polygon, either x-intercept or y-intercept ray are implemented. Figure 5 denotes a simple test for point inside 2D polygon. If the amount of intersecting point between y-intercept ray and the border of polygon is even, then the point is outside the polygon, otherwise is inside.

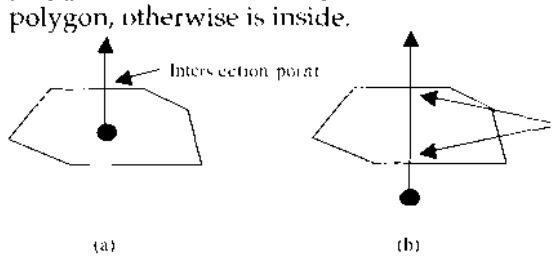


Figure 5: Amount of intersecting point, (a) odd, and (b) even

However, there are 4 special cases exist in reality. There are:

Case 1: y-intercept ray intersects the border of polygon at a line (one of the intersection point)

In certain case, the y-intercept ray crosses the border of polygon at a line. Figure 6 denotes the example of intersection between

y-intercept ray and border of polygon. In this case, the intersection will be considered. The total number of intersection (from Figure 6a) is 1, although the y-intercept ray intersects 2 vertices or a line.

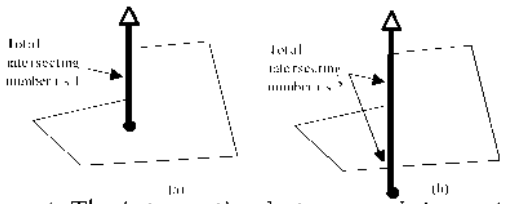


Figure 6: The intersection between y-intercept ray and border of polygon,
(a) Inside, and (b) Outside

Case 2: y-intercept ray intersects the border of polygon at a line (NOT one of the intersection point)

In this case, the y-intercept ray also crosses the border of polygon at a line. Figure 7 denotes the example of intersection between y-intercept ray and border of polygon. The intersection will not be considered. The total number of intersection (from Figure 7a) is 1, although the y-intercept ray intersects a line and a vertex.

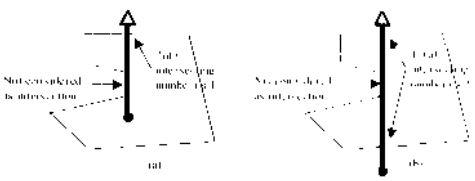


Figure 7: The intersection between y-intercept ray and border of polygon,
(a) Inside, and (b) Outside

Case 3: y-intercept ray intersects the border of polygon at a point (one of the intersection point)

In certain case, the y-intercept ray crosses the border of polygon at a point. Figure 8 denotes the example of intersection between y-intercept ray and vertex of polygon. In this case, the intersection will be considered. The total number of intersection (from Figure 8a) is 1.

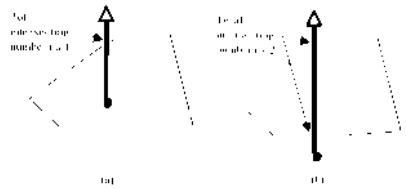


Figure 8: The intersection between y-intercept ray and border of polygon,
(a) Inside, and (b) Outside

Case 4: y-intercept ray intersects the border of polygon at a line (NOT one of the intersection point)

In certain case, the y-intercept ray crosses the border of polygon at a point. Figure 9 denotes the example of intersection between y-intercept ray and vertex of polygon.

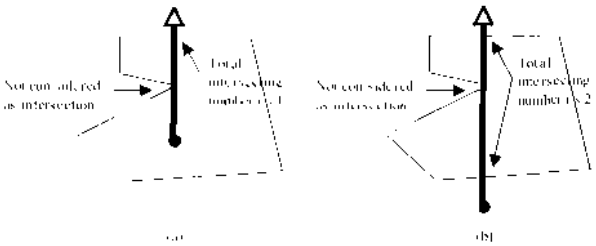


Figure 9: The intersection between y-intercept ray and border of polygon,
(a) Inside, and (b) Outside

In this case, the intersection will not be considered. The total number of intersection (from Figure 9a) is 1, although the y-intercept ray intersects 2 vertices.

4.2 The YZ plane test

The YZ plane test will be carried out if and only if the 0D feature is located at the border of the 3D polygon at XY plane. Figure 10 denotes the 0D feature is located at the border of 3D polygon at XY plane.

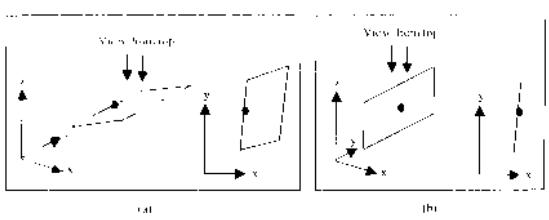
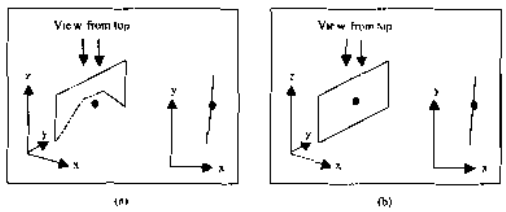


Figure 10: 0D feature is located at the border of 3D polygon at XY plane

In certain case, 0D feature is located at the

border of 3D polygon at XY plane, but it is located outside the polygon in reality. Figure 11 denotes the 0D feature is located at border of 3D polygon at XY plane but outside the polygon in reality. Figure 11: 0D feature is located at border of 3D polygon at XY plane, but (a) outside, and (b) inside polygon in reality



Refer to the Figure 11, the YZ plane test needs to be carried out in order to verify correctly the 0D feature is located either *inside* or *outside* the 3D planar polygon. However, the YZ plane test is same as the XY plane. The YZ test needs also to compute the amount of intersection between the y-intercept ray of 0D feature and border of 3D polygon. If the amount is even, the 0D is located outside the polygon, otherwise is located inside the 3D polygon. All the special cases mentioned in XY plane test will need to follow as well.

5. The Experiment

The experiment is implementing the C++ program to test the approach mentioned in section 2, 3, and 4. Figure 12 denotes the methodology for the determination of the 0D feature is located *inside/outside* 3D planar polygon.

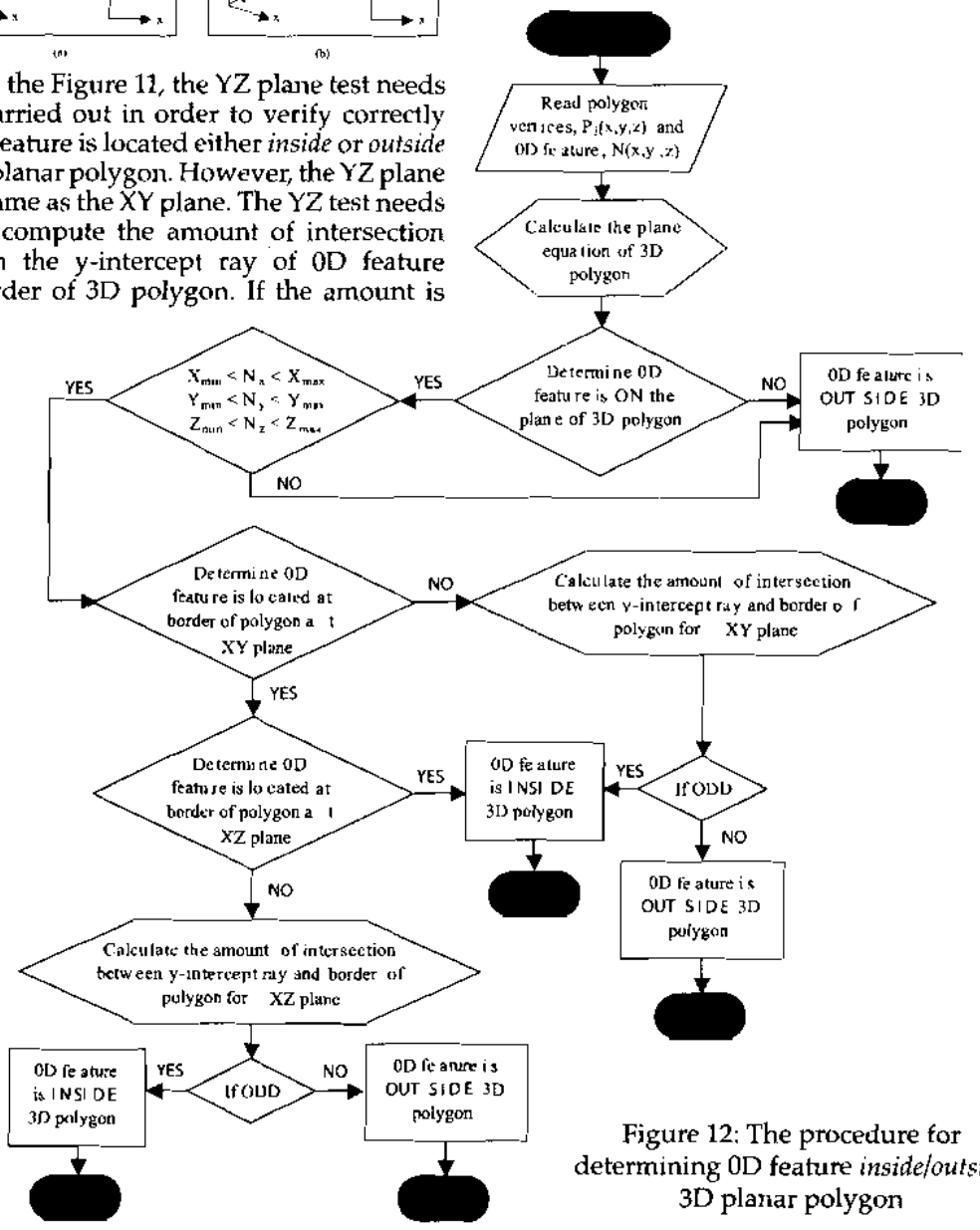


Figure 12: The procedure for determining 0D feature *inside/outside* 3D planar polygon

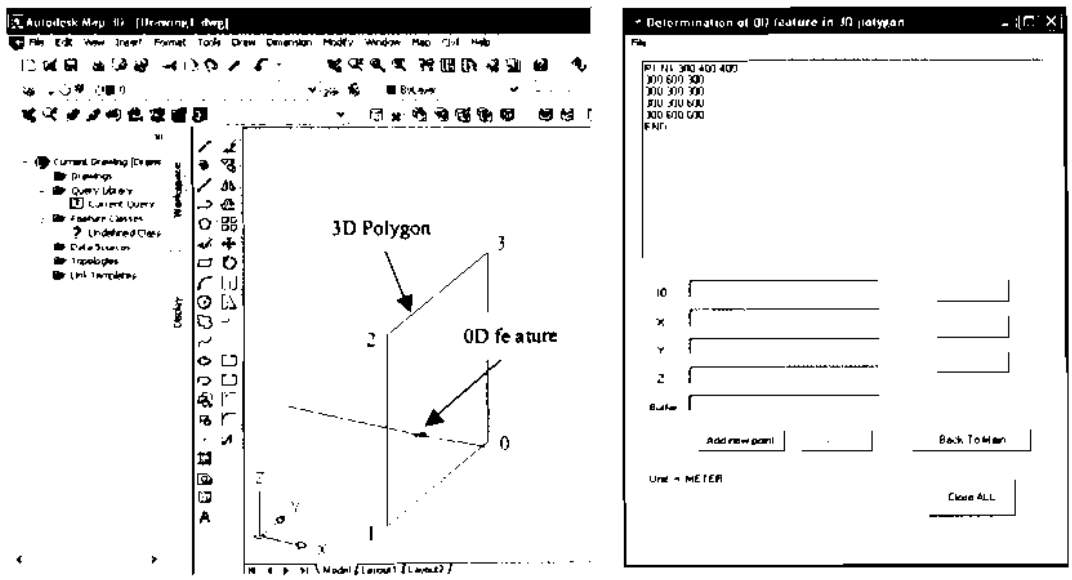


Figure 13: (a) Experiment, and (b) Software module

The experiment is given in Figure 13 is to verify the capability of the methodology from Figure 12 in producing the correct result. C++ program is used to create a software interface that consists of all related algorithms given from Figure 11. In this experiment, a 3D polygon created from 4 vertices is tested. There are P_1 (300,600,300), P_2 (300,300,300), P_3 (300,300,600), and P_4 (300,600,600). The 0D feature is given as N (300,400,400). The software module read the dataset from a text file that consists of the 3D polygon and 0D feature dataset. The interface read P1 as the ID number for 3D planar polygon, whereas N1 is the ID number for 0D feature (see Figure 13b). The following triplet is the dataset for 0D feature. Later on, the vertices for 3D polygon are followed until the character "END" is reached. The result from the implementation is given as a test file (see Figure 14).

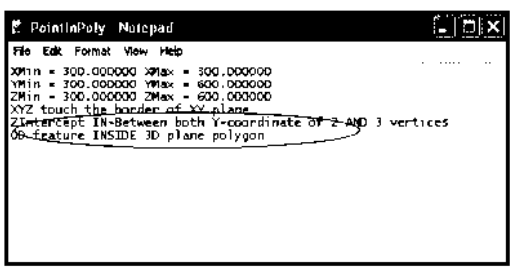


Figure 14: The result

6. Concluding Remarks

The paper presents an approach for determining 0-D feature inside/outside 3D planar polygons. This research work could provide an initial analytical computation for 3D spatial analysis as they are needed in such 3D spatial systems (i.e. 3D GIS). Providing such operations for the systems is one of our research efforts. The proposed method and algorithm was programmed using C++ language. Initial results show the methodology works for the typical 3D planar polygons. These works could be extended and incorporated into 3D GIS analytical operations problems include:

1) 3D union,
2) 3D intersection, and other operations.

Obviously, the initial outcomes of this experiment certainly would be beneficial to the development of 3D GIS software.

References

[1] Foley, J.D., A. van Dam, S.K. Feiner, J.F. Hughes (1990), Computer Graphics: Principles and Practice, 2nd Edition, Addison-Wesley.

[2] Franklin, R. (2005), PNPOLY - *Point inclusion in polygon test*, http://www.ecse.rpi.edu/Homepages/wrf/geom/pnpoly.html

[3] Haines, E. (1994a), *point in polygon inside/outside code*, http://www.acm.org/tog/GraphicsGems/gemsiv/ptpoly_haines/ptinpoly.c

[4] Haines, E. (1994b), *Point in polygon strategies*, In: P. Heckbert (Ed.), Graphic Gems IV, Academic Press, Boston, MA, pp. 24-46.

[5] Harrington, S. (1983), *Computer Graphics: A Programming Approach*, McGraw-Hill.

[6] Mehlhorn, K., and S. Näher, (1999), LEDA: *A Platform for Combinatorial and Geometric Computing*, Cambridge University Press.

[7] Nievergelt, J., and K. Hinrichs (1993), *Algorithms and data structures: with applications to graphics and geometry*, Prentice-Hall.

[8] O'Rourke, J. (1998), *Computational geometry* in C, 2nd Edition, Cambridge University Press.

[9] Sedgewick, R. (1998), *Algorithms*, 2nd Edition, Addison-Wesley.

[10] Stein, B. (1997), *A point about polygons*, Linux Journal 35.

[11] Theoharis, T., and A. Böhm (1999), *Computer graphics: principles & algorithms*, Symmetria.

[12] Weiler, K. (1994), *An incremental angle point in polygon test*, In: P. Heckbert (Ed.), Graphic Gems IV, Academic Press, Boston, MA, pp. 16-23.

[13] Woo, M., J. Neider, and T. Davis (1997), *OpenGL programming guide*, 2nd Edition, Addison-Wesley.

Appendix

```
FILE *InputData = fopen("Input.dat", "rt" );
FILE *OutputData = fopen("Output.dat", "w" );

// Read planar polygon dataset, P and 0D data, N(X,Y,Z).
// Calculate the plane equation of planar polygon
// Ax + By + Cz + D = 0

IF (A(xN) + B(yN) + C(zN) + D = 0 )
{
  // N(X,Y,Z) is located on the planar polygon
  // N(X,Y,Z) will be tested for other conditions
}
else
{
  // N(X,Y,Z) is NOT located on the planar polygon
     fprintf (OutputData, "0D is OUTSIDE planar polygon = %f %f %f \n", Nx, Ny, Nz);
     exit;
}

// Determination of minimum & maximum of x-coordinate from polygon P
// Determination of minimum & maximum of y-coordinate from polygon P
// Determination of minimum & maximum of z-coordinate from polygon P

IF ((Min xP < xN < Max xP) && (Min yP < yN < Max yP) && (Min zP < zN < Max zP))
{
  // N(X,Y,Z) is located on the planar polygon
  // N(X,Y,Z) will be tested for other conditions
}
else
{
  // N(X,Y,Z) is NOT located on the planar polygon
     fprintf (OutputData, "0D is OUTSIDE planar polygon = %f %f %f \n", Nx, Ny, Nz);
     exit;
}

// Compute N(X,Y,Z) INSIDE or OUTSIDE
polygon for the (X & Y) and (Y & Z) plane

IF (N(X,Y,Z) touch the BORDER of polygon P in the X & Y plane)
{
   // check the Y & Z plane

   IF (N(X,Y,Z) touch the BORDER of polygon P in the Y & Z plane)
   {
      // N(X,Y,Z) is located INSIDE the polygon P
      fprintf (OutputData, "0D is INSIDE planar polygon = %f %f %f \n", Nx, Ny, Nz);
   }
   ELSE
```

```
    {
        //   Calculate the amount of intersection point of Z-axis
        IF (Amount is odd)        //  MUST involve all special cases, please refer to Sec. 4.1
        {
             //   N(X,Y,Z)  is located INSIDE the polygon P
             fprintf (OutputData, "0D is INSIDE planar polygon =  %f   %f   %f \n", Nx, Ny,
Nz);
        }
        ELSE
        {
             //   N(X,Y,Z)  is located OUTSIDE the polygon P
            fprintf (OutputData, "0D is OUTSIDE planar polygon =  %f   %f   %f \n", Nx, Ny,
Nz);
        }
    }
}
ELSE
{
    //   Calculate the amount of intersection point of Y-axis
    IF (Amount is odd)        //  MUST involve all special cases, please refer to Sec. 4.1
    {
         //   N(X,Y,Z)  is located INSIDE the polygon P
        fprintf (OutputData, "0D is INSIDE planar polygon =  %f   %f   %f \n", Nx, Ny, Nz);
    }
    ELSE
    {
         //   N(X,Y,Z)  is located OUTSIDE the polygon P
         fprintf (OutputData, "0D is OUTSIDE planar polygon =  %f   %f   %f \n", Nx, Ny,
Nz);
    }
}
```