3D SOLID BUFFERING FOR 3D GIS

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

3D analysis is one of the important components in GIS as it defines as decision making tools for geographic features. Next generation of GIS software would highly dependent on the 3D analysis in solving geospatial problems. One of the desired components in the next generation software is to deal with the 3D analytical solutions. This paper presents a portion of the 3D spatial information problems, that is 3D solid buffering. Discussions related to the implementation of buffering model for solid object are the main concern in this paper. The primitives of objects such as point, line, and face will be addressed and integrated for the development of 3D solid buffering. The mathematics, the geometry, and the algorithms involved in the development will be presented. The approach is verified by using simulated data sets and a commercial GIS package for visualization purposes. Finally, we discuss the outlook of the 3D analytical solutions for 3D GIS.

Key words: 3D solid buffering, and 3D GIS

1.0 INTRODUCTION

3D buffering zone, of which the definition is an operation to generate proximity information of a spatial object, for instances, phenomenon along linear features or polygon features in a three-dimensional space. It is categorized as one of the 3D analytical solutions. In GIS, analytical tools are important for geographic analysis. User may want to know the shortest road from a place to another place, the distance between two places, or even the area of a housing estate. All these demands increase the need of the analysis tool in geographic information (Bernhardsen (1999); Gaiani, *et. al* (2001); Koh and Chen (1999); Zlatanova, *et. al* (2002)). Therefore, the geometrical modeling for buffering zone of the 3D solid object will be discussed in this research. In the section 2, each of the buffering objects will be studied in detailed (i.e. how to construct them geometrically) based on the geospatial primitives (point, line, and polygon. The approach of the buffering object for each of the primitives will be mentioned. All the mathematics that follows the expected buffering result will be highlighted. In the section 3, 3D solid buffering will be discussed, together with the approach and mathematics. The experiment and discussions will be mentioned in section 4. Section 5 concludes the research and research finding.

2.0 3D BUFFERING FOR PRIMITIVES OF OBJECTS

Any feature object appears in the real world is constructed by primitive object. The feature object could be a road, building, or just a simple traffic sign. They are represented by 3 kinds of primitives, i.e. point, line, and polygon. A point could construct the point itself, lines, faces and solid objects, whereas line is part of lines, faces or solid objects. From the Figure 1, a complete constructive model for primitive object is given.

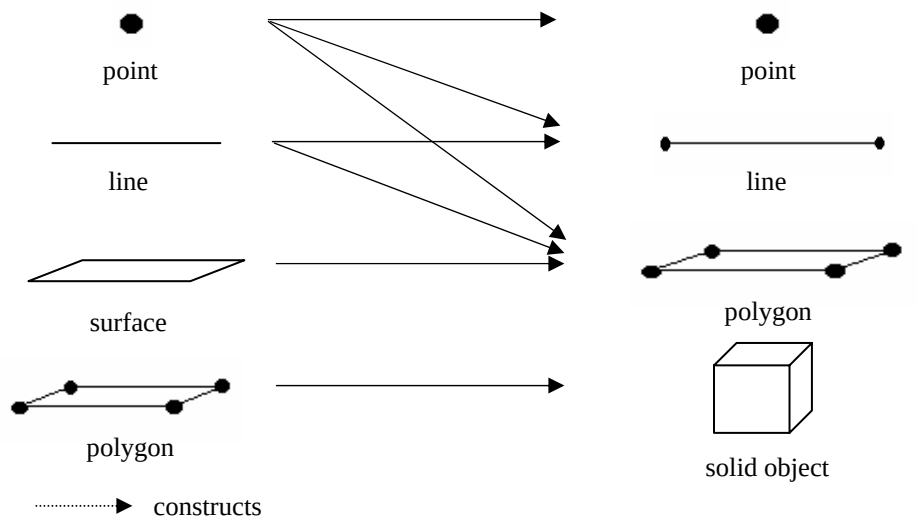


Figure 1: Geospatial object

Any spatial object could be used to produce spatial analysis, e.g. 3D buffering model. From the Figure 1, the buffering models of each feature object are different according to the primitive that constructs them. The buffering model for primitive point is sphere, whereas the primitive line is a cylinder. Since nodes appear in the line segment, adding the point buffering model into the line buffer model becomes a necessity. Figure 2 shows the method to create a line buffering output.

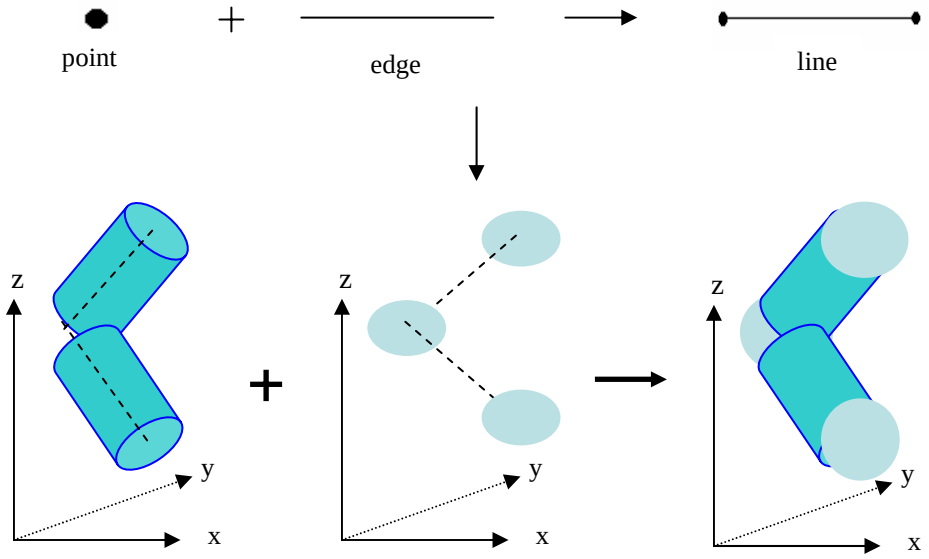


Figure 2: Line's properties and buffering model

Refer to Figure 2, the buffering model for a line is a combination between spheres and cylinder. However, this combination creates an overlap segment (see Figure 3). This unnecessary segment is the internal segment and need to be removed. The complete modeling of line buffering object is discussed in Chen and Abdul-Rahman (2003).

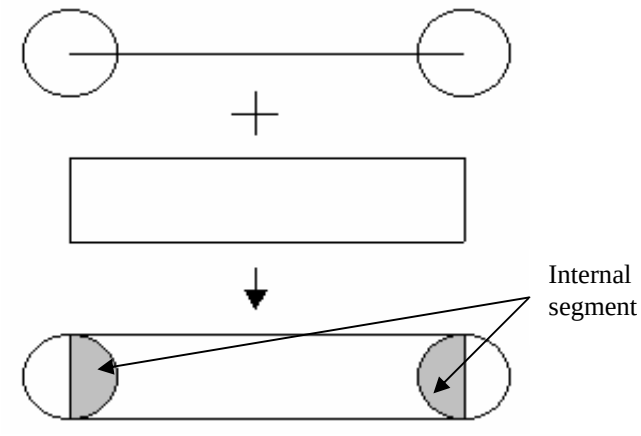


Figure 2: Internal segment of line buffering model

The third kind of primitive is polygon. A polygon feature is represented by some arc/node structure that determines an area's boundary. With at least three or more nodes joining together consecutively and return to the starting node define a polygon. The buffering model for polygon is constructed by upper and lower faces (see Figure 3). However, to perform geometrically a buffering operation towards a polygon in three-dimensional space, the joined components of a single polygon need to be identified, which are the nodes, and edges. Therefore, the point and line buffering model should be considered in constructed polygon buffering model. The combination of point, line and face buffering model create internal segments that need to be removed. The complete modeling of polygon buffering object, including the removal of internal segment is discussed in Chen and Abdul-Rahman (2003).

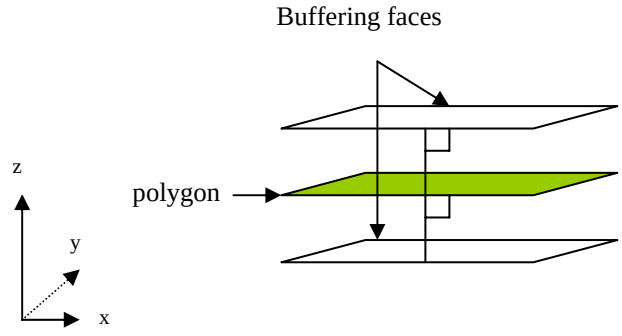
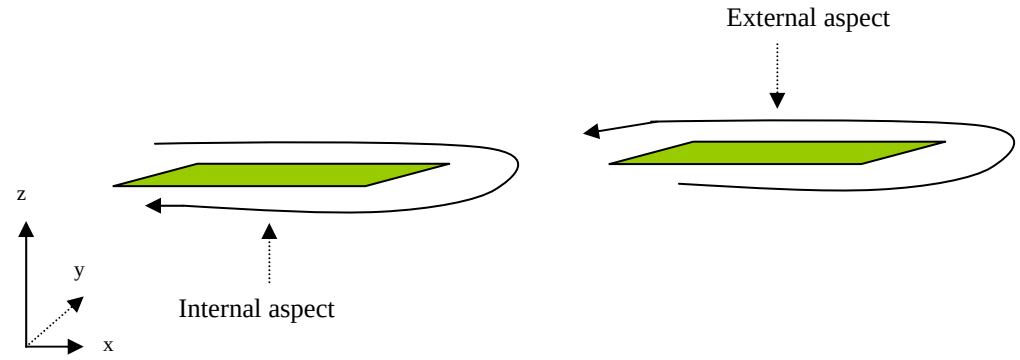


Figure 3: Two buffering faces

3.0 3D SOLID BUFFERING MODEL

A solid object constructed by faces could perform the buffering process. These faces connect each other, which shares the common vertices and lines, form a closed boundary. Therefore, the

construction rule for a face is important in determine the internal and external aspect of a solid object. This is because the external aspect will be used in modeling the solid buffer zone. A series of points that construct a face in counter-clockwise direction is facing the external aspect; otherwise is internal. The main idea of differentiating the face's aspect is to determine the upper and lower faces for polygon buffering (see Figure 4).



(a) Clockwise direction (b) Counter clockwise direction

Figure 4: Upper and lower faces of polygon buffering model

The buffering layers are expanded upper and lower starting from the center of polygon, respective to the buffer length. Both of them should be parallel to the main polygon. To define these polygons buffer, the vector's cross product is used. This vector produces a result that is perpendicular to the main polygon (*v* and *u*). For vectors *u* and *v*, the cross product is defined by (see Figure 5):

$$\begin{aligned} u \times v &= \hat{x}(u_y v_z - u_z v_y) - \hat{y}(u_x v_z - u_z v_x) + \hat{z}(u_x v_y - u_y v_x) \\ &= \hat{x}(u_y v_z - u_z v_y) + \hat{y}(u_z v_x - u_x v_z) + \hat{z}(u_x v_y - u_y v_x) \end{aligned} \qquad \text{(Eq.1)}$$

This can be written in a shorthand notation, which takes the form of a determinant (see Eq.1).

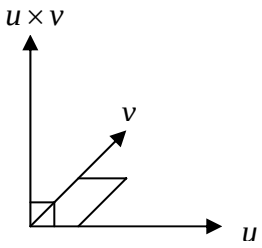


Figure 5: Vector (*u* x *v*)

Here, the (*u* x *v*) is always perpendicular to both *u* and *v*, with the orientation determined by the right-hand rule.

$$u \times v = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix} \qquad \text{(Simplified from Eq.1)}$$

Some of the mathematics is illustrated below (refer to Figure 6). The vector *u* (A) and *v* (B) need to be compute in the early step (see Figure 6 and Equation 2 to 5). The purpose of computing those vectors is to find out the vector scale [$|N_X|$, $|N_Y|$, $|N_Z|$]. Later on the vector scale are used to compute the point sets that form the polygon.

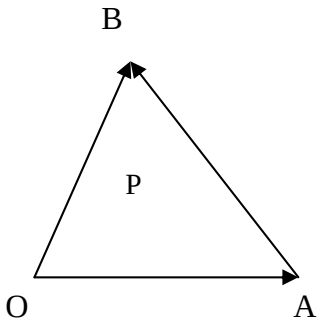


Figure 6: Vector for polygon P

$$\overrightarrow{OA} \text{ x } \overrightarrow{OB} = \begin{bmatrix} |n_1| & |n_2| & |n_3| \end{bmatrix} \qquad \text{(Eq.2)}$$

$$|n_1| = \begin{vmatrix} Y_{oa} & Z_{oa} \\ Y_{ob} & Z_{ob} \end{vmatrix} \qquad \text{(Eq.3)}$$

$$|n_2| = \begin{vmatrix} Z_{oa} & X_{oa} \\ Z_{ob} & X_{ob} \end{vmatrix} \qquad \text{(Eq.4)}$$

$$|n_3| = \begin{vmatrix} X_{oa} & Y_{oa} \\ X_{ob} & Y_{ob} \end{vmatrix} \qquad \text{(Eq.5)}$$

$$\vec{AB} = \sqrt{\left(|n_1|^2 \quad |n_2|^2 \quad |n_3|^2 \right)} \qquad \text{(Eq.6)}$$

$$\hat{AB} = \frac{\begin{bmatrix} |n_1| & |n_2| & |n_3| \end{bmatrix}}{\vec{AB}}$$

$$= \left(|N_x| \quad |N_y| \quad |N_z| \right) \qquad \text{(Eq.7)}$$

$$x_c = x_o + |N_x| \times r \qquad \text{(Eq.8)}$$

$$y_c = y_o + |N_y| \times r \qquad \text{(Eq.9)}$$

$$z_c = z_o + |N_z| \times r \qquad \text{(Eq.10)}$$

where x_c, y_c, z_c are the processed coordinates for the point *C*.

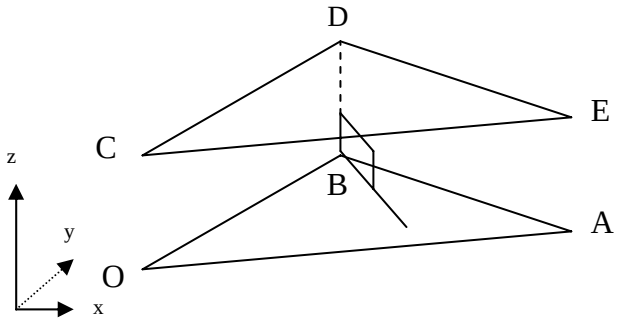


Figure 7: Upper polygon buffer CDE

Later on, point D and E will be calculated (see Figure 7) by using the same equations start from 2 to 10. From the Figure 7, three points (*C*, *D*, and *E*) create an upper layer of face for a polygon buffer model. Recall that a polygon is a combination of point, line and surface. The polygon buffering zone should implement point and line buffer model. After the upper polygon buffer is defined, the line (cylinders) and point (sphere) buffering objects need to be combined. However, the combination may produce internal segment between line and the polygon buffer. Same situation also occurs between point and line buffer. Therefore, the unnecessary internal segment needs to be removed.

To remove the internal segment between point and line buffer, the same approach as shown in Figure 2 is used. On the other hand, to remove the internal segment between line and polygon buffer, the approach is shown in Figure 8.

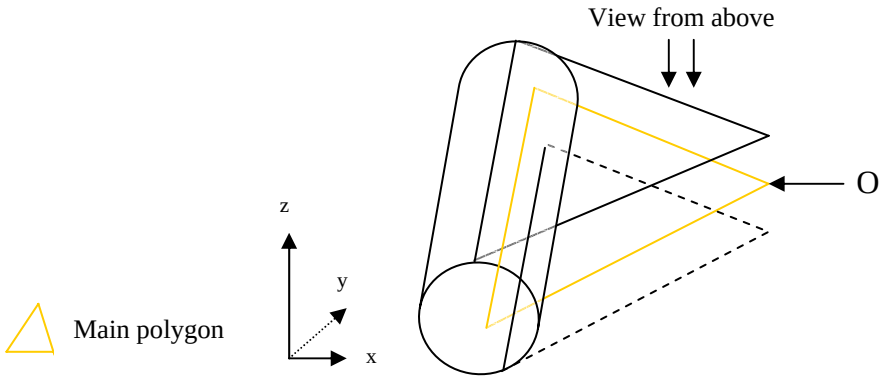


Figure 8: Combination of cylinder and surface buffer

The cylinder and the point O from polygon (refer to the Figure 9), which is the point opposite to the cylinder and it is being a part of the main polygon. Either the third point that creates a polygon is O, O', or even O" from the Figure 9, the representation remains the same as in Figure 10.

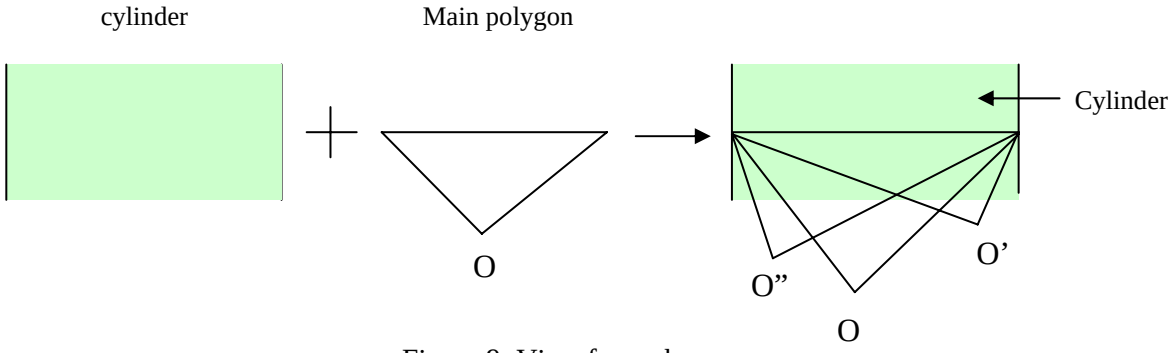


Figure 9: View from above

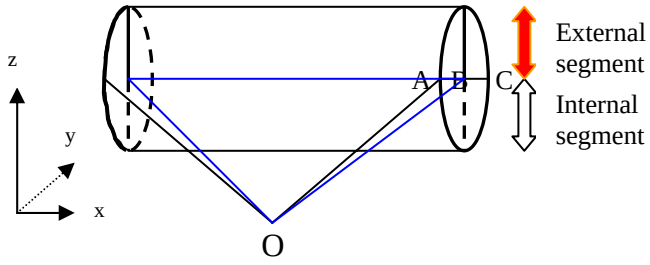


Figure 10: Method to remove internal segment

The cylinder is divided into two segments, which are the internal and external part. The main problem is to determine where the internal is and the external segment. Refer to Figure 11, *d* is the distant limit. The distance between cylinder segment and *O* (from the polygon) that exceeds *d* are identified as external

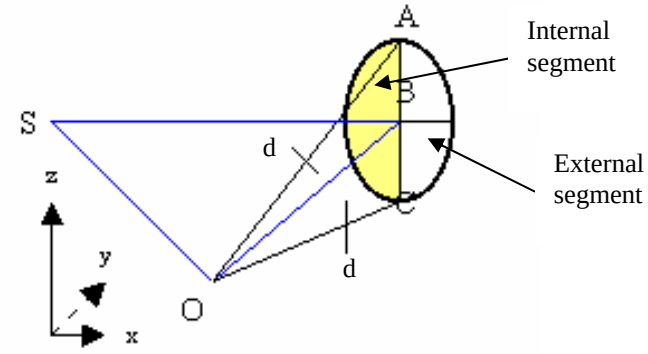


Figure 11: Calculate distance between OA, OB, and OC

The distance of OA, and OC will be computed. Both of them are in the same situation because the surface of circle is perpendicular to the line AC. Points A and C are the upper and lower point of the circle (see Figure 11). Therefore, there is a condition that needs to be fulfilled in order to remove the internal segment of the cylinder. That is the distance of either OA or OC is the benchmark of that condition. If any distance from the circle to point O is less that OA or OC, then it should be the internal part of the cylinder and need to be removed. Finally, Figure 12 shows the processed result.

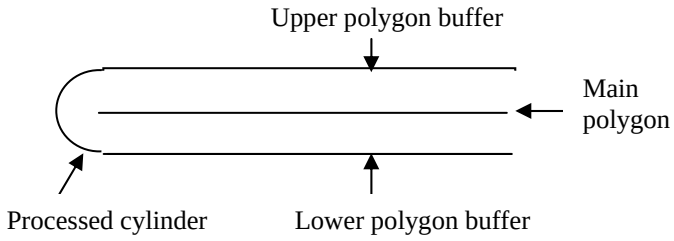


Figure 12: View from above

The remaining external segment (see Figure 12) will be used for another process, which the adjacent polygon buffer is combined (see Figure 13).

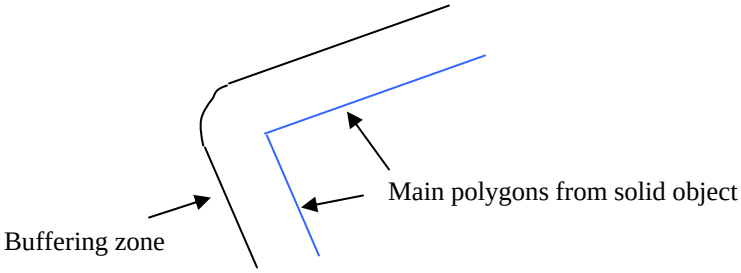


Figure 13: Final buffering zone

4.0 THE EXPERIMENT AND DISCUSSIONS

In this project, the C++ language is used to create a software interface module called *3D Solid Buffering Tools*. It is tested using simulated dataset. The representation of 3D solid buffering module / software will display the result using a commercial package called 3D Analyst. The input is in ASCII format and the result is presented in the shapefile (*.shp) format. Figure 14 shows the snap shots of the interface of the software, whereas Figure 15 and 16 present the output of the research.

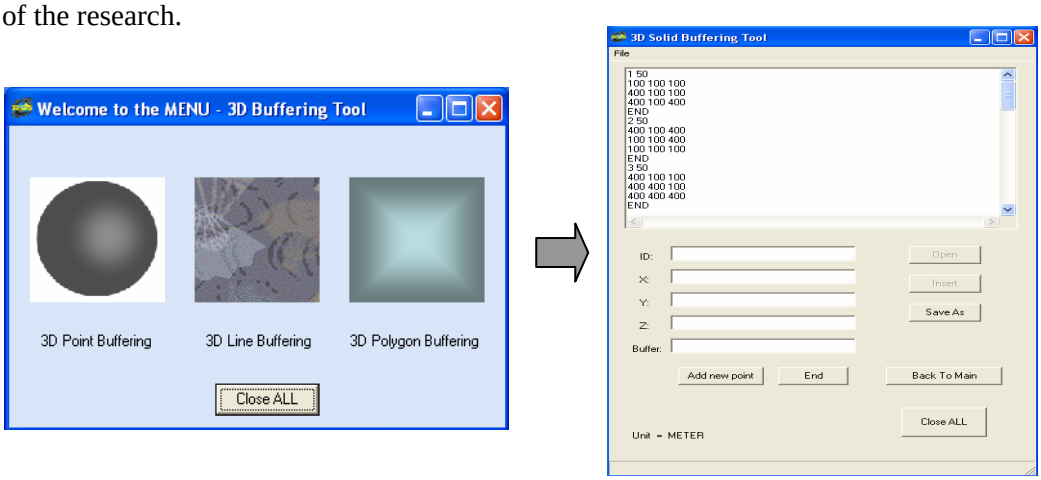
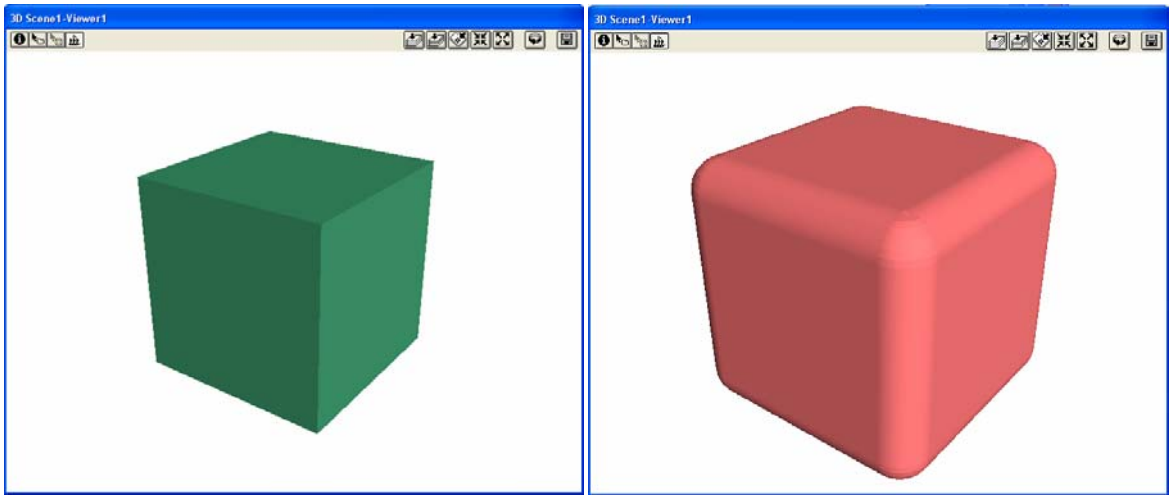


Figure 14: Software interface



(a) Input dataset (b) Solid buffering result

Figure 15: Solid buffering for simple object

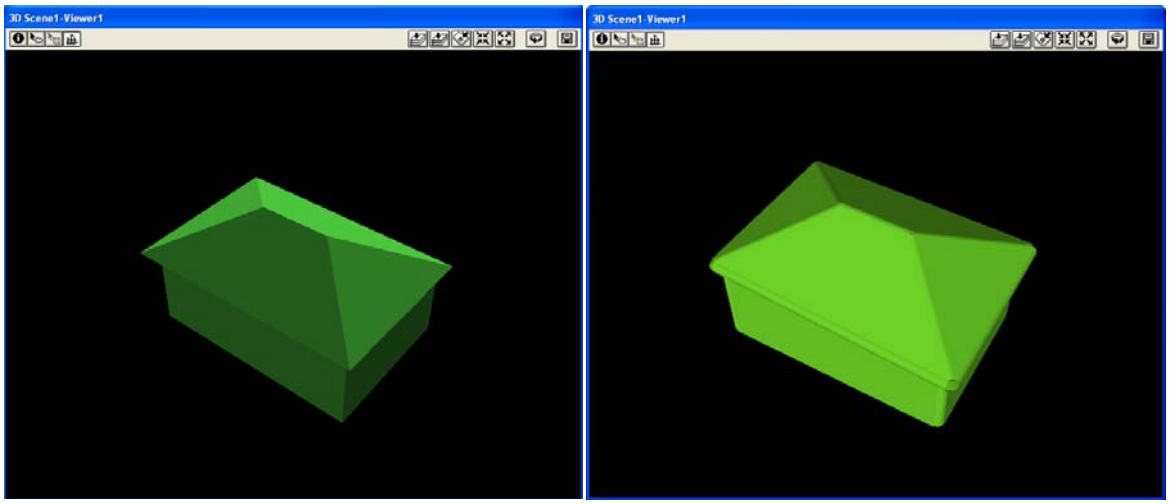


Figure 16: Solid buffering for complex object

5.0 CONCLUDING REMARKS

The paper presents an approach for integrating all geometrical modeling of primitive buffering object in solving new spatial operations, i.e. 3D solid buffering. The integration is expected to be providing complete modeling for measurement-based 3D GIS analysis. The methodology was tested using C++ program and the result gives a positive output respected to the algorithm. Obviously, the initial outcomes of this experiment certainly will be utilized in the 3D GIS software development in the very near future, i.e. the 3D analytical operations.

REFERENCES

Chen, T.K., and Abdul-Rahman, A., (2003) "The Mathematics Of 3D Buffering For Simple Geospatial Primitives – An Effort To Realize 3D Buffering Operation For 3D GIS". Geoinformation Science Journal, Vol. 3, No. 1, pp. 66-73.

Bernhardsen, Y., (1999) "Choosing a GIS in: Geographical Information Systems: Principles, Technical, Management and Applications". John Wiley & Sons, Inc., 2nd edition, pp. 580-600.

Gaiani, M., Gamberini E. and Tonelli, G., (2001) "VR As Work Tool For Architectural & Archaeological Restoration: The Ancient Appian Way 3D Web Virtual GIS". Virtual Systems and Multimedia, Proceedings. Seventh International Conference, 25-27 Oct. 2001, pp. 86 –95.

Koh, B., and Chen, T., (1999) "Progressive Browsing Of 3D Models". Multimedia Signal Processing, IEEE 3rd Workshop, pp. 71-76.

Zlatanova, S., Abdul-Rahman, A., and Pilouk, M., (2002) "Present Status of 3D GIS". G.I.M. International, pp 41-43.

APPENDIX

The algorithm that implements the mathematics for solid buffering is given below:

```
{
  declare all the variable;

  WHILE (if data != EOF)
  {
      WHILE (read data file != END)
      {
          read input data: ID, X, Y, Z, Buffer Length;
          calculate the amount of data: n++;
      }

  Calculate the Vector for X (point A and B);
  Calculate the Vector for Y (point A and B);
  Calculate the Vector for Z (point A and B);
  Calculate the determinant for upper-plane;
  Calculate the Vector length for upper-plane;
  Calculate the Normalized Vector for upper-plane;

  FOR (i=0;i<n;i++)
  {
      compute the upper polygon for buffering;
      PRINT to file A;
  }

  Extract the entire line data from polygon with no redundancy;
  (Generate line buffering dataset)
```

```
FOR (k=0;k<n-1;k++)
{
        Calculate the rotation angle for x-axis;
        Calculate the rotation angle for y-axis;
        Calculate the rotation angle for z-axis;

        FOR (a=0;a<360;)
        {
                compute the entire dataset for a circle;
                rotate the circle towards the x-axis;
                rotate the circle towards the y-axis;

                Determine the partial rotation of z-axis;
                Substract the partial rotation with the rotation angleof z-axis;
        }
}

FOR (k=0;k<n-1;k++)
{
        FOR (a=0;a<360;)
        {
                compute the entire dataset for a circle;
                rotate the circle towards the x-axis;
                rotate the circle towards the y-axis;
                rotate the circle towards the z-axis;
        }
}

FOR (k=0;k<n-1;k++)
{
        FOR (a=0;a<360;)
        {
                PRINT the entire dataset into file A;
        }
}
        extract all the point from line data with no redundancy;

        (Generate point buffering dataset)
        FOR (k=0;k<n-1;k++)
        {
                Compute the array of Z value;
                Compute the array of buffer length correspond to the Z value;

                FOR (i=0;i<360;i++)
                {
                        compute the circle using processed buffer length;
                }
                compute the upper and lower polar points;

                compute the maximum distance to remove the internal segment;
```

```
            FOR (i=0;i<360;i++)
            {
               IF (buffering datasets > maximum distance)
               {
                  PRINT to file A;
               }
            }

            FOR (i=0;i<360;i++)
            {
               IF (buffering datasets > maximum distance)
               {
                  PRINT to file A;
               }
            }
      }

      OPEN processed data file A;
      READ data from the file A;
      CALL the *.dbf routine and generate *.dbf file;
      CALL the *.shp routine and generate *.shp file;
      CALL the *.shx routine and generate *.shx file;

      free all the allocated memory of each variable;

      END
}
```