Implementation of Weiler Atherton Polygon Clipping using OpenGL

Tarun Kumar
UE143103
Computer Science Engineering
University Institute of Engineering and Technology
Panjab University
Chandigarh, 160030
Email: tarunk.1996@gmail.com

***Abstract*—Building an OpenGL application to draw various 2D polygons. Provide an option to user, which let user to clip two or more polygons(Find intersection, union and difference). FreeGlut library is used in development of this application.**

I. Weiler Atherton Polygon Clipping

A polygon is generally stored as a collection of vertices. Any clipping algorithm takes one collection, and outputs a new collection. A clipped window is also a polygon. In this paper, Weiler Atherton Polygon Clipping is implemented. It is best polygon clipping method for non-rectangular clipping window.

Weiler Atherton Polygon Clipping can be used to find:

- Union
- Intersection

A. Union

Approach used for finding the Union of two polygons:

- Create a list of vertices of polygon and also consisting of intersection points in clockwise direction. (Two types of intersection points are possible either edge is moving into the other polygon or going outwards). Store the intersection point with direction(i.e.inwards or outwards) based on the situation.
- Select Polygon 1 and start from leftmost top vertex of that polygon. Now move along the boundary (storing vertices that are being accessed) of that polygon until any intersection point is not encountered.
- Once intersection point is encountered shift to other polygon and start accessing its vertex in same direction until another intersection point is not encountered.
- Repeat these steps until, it reaches to the vertex it was started earlier.
- Now outline the clipped polygon generated from union of polygon 1 and polygon 2.

B. Intersection

Approach used for finding the Intersection of two polygons:

- Create a list of vertices of polygon and also consisting of intersection points in clockwise direction. (Two types of intersection points are possible either edge is moving into the other polygon or going outwards). Store the intersection point with direction(i.e.inwards or outwards) based on the situation.
- Select Polygon 1 and find first intersection point in it (intersecting point must be going inwards to other polygon). Now start accessing the vertices until another intersection point is not encountered which is going outwards.
- Shift to other polygon and access the vertices in same direction until intersection point is not encountered which is going outwards.
- Repeat these steps until same vertex is not encountered. Look for more intersecting points and process them in same way (In case of multiple intersecting area).

II. About FreeGlut

FreeGLUT is a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) library. GLUT was originally written by Mark Kilgard to support the sample programs in the second edition OpenGL 'RedBook'. Since then, GLUT has been used in a wide variety of practical applications because it is simple, widely available and highly portable.

GLUT (and hence FreeGLUT) takes care of all the system-specific chores required for creating windows, initializing OpenGL contexts, and handling input events, to allow for trully portable OpenGL programs.

FreeGLUT is released under the X-Consortium license.

III. Some FreeGlut Functions

Following are functions used in developing the application with brief intro about it.

- **Creating A Window**
  The glutInitWindowPosition and glutInitWindowSize functions specify a desired position and size for windows that freeglut will create in the system. This glutCreateWindow("Line Drawing Algorithm") will create a Glut Window with a Title Line Drawing Algorithm on it.
- **Setting Background Color**
  This will set the background color of the screen to white.
  ```
  glClearColor(1, 1, 1, 0);
  ```

```
glClear(GL_COLOR_BUFFER_BIT);
glFlush();
```

- Plotting X and Y axis in the window. Set all the pixel along X-axis to black color and similar to that in Y-axis. (Remember according to FreeGlut co-ordinate system (0,0) will lie at the top-left corner of the window, So it should be shifted to the center of the screen first). In order to create X and Y - axis, you can either use inbuilt line drawing function or can set each pixel along that line to color black. Both will gives the same result.
- **Showing Text**
  To display some text at any co-ordinate in the OpenGL window. It can be done by using the following code:

```
drawStrokeText(*string, x, y, z)
{
 char *c;
 glPushMatrix();
 glTranslatef(x, y+8,z);
 glScalef(0.09f,0.08f,z);
 for (c=string; *c != '\0'; c++)
 {
  glutStrokeCharacter(GLUT_STROKE_ROMAN,*c);
 }
 glPopMatrix();
}
```

  In the above function drawStrokeText accepts the string, x, y and z co-ordinates. It will print the string as in character by character format. $GLUT_STROKE_ROMAN$ is the text format that will be displayed on the screen.
- **Creating a Button**
  FreeGlut library as such doesnt include any provision for creating a clickable buttons and associating an action when a button is clicked. So in order to create a button, Create a button look alike box which will perform some action whenever there will a left mouse click on it. Following code will create a box in the glut screen of 80x25 size with a background color of red. GL_QUADS is used to create a quadrilateral in the OpenGL window.

```
glBegin(GL_QUADS);
glColor3f(1,0, 0);
glVertex2i(0,0);
glVertex2i(80,0);
glVertex2i(0,25);
glVertex2i(80,25);
glEnd();
drawStrokeText("Button",5,0,0);
```

  Now, to make this box a fully functional button. Mouse Left click event must be introduced in it.

```
mouse(int btn, int state, int x, int y)
  if(btn == GLUT_LEFT_BUTTON
&& state == GLUT_DOWN)
//Do some event
```

- **OpenGL Color Functions** To set the color display mode glut initialize display mode is used :

```
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB)
```

  Setting the color of a pixel can be done by 3 methods:
  Floating point values

```
glColor3f(0.0,1.0,0.0)
```

  An array specification

```
glColor3fv(colorArray)
```

  Integer Values

```
glColor3i(0,255,0)
```

- **Get properties of certain pixel** To read the seed pixel and extract the properties out of it is can be done by using $glReadPixels()$ function. Syntax of $glReadPixels()$

```
glReadPixels(X,Y,W,H,format,type,data);
```

  Here, X, Y specifies the window coordinates of the first pixel that is read from the frame buffer. W, H specifies the dimensions of the pixel rectangle, width and height of one correspond to a single pixel. $format$ specifies the format of the pixel data. $type$ specifies the data type of the pixel data. $data$ returns the pixel data(can be used to return the color of the seed pixel).