Wireframes, Characters, and Anti-aliasing

CS116A

Chris Pollett

Oct 6, 2004.

Introduction

- OpenGL Wireframe Methods
- OpenGL Front-face function
- Character Attributes
  - General
  - OpenGL
- Anti-aliasing

OpenGL Wireframe Methods

- When one draws a polygon one can choose to display only edges
- To say how to draw polygons one uses:
  - glPolygonMode(face, displayMode);
  - face is one of: GL_FRONT, GL_BACK, GL_FRONT_AND_BACK
  - displayMode is one of: GL_POINT, GL_LINE, GL_FILL

More WireFrame

- Can combine modes by drawing twice:

```
glColor3f(0.0, 1.0, 0.0);
glPolygonMode(GL_FRONT, GL_FILL);
/* Draw polygon */
glColor3f(1.0, 0.0, 0.0);
glPolygonMode(GL_FRONT, GL_LINE);
/* Draw polygon */
```

3D Issues

- If a polygon is in 3-space rather than xy-plane. Doing this may result in gaps around the edges called **stitching**.
- This is caused by slight differences in the scan-line fill and the line drawing algorithm.
- To fix use:

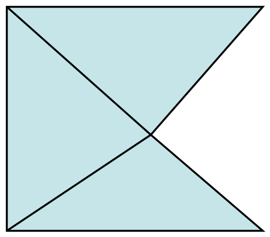
  ```
  glEnable(GL_POLYGON_OFFSET_FILL);
  glPolygonOffset(factor1, factor2);
  ```

  – factor1 is related to steepest z slope of the polygon

  – factor2 is related to constant for this steepest edge
- Typically, need to experiment with these factors but a value between .75 and 1.0 often works.

Issues with Concave Polygons

- Remember we are drawing concave polygons by breaking them up into convex ones.



- In wireframe mode don’t want to see interior lines

More Concave Polygons

- To avoid the previously mentioned issue can use an edge flag:

```
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
glBegin(GL_POLYGON);
    glVertex3fv(v1);
    glEdgeFlag(FALSE);
    glVertex3fv(v2);
    glEdgeFlag(TRUE);
    glVertex3fv(v3);
glEnd();
```

- Can also use an edge flag array with

```
glEnableClientState(GL_EDGE_FLAG_ARRAY);
glEdgeFlagPointer(offset, edgeFlagArray);
```

OpenGL Front-face function

- By default the ordering of the vertices says whether something is a front or back face.
- This can also be set using:

  glFrontFace(vertexOrder);

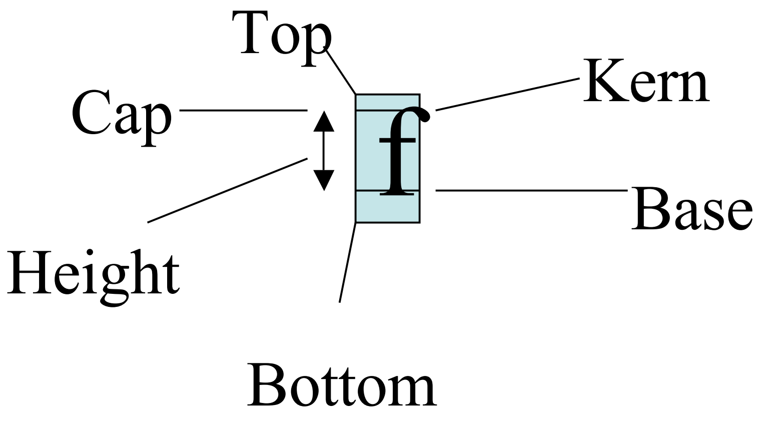
  where vertexOrder is GL_CW (clockwise) or GL_CCW (counterclockwise).

Character Attributes

- Attributes can be set to control the way characters are drawn to the screen.
- These can be used to specify font, <u>underlining style</u>, **boldface**, or *italics*.
- The current color of the system's state is used to draw a character.
- The character's size can be adjusted as well. A common unit for size is a point (1/72 of an inch).

Character dimensions

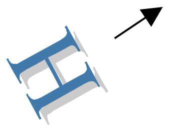
- Width and height are used to specify the character *body*.
- Characters can extend outside this body if they are *kerned*.



More on Characters

- Width and Height are often changed together: H H H.
- Sometimes they are changed independently:

- Sometimes orientation of a character is also changed. So characters points in the direction of some **character up**:



OpenGL Character Attributes

- Already saw can use glutStrokeCharacter(font, char) or glutBitmapCharacter(f,c) to draw characters.
- Can set the font with flags like GLUT_BITMAP_9_BY_15.
- Color is specified by the current state
- glLineWidth can be used to change the width of lines in font.
- glLineStipple can be used to affect if dashed or not.
- Chapter 5 has more effects.

Anti-aliasing

- In line drawing, having to set a pixel entirely on or off. This often causes a step-like appearance to the line.
- This distortion is to due to how often we can sample the "real" line and is called **aliasing**.
- It is known to avoid losing information when sampling from a periodic object need to sample at twice the highest frequency. (Nyquist sampling frequency).
- Supersampling, area sampling and pixel phasing are various techniques for solving the aliasing problem. i.e., to do anti-aliasing.

Supersampling Straight-line Segments

- Divide each pixel on screen into subpixels.
- Draw line as if it had these subpixels.
- Vary the intensity of the actual pixels on the screen according to how many of its subpixels were 1 in the given pixel.
  - Ex: Suppose had 9 subpixels, the line was red, the background blue and 5 subpixels of the pixel were 1. The pixel color would be set to: (5red+4blue)/9
- Supersampling obviously requires more time then usual line drawing.

Subpixel Weighting Masks

- Sometimes supersampling looks better if weigh some subpixels more than others when calculating color intensities:

1	2	1
2	4	2
1	2	1

- Doing this is called using a pixel mask

Area Sampling Straight Line Segments

- If we pretend the line has some finite width, then a line segment has an area and one can calculate exactly how big the intersection is between this area and that of the pixel
- Then we can set the intensity accordingly.
- This is called area sampling.

Filtering Techniques

- One can refine the idea of pixel masks by using a continuous 2D- distribution rather than a fixed mask.
- Examples: Gaussian filter, Cone filters, etc.

Pixel Phasing

- Some raster displays support subpixel positioning
- These systems allow one to draw a pixel than move over a 1/4 pixel and draw a pixel. (pixel phasing)
- This can be used to smooth lines.