Discussion Notes

Week 4

HW1:  • "Cubes-and-Donuts" for letters & numbers

Cubes — offered by glut.

Donuts — Step 1: Draw two faces identically and displace them slightly in the Z direction;

Step 2: Draw a curved surface that connects the two faces;

Both steps can be completed using GL_QUAD_STRIP.

parameters (suggestions):

1. the radius of the inner circle for each face;
2. the displacement in Z between the two faces;
3. how much of a circle do you want to draw (e.g. "0" requires the whole circle, "0.5" requires only half a circle);
4. resolution (how fine is the strip)

• Lighting Problem:

Turning on light will erase the color. But if you add glEnable (GL_COLOR_MATERIAL);
then you can hold the color when light turned on.
• If you’re a little advanced (e.g. you have added lighting, you can change the viewpoint, you’ve made block moving, you’ve had number flying...), it’s fine.

Simple review

**Line Clipping**

• Cohen-Sutherland Line-Clipping Alg.
  Use outcodes (4 bit)
  Rules:
  1. Both end points have outcodes zero \(\rightarrow\) accept
  2. Outcode1 \& Outcode2 nonzero \(\rightarrow\) reject
     \(\times\) Note: This is not a two-direction statement, it’s sufficient, but not necessary!
  3. Neither: Mid-point Alg.
     (Recursive procedure, ensured to end in \(O(\log n)\) steps)

• Cyrus-Beck (Leung-Basky) Line Clipping
  Steps:
  1. Find intersections
     Using parameter equation of lines.
\[ t_e = \frac{N_e (P_e - P_o)}{N_e (P_i - P_o)} \]

Calculate \( t \) for each edge of the clipping window.
Use \( t \in (0, 1) \) rule to reject "false" intersections.

2. Classify intersections
   - PE: potentially entering
   - PL: potentially leaving
   - Locate the largest PE point & \( t > 0 \)
   - Locate the smallest PL point & \( t < 1 \)
   - If \( t_{PE} < t_{PL} \), the line segment between PE and PL is valid.

**Polygon Clipping (Sutherland–Hodgman)**

Given: an ordered sequence of polygon vertices
A convex clipping polygon
Output: ordered clipped polygon vertices

Use **Divide-and-Conquer**, one clipping edge at a time.
The algorithm goes around the polygon from \( V_n \) to \( V_1 \), and then back to \( V_n \); at each step examining the
relationship between successive vertices and the currently considered clip edge.

At each step, 0, 1 or 2 vertices are added to the output list of vertices that defines the clipped polygon.

Example:

![Diagram showing the relationship between successive vertices and the current clip edge with examples of vertices added at different boundaries.](image)