Florida Tech

CSE 5255

INTRODUCTION TO COMPUTER GRAPHICS CLASS NOTES

Dr. William D. Shoaff

Spring 1996

Table of Contents

Clipping Concepts

- Clipping (internal) – removing picture parts outside an area
- External clipping – removing picture parts inside an area
- Cohen–Sutherland algorithm (line clipping)
- Liang–Barsky/Cyrus–Beck algorithm (line clipping)
- Sutherland–Hodgman algorithm (polygon clipping)
- Weiler-Atherton algorithm (polygon clipping)
- Text character clipping

- Scissoring — clip the primitive during scan conversion to pixels
  - Consider a polygon filled on scan line at a time
  - Only the extreme points on the scan line need to be clipped
- Bit (Pixel) block transfers (bitblts/pixblts)
  - Copy a 2D array of pixels from a large canvas to a destination window
  - Useful for text characters, pulldown menus, etc.
- Analytic methods
  - Computer intersections of primitives with the clipping window's boundary
- For floating point graphics packages, clip analytically in floating point coordinate system then scan convert the clipped primitives
- For integer based packages either analytic clipping or scissoring can be used
- Bitblts/Pixblts often used in windowing systems

Point, Line, and Polygon Clipping

- 2-dimensional point clipping
  - A point is saved if

$$x_{min} \leq x \leq x_{max} \quad \text{and} \quad y_{min} \leq y \leq y_{max}$$

  - Very expensive to test each point on a primitive
  - For 3-dimensional point clipping, need to compare z values as well

$$z_{min} \leq z \leq z_{max}$$

- Line clipping
  - Determine if line segment is wholly within the window
  - Determine if line segment is wholly outside window
  - Determine if line segment is partially inside, partially outside
  - Efficient line clipping algorithms will be studied
- Polygon clipping
  - Determine areas of polygon inside window and areas outside window

Cohen–Sutherland Line Clipping

1001	0001	0101
1000	0000 Window	0100
1010	0010	0110

- Clip a line to an upright rectangular window
- Extended window boundaries to define 9 regions: top left, top center, top right, center left, center, center right, bottom left, bottom center, bottom right
- Assign 4 bit code to identify each region
- For each point (x, y)
  - First bit set (1) $\Rightarrow$ point lies to left of window
  - Second bit set (1) $\Rightarrow$ point lies to right of window
  - Third bit set (1) $\Rightarrow$ point lies below window
  - Fourth bit set (1) $\Rightarrow$ point lies above window
- LRBT (Left, Right, Bottom, Top) – somewhat arbitrary sequence
- On the window edge (boundary) $\Rightarrow$ inside window (bit$= 0$)

- Given a line segment with end points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$
- Compute 4-bit codes for each end point
  - If both codes are 0000, (bitwise OR of codes yields 0000) line is totally visible – pass end points to draw routine
  - If both codes have 1's in the same bit position (bitwise AND of codes is <u>not</u> 0000), line is totally invisible
  - Otherwise, indeterminate case – line may be partially visible or not visible
- In the indeterminate case, analytically compute the intersection of the line with the appropriate window edges

- One of two end-points must be outside the window, say it is $P_1 = (x_1, y_1)$
- Read P_1's 4-bit code in order, say left-to-right
- When a set bit (1) is found, compute intersection I of corresponding window edge with line from P_1 to P_2.
  - Suppose we want intersection with the left window edge
  - The x value of the intersection is $x_{\min}$
  - The y value of the intersection is found by substituting $x_{\min}$ into the line equation (from P_1 to P_2)
  $$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$
  and solving for y
  $$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_{\min} - x_1)$$
  - Other cases are handled similarly
- Replace P_1 with I and repeat Cohen–Sutherland algorithm

- Define window by

$$
\begin{aligned}
x &= 0 \quad \text{Left edge} \\
y &= 1 \quad \text{Top edge} \\
y &= 0 \quad \text{Bottom edge} \\
x &= 1 \quad \text{Right edge}
\end{aligned}
$$

- End-points $(1/2,\, 1/2)$ and $(1/4,\, 3/4)$ both have 4-bit codes 0000 – draw line between them
- End-point $(3,\, 3)$ has 4-bit code 0101 and end-point $(-2,\, 5)$ has 4-bit code 1001.
  Logical AND: $0101 \wedge 1001 = 0001 \neq 0000 \Rightarrow$ both end-points to right of window. Therefore line is invisible
- End-point $(3,\, 3)$ has 4-bit code 0101 and end-point $(0,\, 1/2)$ has 4-bit code 0000.
  - $(3,\, 3)$ is outside
  - Reading 4-bit code from left-to-right, "top" bit is set
  - Intersection with top edge is at $I =$?
  - I has 4-bit code ?

Cohen–Sutherland Midpoint Subdivision

- In the indeterminate case, compute the midpoint of the line segment
- Midpoint calculation

$$
\begin{aligned}
x_m &= (x_1 + x_2)/2 \\
y_m &= (y_1 + y_2)/2
\end{aligned}
$$

- Process each half of line segment
- Continue until intersection point is found or single pixel is reached
- Easy to implement in hardware

- Extended 3D-window (NDC cube) boundaries define 27 regions
- Assign 6 bit code to each region
- For each point (x, y, z)
  - Left bit set (1) $\Rightarrow$ point lies to left of window
  - Second bit set (1) $\Rightarrow$ point lies to right of window
  - Third bit set (1) $\Rightarrow$ point lies below window
  - Fourth bit set (1) $\Rightarrow$ point lies above window
  - Fifth bit set (1) $\Rightarrow$ point lies to in front of window (near)
  - Sixth bit set (1) $\Rightarrow$ point lies to behind of window (far)
- LRBTNF (Left, Right, Bottom, Top, Near, Far) – somewhat arbitrary sequence
- On the window edge (boundary) $\Rightarrow$ inside window (bit$= 0$)

- 2D algorithm extends naturally
  - Trivial accept (visible) if both end points have code 000000
  - Trivial reject (invisible) if both end points have code with common bit set
  - Indeterminant case, use parametric form of the line

$$\begin{aligned} x &= x_1 + t(x_2 - x_1) \\ y &= y_1 + t(y_2 - y_1) \\ x &= z_1 + t(z_2 - z_1) \end{aligned}$$

  - To clip against a face, say $y = y_{\max}$, compute

$$t = \frac{y_{\max} - y_1}{y_2 - y_1}$$

    and use it to evaluate the x and z intersections

Liang-Barsky/Cyrus-Beck Line Clipping

- The Liang-Barsky and Cyrus-Beck algorithms use the same ideas, but Liang-Barsky has be optimized for an upright rectangular clip window
- Use of parametric equations, clip window edge normals, and inner products can improve the efficiency of line clipping
- Let $P(t) = P_1 + t(P_2 - P_1)$, $0 \leq t \leq 1$ denote the parametric equation of the line segment from P_1 to P_2
- Let $\vec{n}$ denote the outward pointing normal of the clip window edge E
- Let P_E be an arbitrary point of edge E

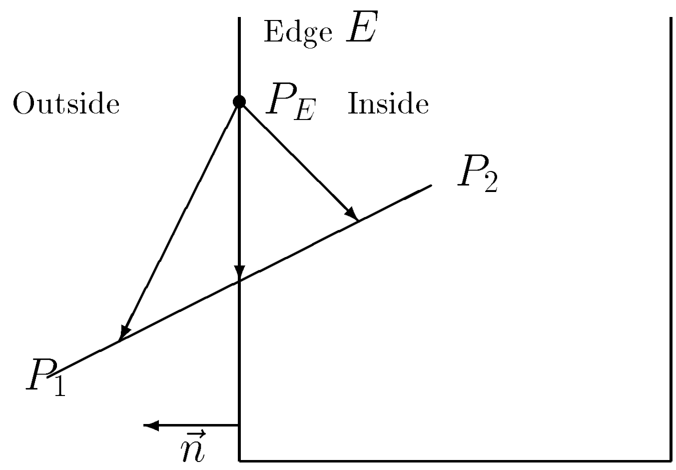
- Consider the vector $P(t) - P_E$ from P_E to a point on the line $P(t)$
- At the intersection of $P(t)$ and E the inner product of $\vec{n}$ and $P(t) - P_E$ is zero

$$\begin{aligned}
\vec{n} \cdot (P(t) - P_E) &= \vec{n} \cdot (P_1 + t(P_2 - P_1) - P_E) \\
&= \vec{n} \cdot (P_1 - P_E) + t\vec{n} \cdot (P_2 - P_1)) \\
&= 0
\end{aligned}$$

- Solving for t yields

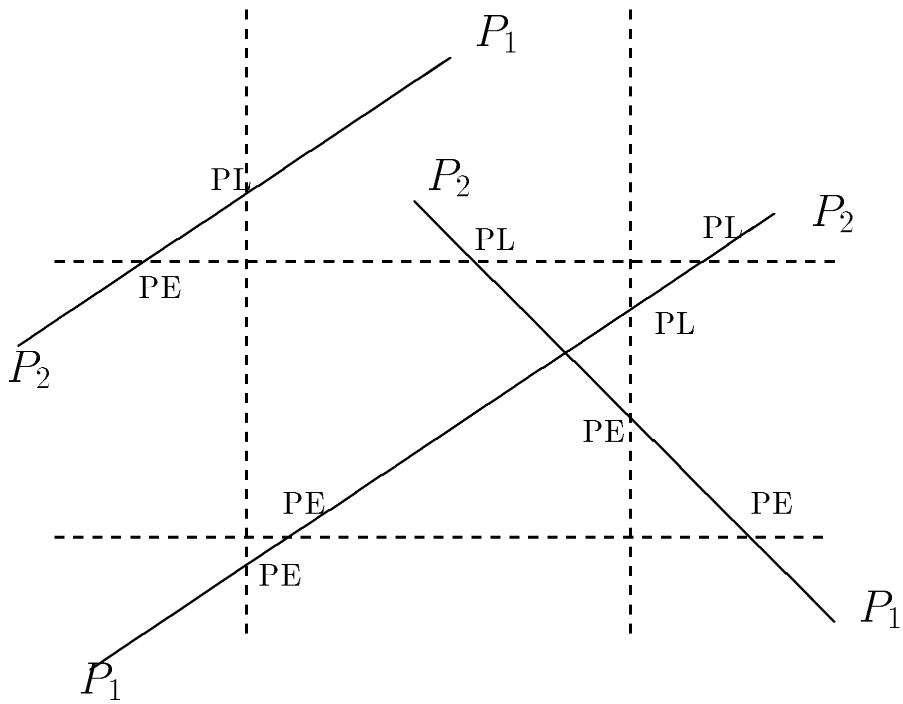
$$t = \frac{\vec{n} \cdot (P_E - P_1)}{\vec{n} \cdot (P_2 - P_1)}$$

- Note that checks need to be made that the denominator above is not zero



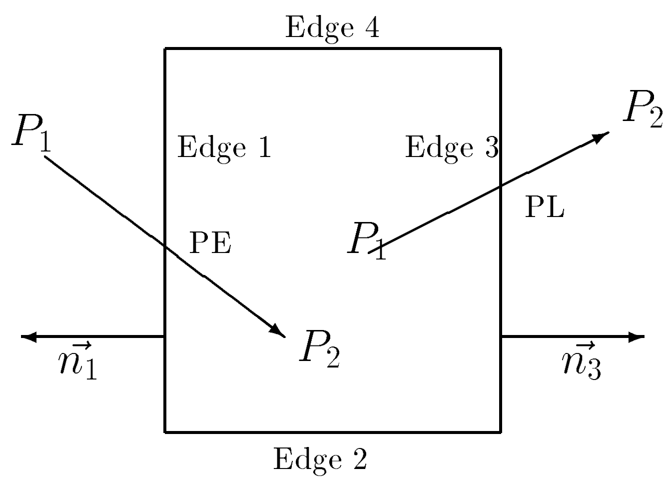
- Using the 4 edge normals for an upright rectangular clip window and 4 points, one on each edge, we can calculate 4 parameter values where $P(t)$ intersects each edge
- Let's call these parameter values t_1, t_2, t_3, t_4
- Note any of the t's outside of the interval $[0, 1]$ can be discarded
- The remaining t values are characterized as "potentially entering" (PE) or "potentially leaving" (PL)
  - The parameter t_i is PE if when traveling along the (extended) line from P_1 to P_2 we move from the outside to the inside of the window with respect to the edge i
  - The parameter t_i is PL if when traveling along the (extended) line from P_1 to P_2 we move from the inside to the outside of the window with respect to the edge i

Liang-Barsky Line Clipping, continued



Liang-Barsky Line Clipping, continued

- The inner product of the outward pointing edge normal $\vec{n}_i$ with $P_2 - P_1$ can be used to classify the parameter t_i as PE or PL
- If
$$\vec{n}_i \cdot (P_2 - P_1) < 0$$
the parameter t_i is potentially entering (PE)
- If
$$\vec{n}_i \cdot (P_2 - P_1) > 0$$
the parameter t_i is potentially leaving (PL)
- Let t_E be the largest PE parameter value and t_L the smallest PL parameter value
- The clipped line extends from t_E to t_L, where $0 \leq t_E \leq t_L \leq 1$



- Let (x, y, z, w) denote a point in homogeneous space and let (X, Y, Z) represent the same point in 3-space
- The clipping plane boundaries $X = 0$, $X = 1$, $Y = 0$, $Y = 1$,$Z = 0$, $Z = 1$ correspond to homogeneous boundary conditions $x = 0$, $w - x = 0$, $y = 0$, $w - y = 0$,$z = 0$, $w - z = 0$
- We can represent these homogeneous boundary conditions as dot products of (x, y, z, w) with column boundary vectors

$$B_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad B_2 = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad B_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$B_4 = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} \quad B_5 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad B_6 = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix}$$

being equal to zero

- Let $P = (x, y, z, w)$ be a point and notice if the dot product

$$P \cdot B_j$$

is less than, equal to, or greater than zero, the point is outside, on, or inside the boundary, respectively

- Consider a line

$$P(u) = P_0 + u(P_1 - P_0)$$

from $P_0 = (x_0, y_0, z_0, w_0)$ to $P_1 = (x_1, y_1, z_1, w_1)$ that crosses a boundary (dot products of P_0 and P_1 with the corresponding boundary vector B_j have opposite signs)

- We find the parameter value u where the line crosses by taking the dot product of the line with B_j

$$P(u) \cdot B_j = P_0 \cdot B_j + u(P_1 \cdot B_j - P_0 \cdot B_j)$$

- Since we want $P(u)$ to be on the boundary, its dot product is zero, so

$$u = \frac{P_0 \cdot B_j}{(P_0 \cdot B_j - P_1 \cdot B_j)}$$

- Given a list $(Q_0, Q_1, \ldots, Q_{n-1})$ of polygon vertices, we clip and move to the first point, then clip and draw the remaining points back to Q_0

  $clip(Q_0,\ move)$;

  for $i = 1$ **to** $n-1$ **do**

   $clip(Q_i,\ draw)$;

  $clip(Q_0,\ draw)$;

- The $clip$ routine calculates outcodes for the boundary planes based on the dot product values

  $kalc\text{-}kodes(bc_1,\ kode_1)$;

  $\{$

   $kode_1 = 0$;

   for $j = 1$ **to** 6 **do**

    $bc_1[j] = P_1 \cdot B_j$;

    if $bc_1[j] < 0$ **then**

     $kode_1$ **and** 1

    $shift\ kode_1$ left one bit

  $\}$

- The *clip* routine is called with a point and a *flag*
- If the *flag* is *move* and the point is visible ($kode = 0$) it is passed down the pipeline as the first point of a line segment
- If the *flag* is *draw* and the point can not be trivially rejected and if both points are in view, the last point is passed down the pipe
- If both points are not in view, the non-trivial clip takes placed
- The values of P_1, bc_1, $kode_1$ are saved for the next call

```
clip(P1, flag);
{
  kalc-kodes(bc1, kode1);
  if flag = move then
    if kode1 = 0 then viewpt(P1, move)
  else
    if (kode0 and kode1) = 0
      if (kode0 or kode1) = 0
        viewpt(P1, draw)
      else do nontrivial-stuff();
  P0 = P1; bc0 = bc1; kode0 = kode1;
}
```

nontrivial-stuff()
{
  $klip = kode_0$ **or** $kode_1$;
  $u_0 = 0$; $u_1 = 1$; $\mathsf{mask} = 1$;
  for $i = 1$ **to** 6 **do**
    if $klip$ **and** $mask \neq 0$ **then**
      $u = bc_0[i]/(bc_0[i] - bc_1[i])$;
      if $kode_0$ **and** $mask \neq 0$ **then**
        $u_0 = \max(u_0, u)$;
      else
        $u_1 = \min(u_1, u)$;
      if $(u_1 < u_0)$ **return**;
    shift mask left one bit;
  if $(\mathbf{kode}_0 \neq 0)$ **then**
    $P = P_0 + u_0(P_1 - P_0)$;
    $viewpt(P,\ move)$
  if $(\mathbf{kode}_1 \neq 0)$ **then**
    $P = P_0 + u_1(P_1 - P_0)$;
    $viewpt(P,\ draw)$
  else $viewpt(P_,\ draw)$
}

Sutherland–Hodgman Polygon Clipping

- Since polygons are basic primitives, algorithms have been developed for clipping them directly
- Given a *subject* polygon (to be clipped) with an ordered sequence of vertices

$$v_1,\ v_2,\ \ldots,\ v_n$$

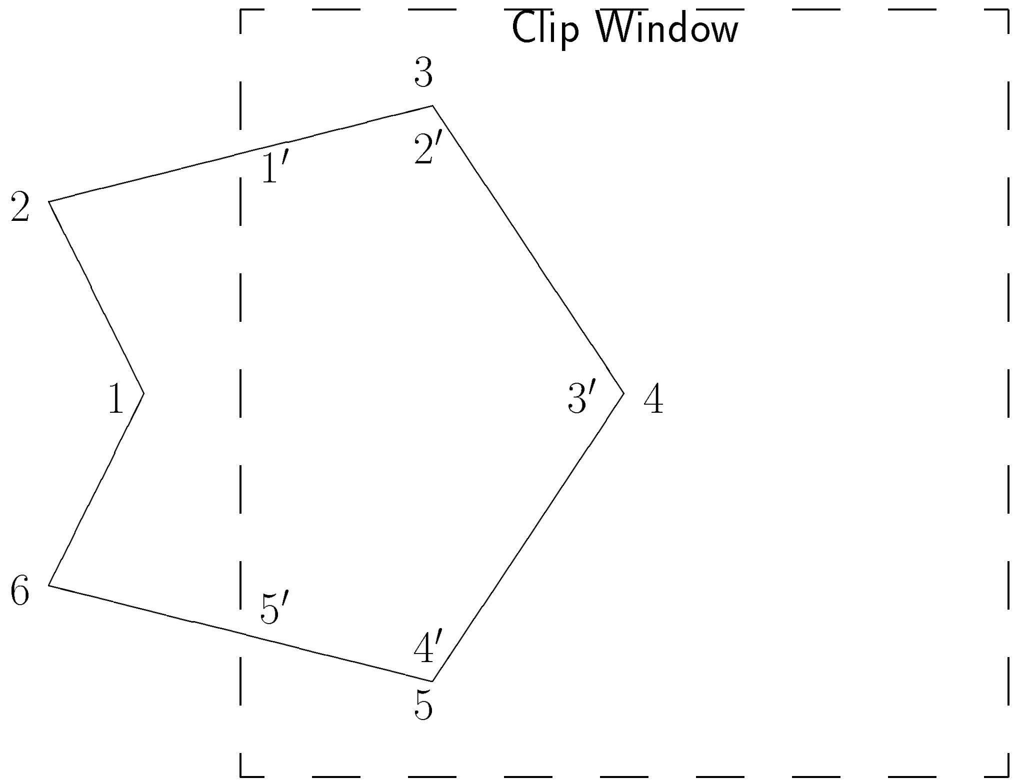
- Compare each subject polygon edge against a single clip window edge
- Save vertices on the in-side of the edge
- Save the intersection points when crossing the edge

- Four cases:
  - Polygon edge goes from outside window edge to outside window edge – save nothing
  - Polygon edge goes from outside window edge to inside window edge – save intersection and inside point
  - Polygon edge goes from inside window edge to outside window edge – save intersection point
  - Polygon edge goes from inside window edge to inside window edge – save second inside point (first was saved previously)
  - Start point – if inside save it, drop it otherwise
  - Last edge – if start point was save, don't save again

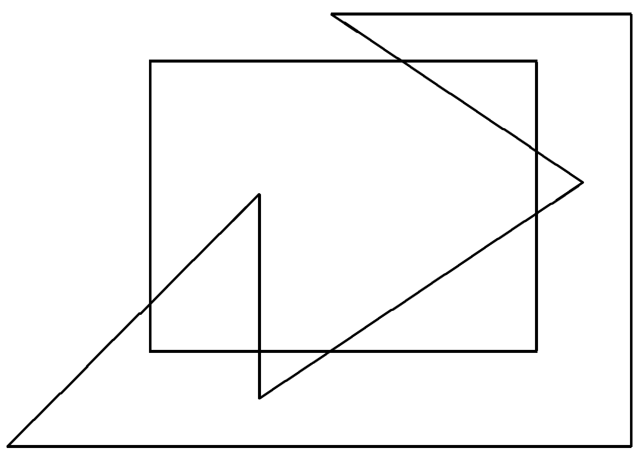
Sutherland–Hodgman Example

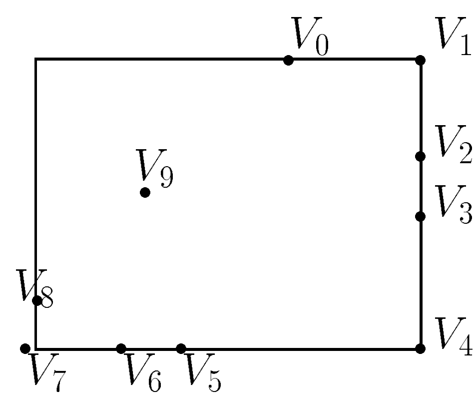
- Clip the polygon against each edge
- Diagram shows result of clipping against the left edge
- Sutherland and Hodgman structure the algorithm so it is reentrant
  - As soon as a vertex is output, clipper calls itself with that vertex
  - Clip is then performed against the next boundary edge
  - No intermediate storage required for partially clipped polygon
  - The is the basis for Clark's "Geometry Engine," which was used in the first Silicon Graphics machines

Sutherland–Hodgman Example



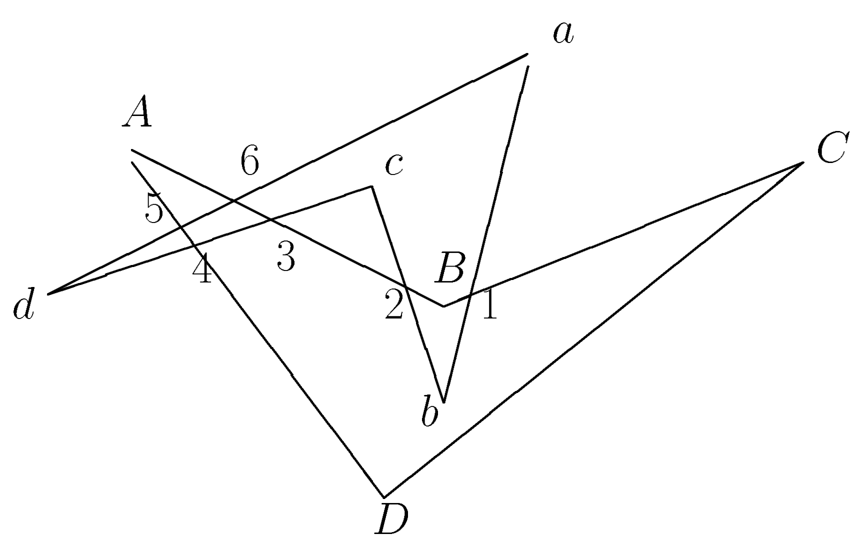
- Sutherland-Hodgman can clip an arbitrary polygon against any *convex* polygonal window (not just rectangles)
- Sutherland-Hodgman may produce connecting lines that were not in the original polygon
- Weiler-Atherton can clip an arbitrary polygon against an arbitrary clipping window and does not produce connecting lines
- Weiler-Atherton is a 2D clip algorithm





- Assume the vertices of the subject polygon are listed in clockwise order (interior is on the right)
- Start at an entering intersection
- Follow the edge(s) of the polygon being clipped until an exiting intersection is encountered
- Turn right at the exiting intersection and following clip window edge until intersection is found
- Turn right and follow the subject polygon
- Continue until vertex already visited is reached
- If entire polygon has not be processed, repeat

- Consider the subject polygon with vertices a, b, c, d and the clip polygon with vertices A, B, C, D
- Insert the intersections in both vertex lists
  - Subject list: a, 1, b, 2, c, 3, 4, d, 5, 6
  - Clip list: A, 6, 3, 2, B, 1, C, D, 4, 5



- Starting at vertex a of the clip polygon, find 1 is first entering intersection
- Traversing the subject, find 2 is exiting intersection
- "Jump" to vertex 2 in clip polygon, follow until vertex 1 (which has be visited)
- Output clipped list 1, b, 2, B

- Jump back to subject list, restarting at c, find 3 is entering intersection
- Traversing the subject, find 4 is exiting intersection
- Jump to vertex 4 in clip polygon, follow until vertex 5 (which is entering)
- Jump to subject, at vertex 5, find 6 is exiting
- Jump to clip, at vertex 6, find 3 is visited
- Output clipped list $3, 4, 5, 6$
- All entering intersections have been visited

Character Clipping

- Characters can be defined as curved or polygonal outlines
  - Computationally expensive to clip and scan convert each character
  - Fonts are "scalable" and can be manipulated by matrix transforms
- Characters can be defined as small rectangular bitmaps
  - Image stored in an offscreen canvas, called a *font cache*
  - Bitblt used to copy character to the frame buffer
  - Bitmaps do not scale well, multiple bitmaps must be defined for each character (at multiple point sizes)
  - A distinct cache is needed for each font
  - Each character can be clipped to destination rectangle on a pixel-by-pixel basis
  - Some systems clip characters or whole strings on an all-or-nothing basis

Problems

1. Given the points below, determine there outcodes relative to the unit cube $0 \le x \le 1$, $0 \le y \le 1$, $0 \le z \le 1$
   - $(0.5,\ 0.5,\ 0.5)$
   - $(5,\ 5,\ 5)$
   - $(-5,\ 5,\ -5)$
   - $(5,\ -5,\ 0.5)$
   - $(5,\ 5,\ 0.5)$
   - $(-5,\ 5,\ -0.5)$
2. Show how the Cohen-Sutherland algorithm would clip the line between the points below. Assume the unit cube as the clipping volume.
   - $(2,\ 3,\ 5)$ and $(-1,\ -3,\ 0)$.
   - $(0.2,\ 0.3,\ 0.5)$ and $(-1,\ -3,\ 0)$.
3. Given a window defined by the lines
   $$x = -1,\ x = 1,\ y = -1,\ y = 1$$
   Show how the Cohen-Sutherland algorithm clips the following lines. That is, find the 4-bit codes for the end-points of the lines, determine that the line is completely visible, completely invisible, or, in the indeterminate case, show the algorithm determines any visible portions of the line.
   1. $P_1 = (0,\ 0)$, $P_2 = (0.75,\ -0.75)$
   2. $P_1 = (-2,\ 2)$, $P_2 = (-0.5,\ 1.5)$
   3. $P_1 = (-2,\ -2)$, $P_2 = (3,\ 3)$
   4. $P_1 = (1,\ 2)$, $P_2 = (4,\ 1)$
4. What are the four cases in the Sutherland-Hodgman polygon clipping algorithm?
5. Use the Sutherland-Hodgman algorithm to clip the polygon $P = (v_0,\ v_1,\ v_2,\ v_3)$, where
   $$v_0 = (-1,\ -2),\ v_1 = (-2,\ -1),\ v_2 = (-2, 2),\ v_3 = (3,\ 2).$$
   against the triangle T where the interior of T is defined by the inequalities
   $$e_0:\ y \le 0,\ e_1:\ x \le 0, \text{ and } e_2: x + y + 4 \ge 0.$$

6. Explain how the Weiler-Atherton clipping algorithm works.

7. Show how to use the Weiler-Atherton algorithm on the polygon $P = (v_0,\, v_1,\, v_2,\, v_3)$, where

$$v_0 = (-1,\, -2),\; v_1 = (-2,\, -1),\; v_2 = (-2, 2),\; v_3 = (3,\, 2).$$

   against the triangle T where the interior of T is defined by the inequalities

$$e_0 :\; y \leq 0,\; e_1 :\; x \leq 0, \text{ and } e_2 : x + y + 4 \geq 0.$$

8. In this question you are to answer questions about the Sutherland-Hodgman polygon clipping algorithm. In each case below, let S and F be the start and final points of a line that crosses edge E and let I be the intersection point.

   1. If both S and F are inside the polygon with respect to the edge E what points will be output to the new polygon?
   2. If S is inside and F is outside the polygon with respect to the edge E what points will be output to the new polygon?
   3. If both S and F are outside the polygon with respect to the edge E what points will be output to the new polygon?
   4. If S is outside and F is inside the polygon with respect to the edge E what points will be output to the new polygon?