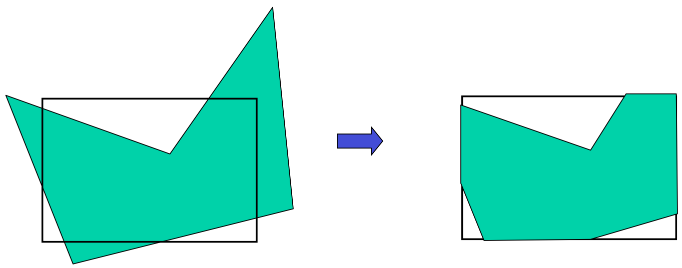
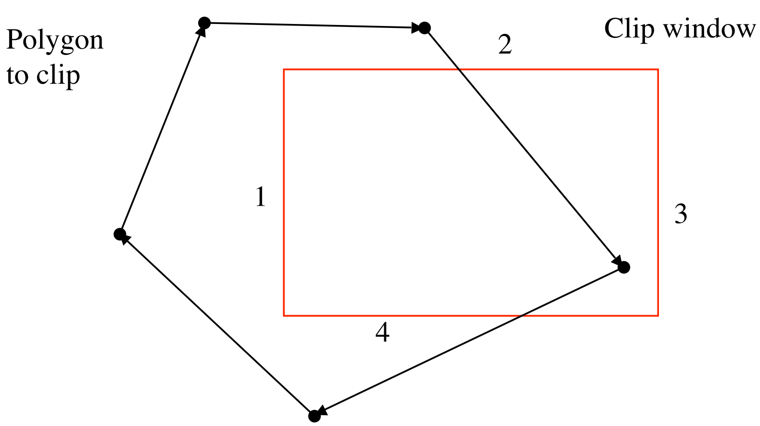
Polygon clip (against convex polygon)



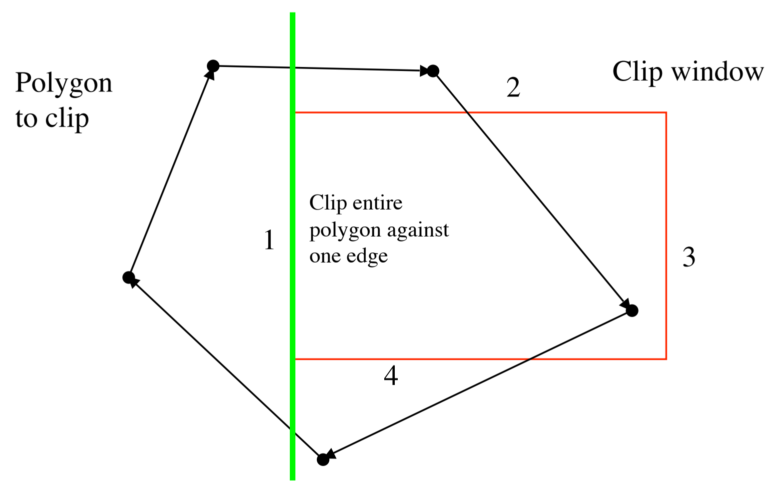
Sutherland-Hodgeman polygon clip

- Recall: polygon is convex if any line joining two points inside the polygon, also lies inside the polygon; implies that a point is inside if it is on the right side of each edge.
- Clipping each edge of a given polygon doesn't make sense - how do we reassemble the pieces? We want to arrange doing so on the fly.
- Clipping the polygon against each edge of the clip window in *sequence* works if the clip window is *convex*.
- (Note similarity to Sutherland-Cohen line clipping)

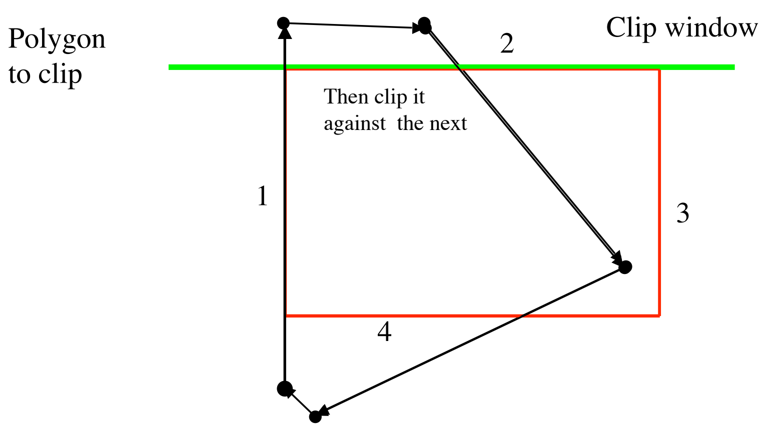
Sutherland-Hodgeman polygon clip



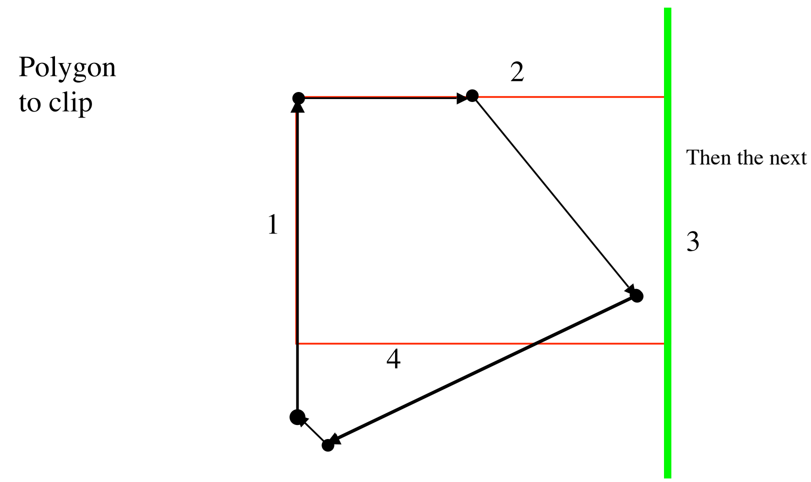
Sutherland-Hodgeman polygon clip



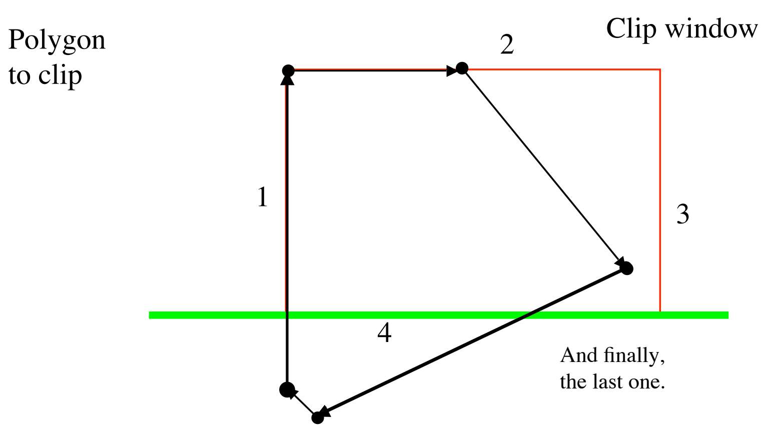
Sutherland-Hodgeman polygon clip



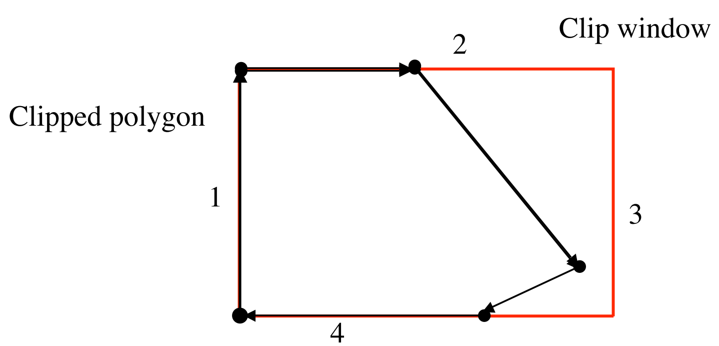
Sutherland-Hodgeman polygon clip



Sutherland-Hodgeman polygon clip



Sutherland-Hodgeman polygon clip



Clipping against **current** clip edge

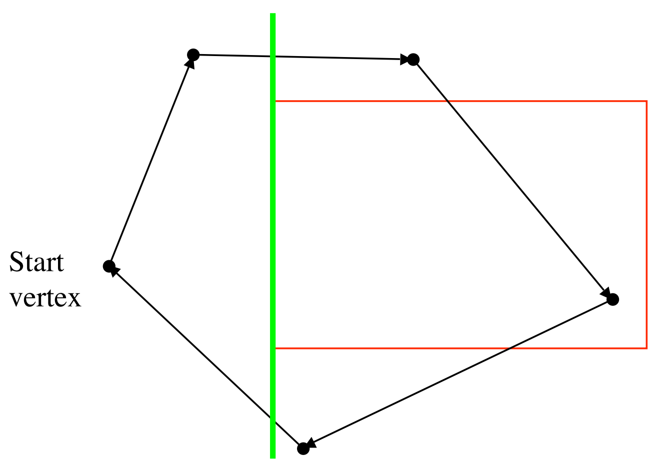
- Polygon is a list of vertices
- Think of process as rewriting polygon, vertex by vertex
- Check start vertex
  - in - emit it
  - out - ignore it
- Walk along vertices and for each edge consider four cases and apply corresponding action.

- Four cases:
  - polygon edge crosses clip edge going from out to in
    -
  - polygon edge crosses clip edge going from in to out
    -
  - polygon edge goes from out to out
    -
  - polygon edge goes from in to in
    -

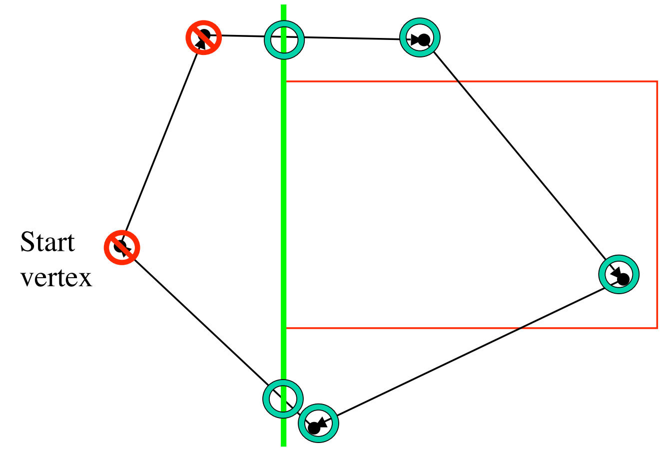
Clipping against current clip edge

- Polygon is a list of vertices
- Think of process as rewriting polygon, vertex by vertex
- Check start vertex
  - in - emit it
  - out - ignore it
- Walk along vertices and for each edge consider four cases and apply corresponding action.

- Four cases:
  - polygon edge crosses clip edge going from out to in
    - emit crossing, next vertex
  - polygon edge crosses clip edge going from in to out
    - emit crossing
  - polygon edge goes from out to out
    - emit nothing
  - polygon edge goes from in to in
    - emit next vertex

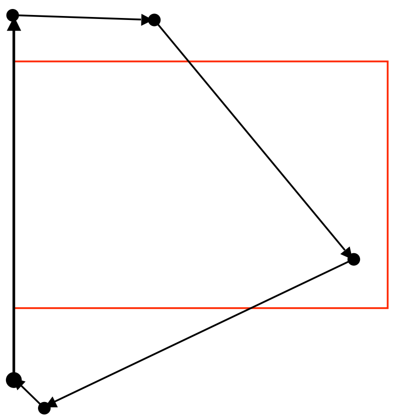


polygon edge crosses clip edge going from out to in ==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out ==> emit crossing
polygon edge goes from out to out ==> emit nothing
polygon edge goes from in to in ==> emit next vertex



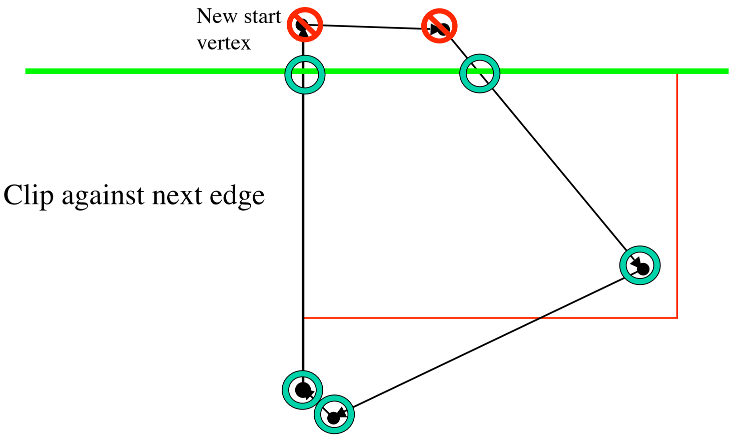
polygon edge crosses clip edge going from out to in ==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out ==> emit crossing
polygon edge goes from out to out ==> emit nothing
polygon edge goes from in to in ==> emit next vertex

Now have



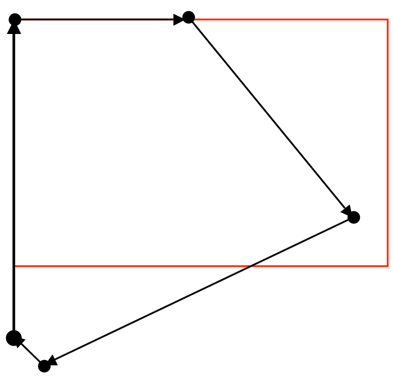
polygon edge crosses clip edge going from out to in ==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out ==> emit crossing
polygon edge goes from out to out ==> emit nothing
polygon edge goes from in to in ==> emit next vertex

Clip against next edge



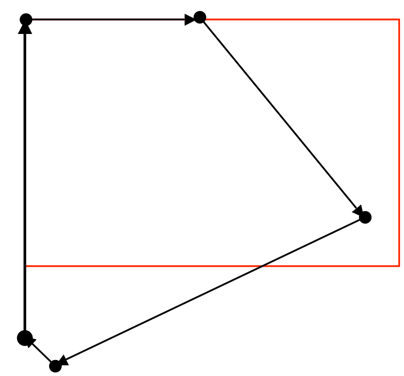
polygon edge crosses clip edge going from out to in ==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out ==> emit crossing
polygon edge goes from out to out ==> emit nothing
polygon edge goes from in to in ==> emit next vertex

Now have



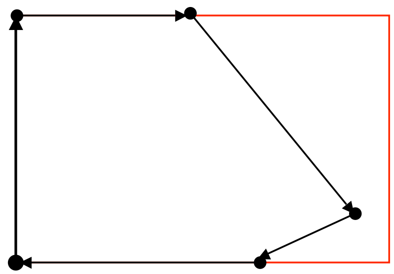
polygon edge crosses clip edge going from out to in ==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out ==> emit crossing
polygon edge goes from out to out ==> emit nothing
polygon edge goes from in to in ==> emit next vertex

Clipping against
next edge (right)
gives



polygon edge crosses clip edge going from out to in ==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out ==> emit crossing
polygon edge goes from out to out ==> emit nothing
polygon edge goes from in to in ==> emit next vertex

Clipping against final(bottom) edge gives



polygon edge crosses clip edge going from out to in ==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out ==> emit crossing
polygon edge goes from out to out ==> emit nothing
polygon edge goes from in to in ==> emit next vertex

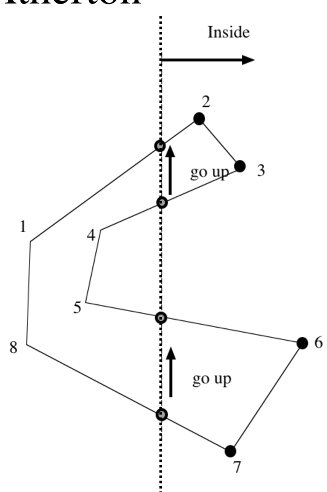
More Polygon clipping

- Notice that we can have a pipeline of clipping processes, one against each edge, each operating on the output of the previous clipper -- substantial advantage.
- Unpleasantness can result from concave polygons; in particular, polygons with empty interior.
- Can modify algorithm for concave polygons (e.g. Weiler Atherton)

Weiler Atherton

For clockwise polygon (starting outside):

- For out-to-in pair, follow usual rule
- For in-to-out pair, follow clip edge (clockwise) and then jump to next vertex (which is on the outside) and start again
- Only get a second piece if polygon is convex



Additional remarks on clipping

- Although everything discussed so far has been in terms of polygons/lines clipped against lines in 2D, all - except Nicholl-Lee-Nicholl - will work in 3D against convex regions without much change.
- This is because the central issue in each algorithm is the inside outside decision as a convex region is the intersection of half spaces.
- Inside-outside decisions can be made for lines in 2D, planes in 3D. e.g testing dx•n>=0
- Hence, all (except N-L-N) can be used to clip:
  - Lines against 3D convex regions (e.g. cubes)
  - Polygons against 3D convex regions (e.g. cubes)
- NLN could work in 3D, but the number of cases increases too much to be practical.

[H&B chapter 5]

2D Transformations

- Represent **linear** transformations by matrices
- To transform a point, represented by a vector, multiply the vector by the appropriate matrix.

[H&B chapter 5]

2D Transformations

- Represent **linear** transformations by matrices
- To transform a point, represented by a vector, multiply the vector by the appropriate matrix.
- Recall the definition of matrix times vector:

$$\begin{pmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- A linear function f(x) satisfies (by definition):

$$f(ax+by) = af(x) + bf(y)$$

- Note that “x” can be an abstract entity (e.g. a vector)–as long as addition and multiplication by a scalar are defined.
- Algebra reveals that matrix multiplication satisfies the above condition

- In particular., if we define f(**x**)=M • **x**, where M is a matrix and **x** is a vector, then

$$\begin{aligned} f(a\mathbf{x}+b\mathbf{y}) &= M(a\mathbf{x}+b\mathbf{y}) \\ &= aM\mathbf{x} + bM\mathbf{y} \\ &= af(\mathbf{x}) + bf(\mathbf{y}) \end{aligned}$$

- Where the middle step can be verified using algebra (next slide)

Proof that matrix multiplication is linear

$$M(a\mathbf{x}+b\mathbf{y}) = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} ax_1 + by_1 \\ ax_2 + by_2 \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}ax_1 + a_{11}by_1 + a_{12}ax_2 + a_{12}by_2 \\ a_{21}ax_1 + a_{21}by_1 + a_{22}ax_2 + a_{22}by_2 \end{pmatrix}$$

$$= a\begin{pmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{pmatrix} + b\begin{pmatrix} a_{11}y_1 + a_{12}y_2 \\ a_{21}y_1 + a_{22}y_2 \end{pmatrix}$$

$$= aM\mathbf{x} + bM\mathbf{y}$$

- Now consider the linear transformation of a point on a line segment connecting two points, **x** and **y**.
- Recall that in parametric form, that point is: $t\mathbf{x} + (1-t)\mathbf{y}$
- The transformed point is: $f(t\mathbf{x} + (1-t)\mathbf{y}) = tf(\mathbf{x}) + (1-t)f(\mathbf{y})$
- Notice that is a point on the line segment from the point $f(\mathbf{x})$ to the point $f(\mathbf{y})$,
- This shows that a linear transformation maps line segments to line segments

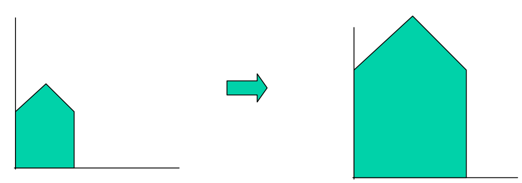
[H&B chapter 5]

2D Transformations of objects

- To transform line segments, transform endpoints
- To transform polygons, transform vertices

2D Transformations

- Scale (stretch) by a factor of k

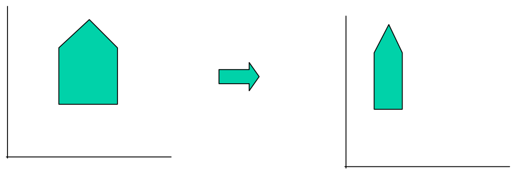


$$M = \begin{vmatrix} k & 0 \\ 0 & k \end{vmatrix}$$

(k = 2 in the example)

2D Transformations

- Scale by a factor of (S_x, S_y)

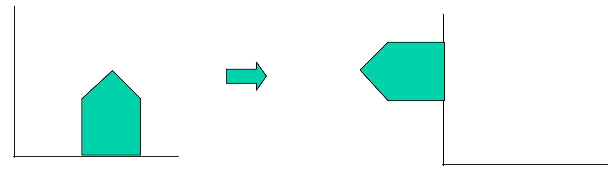


$$M = \begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix}$$

(Above, $S_x = 1/2$, $S_y = 1$)

2D Transformations

- Rotate around origin by θ (Orthogonal)

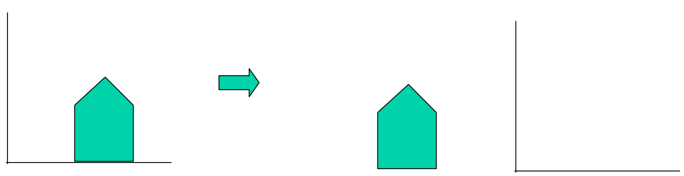


$$M = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$$

(Above, $\theta = 90^o$)

2D Transformations

- Flip over y axis (Orthogonal)

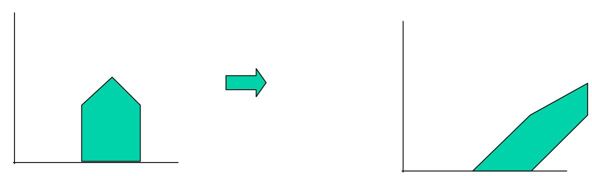


$$M = \begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$$

Flip over x axis is ?

2D Transformations

- Shear along x axis

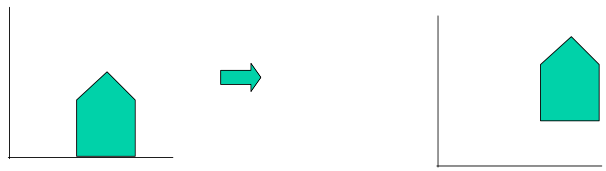


$$M = \begin{vmatrix} 1 & a \\ 0 & 1 \end{vmatrix}$$

Shear along y axis is ?

2D Transformations

- Translation $(\mathbf{P}_{new} = \mathbf{P} + \mathbf{T})$



M =?

Homogenous Coordinates

- Represent 2D points by 3D vectors
- (x,y)-->(x,y,1)
- Now a multitude of 3D points (x,y,W) represent the same 2D point, (x/W, y/W, 1)
- Represent 2D transforms with 3 by 3 matrices
- Can now do translations
- Homogenous coordinates have other uses/advantages (later)

2D Translation in H.C.

$\mathbf{P}_{new} = \mathbf{P} + \mathbf{T}$

$(x', y') = (x, y) + (t_x, t_y)$

$$M = \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix}$$