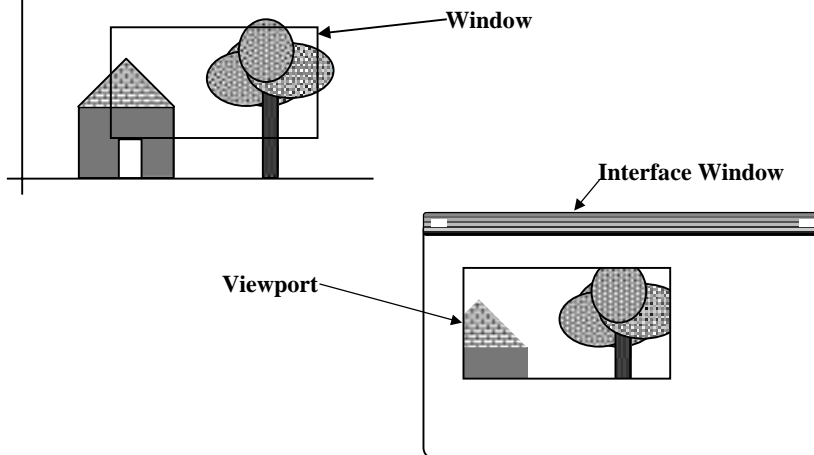


Handouts: Windows, Viewports, Clipping

Windows, Viewports, and Clipping



Windows and Viewports

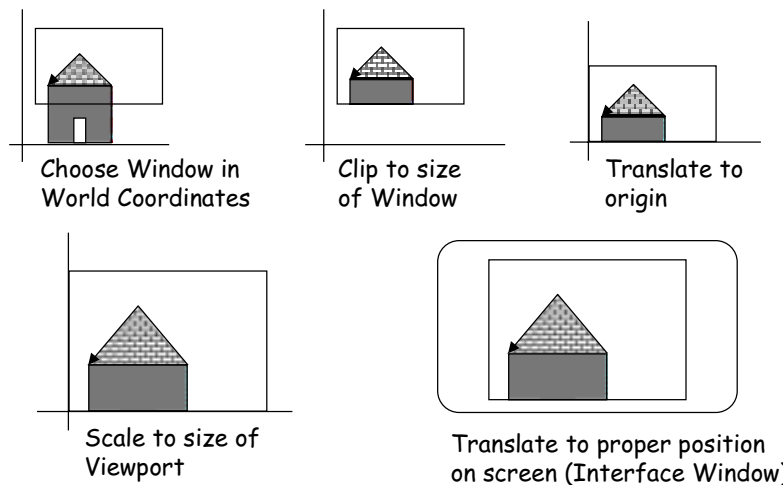


Handouts: Windows, Viewports, Clipping

Terminology

- World Coordinate System (Object Space)
- Screen Coordinate System (Image Space)
- Window (world coordinates)
- Interface Window
- Viewport (screen coordinates)
- Viewing Transformation

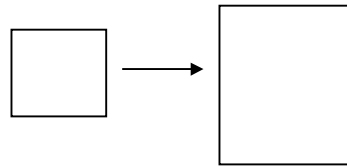
Viewing Transformation



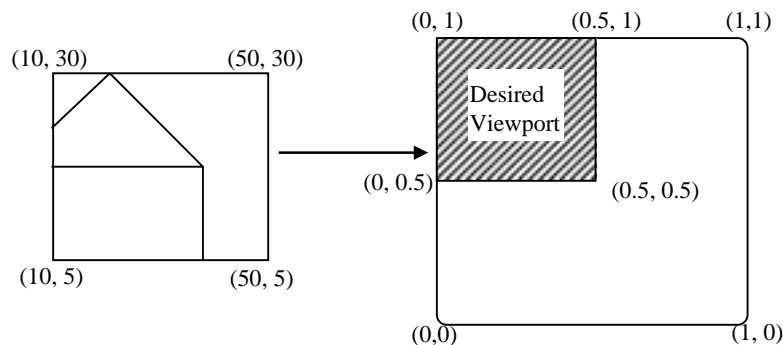
Handouts: Windows, Viewports, Clipping

Notes on Viewing Transformation

- Panning vs. Zooming
- Inverse relationship between window and viewport
 - As the window increases in size, the image in the viewport decreases in size and vice versa
- Beware of aspect ratio



Viewing Transformation Example



Handouts: Windows, Viewports, Clipping

Viewing Transformation Example

$$X \text{ scale} = 0.5/40 = 1/80$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$$

3) Translate to proper coordinates

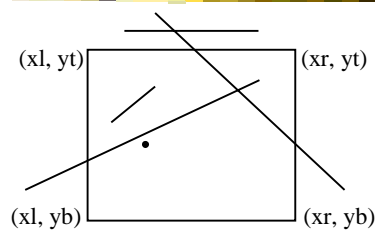
$$\begin{bmatrix} 1/80 & 0 & 0 \\ 0 & 1/50 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2) Scale to correct size

$$\begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

1) Translate window to origin

Clipping



A point is visible if

$$\begin{aligned} &xl < X < xr \\ &\text{and} \\ &yb < Y < yt \end{aligned}$$

- Line visible if both endpoints in window
- "Brute Force Method"
 - Solve simultaneous equations for intersections of lines and edges

Handouts: Windows, Viewports, Clipping

Cohen-Sutherland Algorithm



- Region Checks: Trivially reject or accept lines and points
- Fast for large windows (everything is inside) and for small windows (everything is outside)
- 4-bit outcodes:
 - Bit 1 \leftarrow sign bit of $(yt - Y)$ -- point is above window
 - Bit 2 \leftarrow sign bit of $(Y - yb)$ -- point is below window
 - Bit 3 \leftarrow sign bit of $(xr - X)$ -- point is to right of window
 - Bit 4 \leftarrow sign bit of $(X - xl)$ -- point is to left of window

Cohen-Sutherland Clipping (cont.)



Bit 1: Above
Bit 2: Below
Bit 3: Right
Bit 4: Left

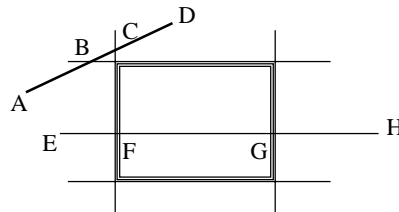
1001	1000	1010
0001	0000	0010
0101	0100	0110

- Trivially accept a line if:
- Trivially reject a line if:

Handouts: Windows, Viewports, Clipping

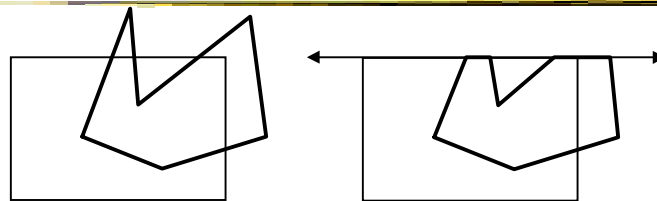
Clipping Lines Not Accepted or Rejected ("divide and conquer")

Example:



- Line AD:
- 1) Test outcodes of A and D --> can't accept or reject.
 - 2) Calculate intersection point B, which is conceptually on the window side of the dividing line. Form new line segment AB and discard the rest of the line because it is above the window.
 - 3) Test outcodes of A and B. Reject.

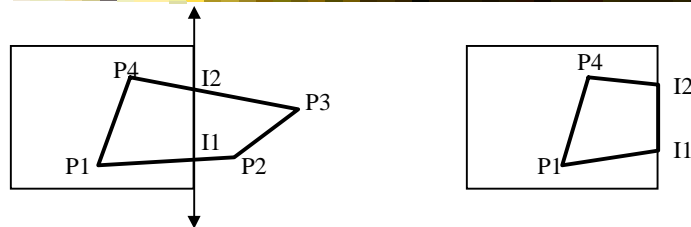
Sutherland-Hodgman Polygon Clipping



- Clip against each edge of the window one edge at a time
- New set of vertices after each clip
 - The number of vertices usually changes and will often increase.

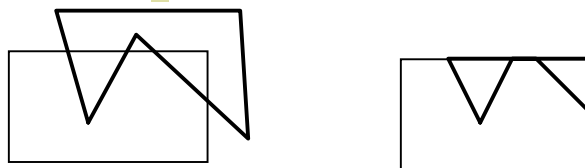
Handouts: Windows, Viewports, Clipping

Polygon Clipping Algorithm



- Window determines a visible and invisible region
- Edge from i to $i+1$ one of four types:
 - Exit visible region - save the intersection
 - Wholly outside visible region - save nothing
 - Enter visible region - save intersection and endpoint
 - Wholly inside visible region - save endpoint

Polygon clipping issues



- Final output, if any, is always considered a single polygon
 - Might be multiple pieces
- Extra edge may not be a problem
 - Always occurs on a window boundary
 - Can be eliminated if necessary

Handouts: Windows, Viewports, Clipping

Pipelined Polygon Clipping



- Clip against each edge independently



- Arrange clipping stages in a pipeline
 - Input polygon clipped against one edge
 - Retained points passed to next stage
- Can avoid intermediate storage