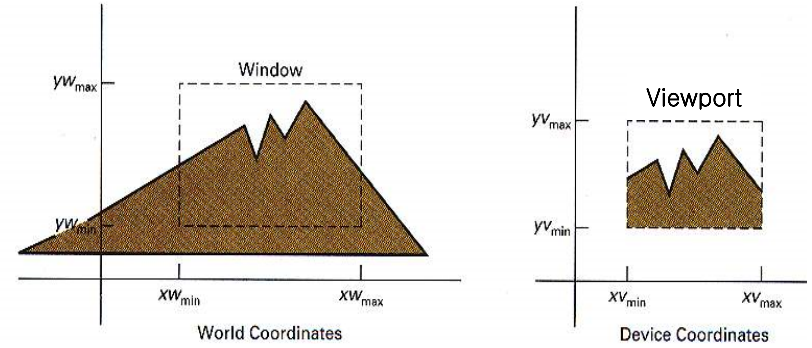
2D Viewing

Fall 2018

Seungyong Lee
POSTECH

Overview

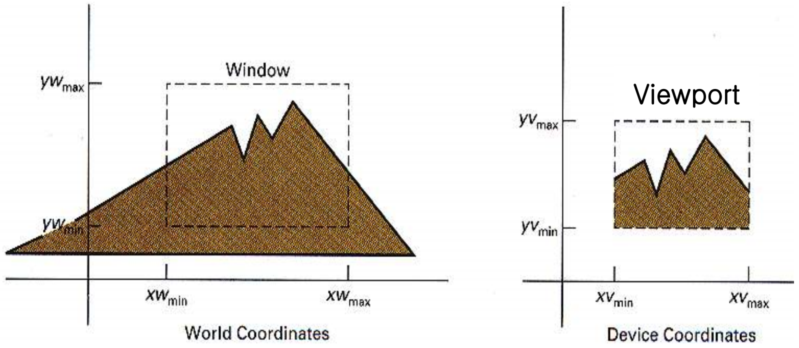
- Coordinate systems
  - world coordinate system
  - device coordinate system



- 2D viewing
  - mapping of a part of a world-coordinate scene to device coordinates
- Window
  - world-coordinate area selected for display
- Viewport
  - area on a display to which a window is mapped

Overview (2)

- Clipping
  - identifies the portions of a scene which are inside of the window
- Chapter summary
  - viewing parameter specification
  - viewing transformation
  - 2D clipping
  - 2D viewing in OpenGL

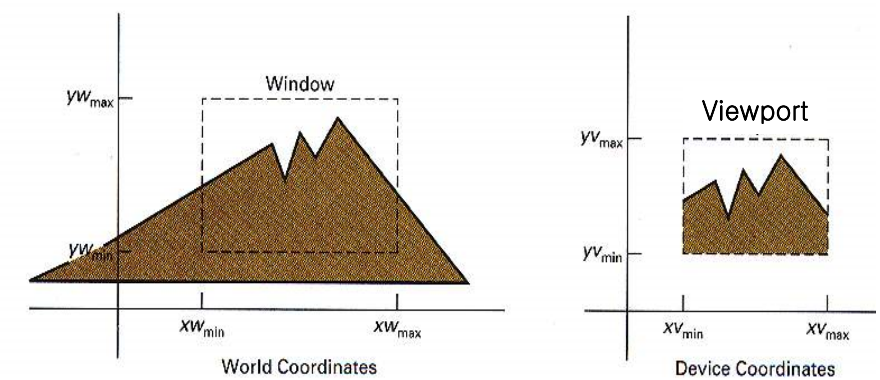


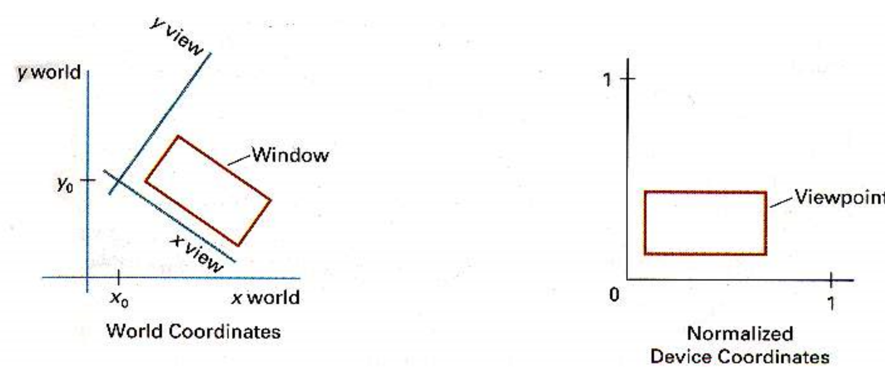
Overview (3)

- Related materials
  - Angel: Section 2.6, Chapter 4, Sections 6.3 - 6.4
  - H&B: Sections 6.1 - 6.3

Viewing Parameter Specification

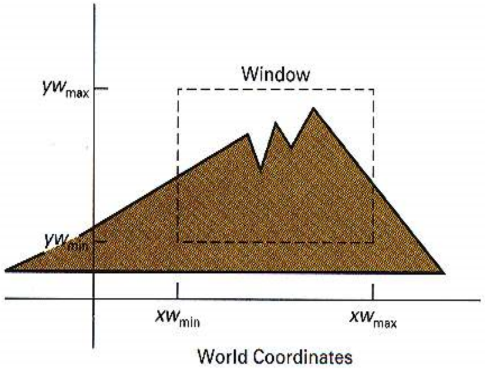
- Viewing coordinate system
  - view reference point
  - view up vector
  - window specification
- Normalized device-coordinate system
  - device-independent graphic system
  - viewport specification

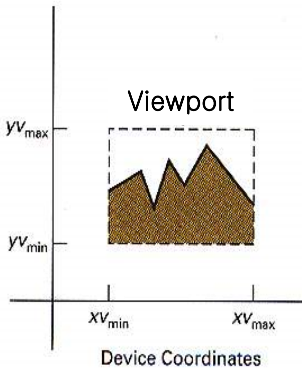


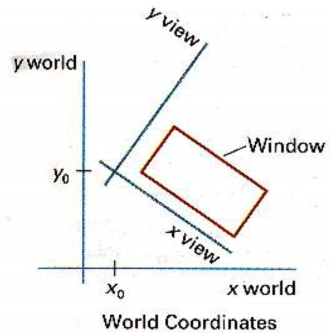


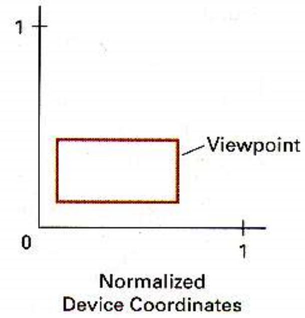
Viewing Parameter Specification (2)

- Viewing operations
  - window and viewport manipulation
  - zoom in/out
  - camera rotation/panning



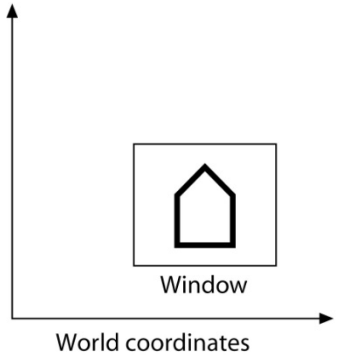


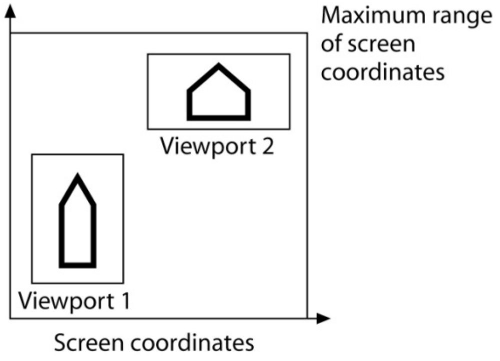


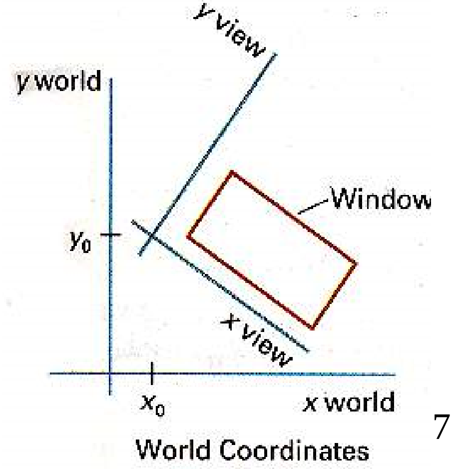


2D Viewing Transformation

- Viewing pipeline
  - break viewing transformation into simple steps
  - clipping
  - WC → VC → NDC → DC
- World-to-view coordinate transformation
  - composition of translation and rotation
  - $M_{VC \leftarrow WC} = RT_1$

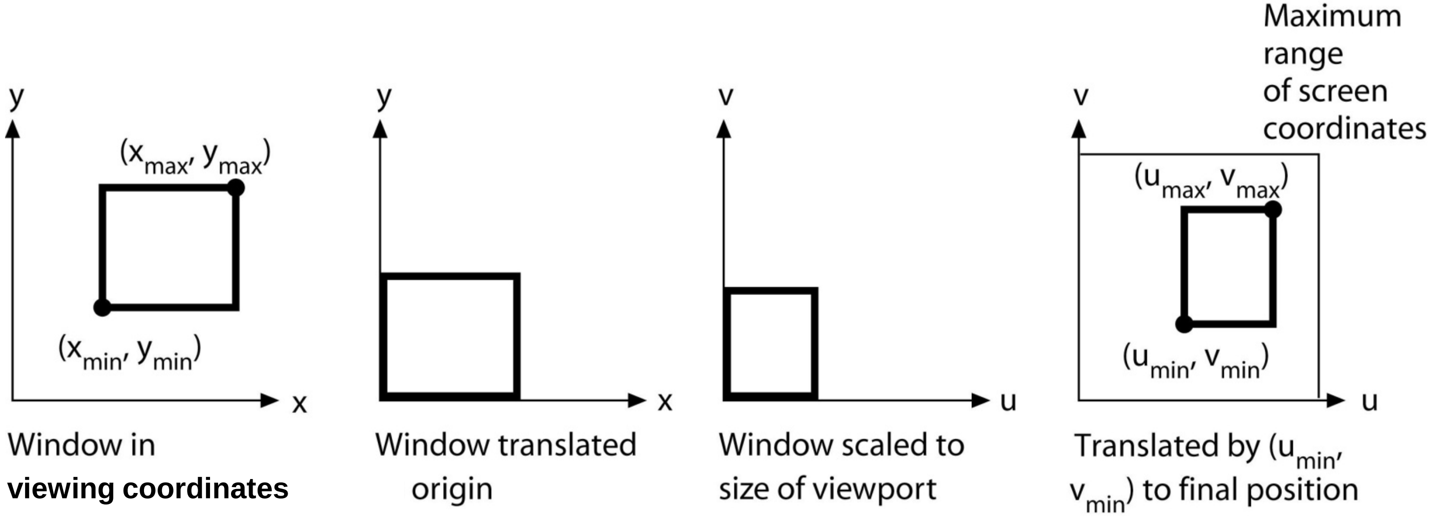






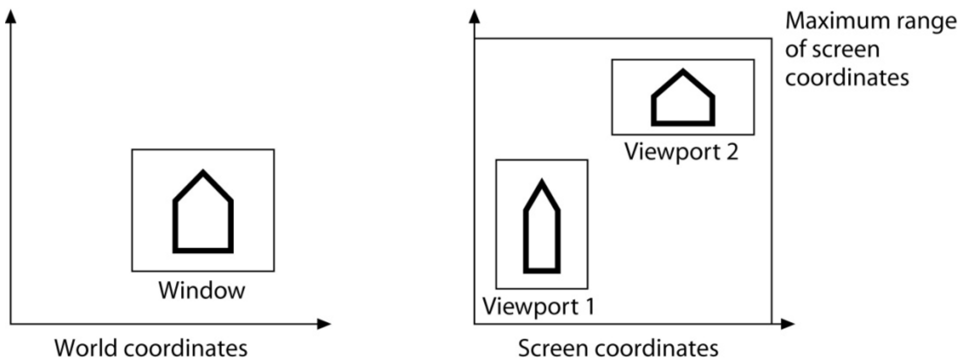
2D Viewing Transformation (2)

- Viewport transformation
  - transform VC to NDC (or DC)
  - $M_{NDC \leftarrow VC} = T_3 S_1 T_2 = T(u_{min}, v_{min}) S\left(\frac{\Delta u}{\Delta x}, \frac{\Delta v}{\Delta y}\right) T(-x_{min}, -y_{min})$
- Device coordinate transformation
  - simple scaling
  - $M_{DC \leftarrow NDC} = S_2$



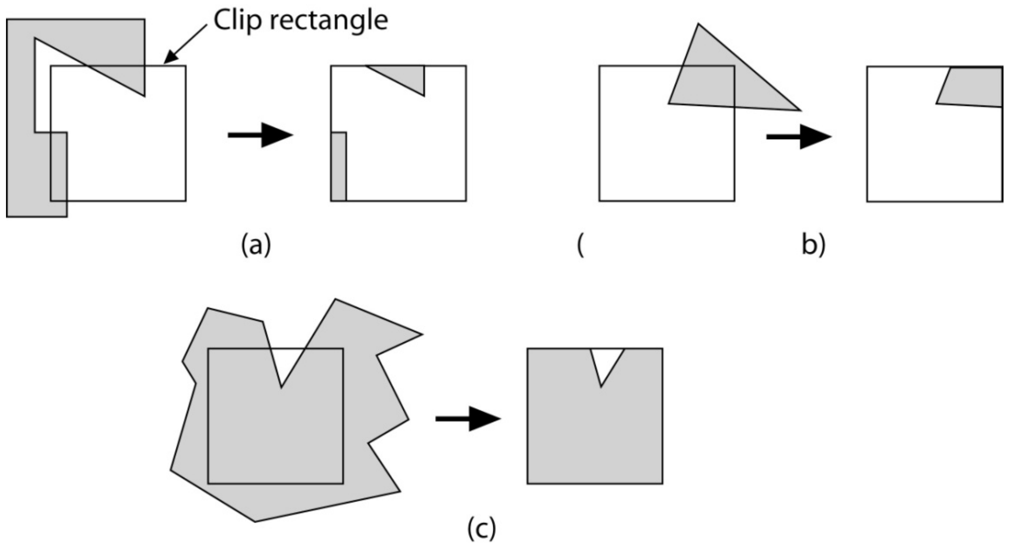
2D Viewing Transformation (3)

- Viewing transformation matrix
  - single 3 × 3 matrix
  - concatenation of matrices from simple steps
  - $M_{DC \leftarrow WC} = S_2 T_3 S_1 T_2 R T_1$



2D Clipping

- What is clipping?
  - identifies those portions of a picture that are either inside or outside of a specified region
- Clip window
  - usually a rectangular region
  - can be a general polygon

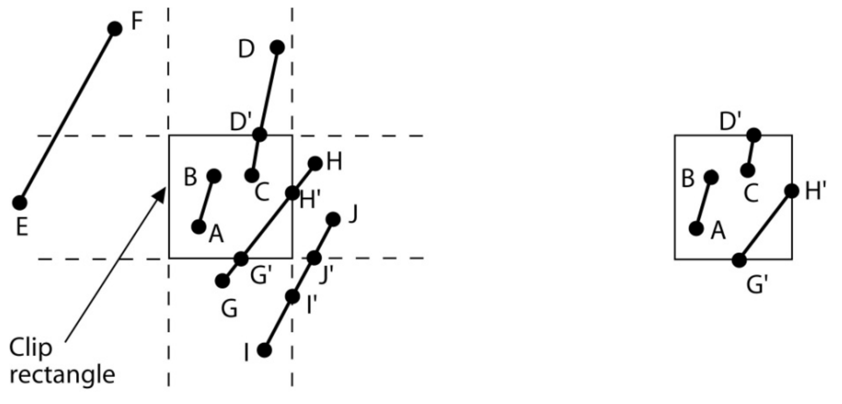


2D Clipping (2)

- Clipping algorithms
  - rectangular clip window
- Point clipping
  - point $P = (x, y)$
  - accepted (saved) if $x_{min} \leq x \leq x_{max}$ and $y_{min} \leq y \leq y_{max}$
  - else rejected (clipped)

2D Line Clipping

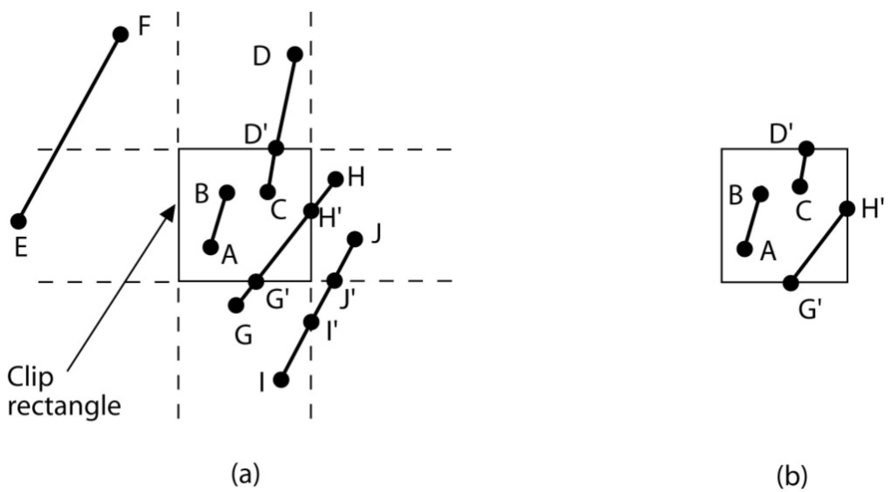
- Clipping endpoints
  - trivially accepted
    - $x_{min} \le x_0, x_1 \le x_{max}$ and $y_{min} \le y_0, y_1 \le y_{max}$
  - trivially rejected
    - $x_0, x_1 \le x_{min}$ or $x_0, x_1 \ge x_{max}$ or $y_0, y_1 \le y_{min}$ or $y_0, y_1 \ge y_{max}$



(a) (b)

2D Line Clipping (2)

- Cohen-Surtherland algorithm
  - region test to avoid intersection calculation
    - trivially reject: code(p0) and code(p1) $\neq$ 0
    - trivially accept: code(p0) or code(p1) = 0
  - if not (trivially reject or accept)
    - divide the line to two lines
    - add the two lines to the line set



2D Line Clipping (3)

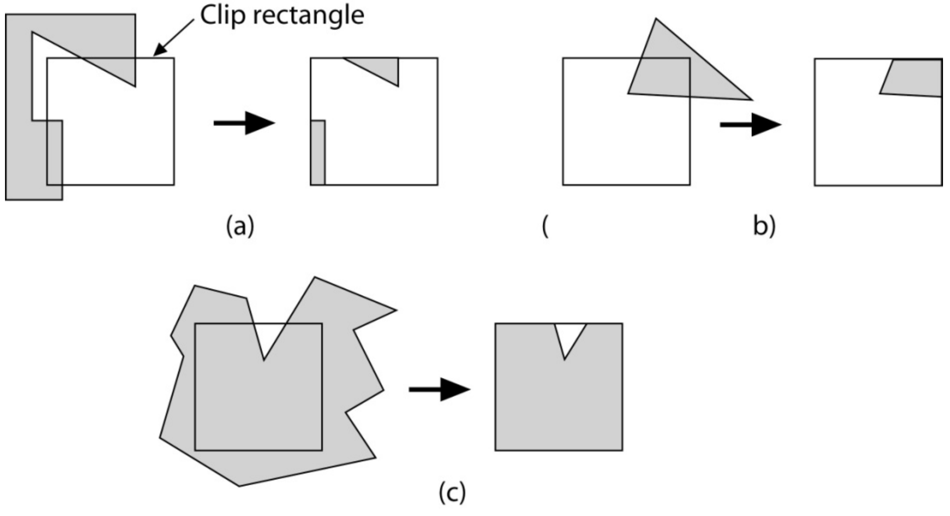
– algorithm

```
lineSet = all input lines;
clippedSet = empty
while (lineSet is not empty) {
   pick and delete a line from lineSet;
   region test
   if (trivially accept)  add the line to clippedSet;
   else if (trivially reject)  continue;
   else  {
         divide the line to two lines;
         add the two lines to lineSet;
   }
}
```

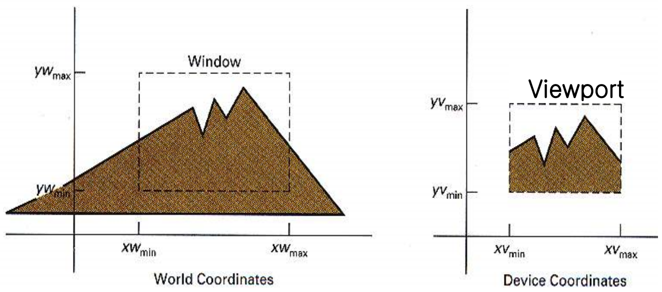
– application?

Clipping Polygons

- Rather complicated
  - each edge of the polygon must be tested against each edge of the clip rectangle
  - new edges may be added
  - existing edges may be discarded, retained, or divided
  - multiple polygons may result from clipping a single polygon



Clipping in Viewing Pipeline



- Rectangular clip window
- Window or viewport clipping?
  - WC → VC → NDC → DC
- Canonical view window
  - normalized view window
  - x = –1, x = 1, y = –1, y = 1
  - for what?
- Process
  - world coordinates → canonical view window
  - 2D clipping
  - canonical view window → viewport

2D Viewing in OpenGL

- Viewing parameter specification
  - gluLookAt(ex, ey, ez, cx, cy, cz, ux, uy, uz);
    - gluLookAt(0, 0, 0, 0, 0, -100, 0, 1, 0);
    - transformation to the viewing coordinates
  - gluOrtho2D(left, right, bottom, top);
    - window specification
    - transformation to the canonical view window
    - clipping is done by hardware in a graphics pipeline
  - glViewport (x, y, width, height);
    - viewport specification
    - transformation to the device coordinates

2D Viewing in OpenGL (2)

- Coordinate systems
  - modeling (local) coordinates
    - drawing objects with output primitives
  - world coordinates
    - modeling transformations for final positions
  - viewing coordinates
    - window specification
  - device coordinates
    - viewport transformation
  - MC → WC → VC → CVC → DC

2D Viewing in OpenGL (3)

- Object drawing
  - set the window with gluOrtho2D
    - in the viewing coordinate system
  - call gluLookAt first
    - before the call, we are in the viewing coordinate system
    - after the call, we are in the world coordinate system
  - draw an object
    - modeling transformations for positioning

2D Viewing in OpenGL (4)

- Two matrix stacks
  - model-view/projection matrix stacks
  - each function constructs a matrix and multiplies it by the current matrix stack
  - glMatrixMode(GL_MODELVIEW/GL_PROJECTION);
  - gluLookAt → model-view matrix stack
  - gluOrtho2D → projection matrix stack
- Vertex transformation process
  - model-view matrix → projection matrix → viewport transformation

2D Viewing in OpenGL (5)

- Model-view matrix stack
  - why gluLookAt in the model-view matrix stack?
  - modeling and viewing transformations are inverse of each other
  - gluLookAt is called before modeling transformations
- Projection matrix stack
  - transformation for camera intrinsic operations
- Why two matrix stacks?
  - will be cleared later

2D Viewing in OpenGL (6)

- Object animation
  - three ways
    - apply transformations to the object
    - change the viewpoint with gluLookAt
    - change the window with gluOrtho2D
  - which way is the best?
- Why gluLookAt for 2D viewing?
  - gluOrtho2D is not enough?

Summary

- 2D viewing
  - world coordinates, device coordinates
  - window, viewport
  - viewing coordinates, normalized device coordinates
- Viewing transformation
  - viewing pipeline
  - coordinate system transformations
- 2D clipping
  - line clipping/polygon clipping
  - canonical view window

Summary (2)

- 2D viewing in OpenGL
  - viewing/world coordinates
  - device coordinates with y-axis from bottom to top
  - viewing transformation functions
  - matrix stacks
  - coordinate system transformation process

Supplementary Slides

2D Line Clipping

- Computing intersection of a line with a clip edge
  - derive t and t' for the intersection point
  - check if $0 \le t, t' \le 1$

$$x = x_0 + t(x_1 - x_0),\ y = y_0 + t(y_1 - y_0),\ 0 \le t \le 1$$

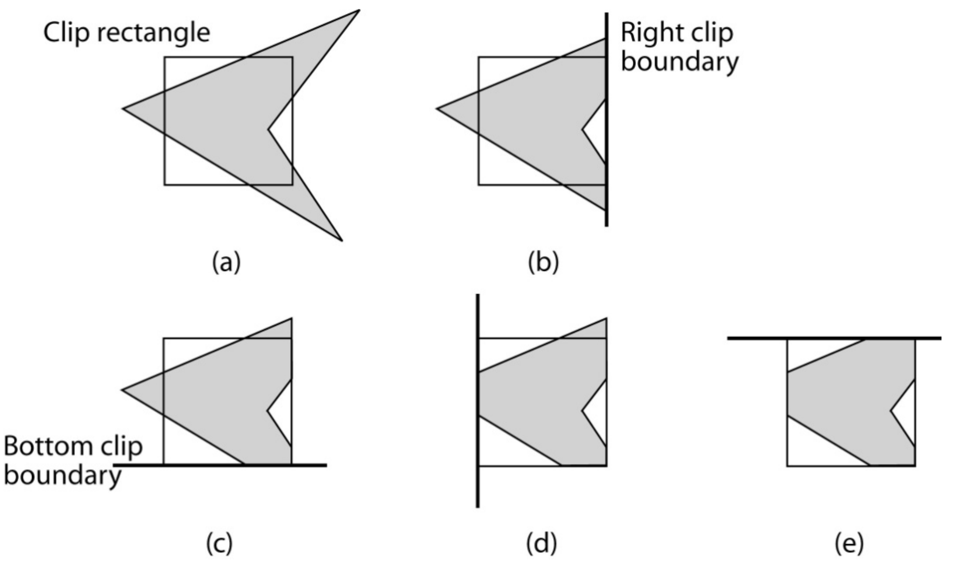
$$x' = x'_0 + t'(x'_1 - x'_0),\ y' = y'_0 + t'(y'_1 - y'_0),\ 0 \le t' \le 1$$

Clipping Polygons (2)

- Bounding box
  - a rectangle including the given polygon
  - trivially accept if the bounding box is included in the clip rectangle
  - trivially reject if bounding box is outside of the clip rectangle
- If not (trivially accept or reject)
  - analytic intersection computation

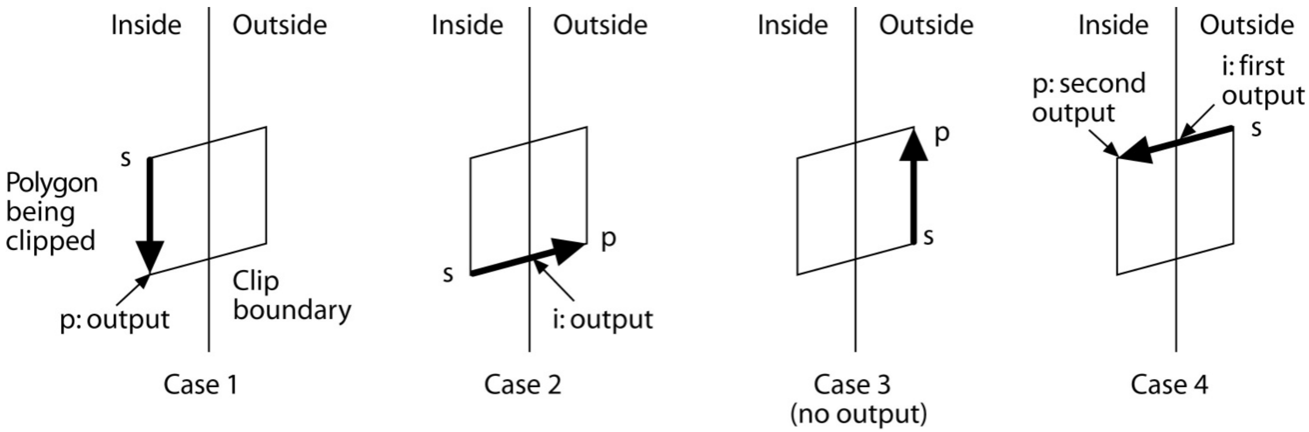
Clipping Polygons (3)

- Sutherland-Hodgman algorithm
  - divide and conquer
    - clip a polygon against a single infinite clip edge
    - repeat this for each edge of the clip rectangle



Clipping Polygons (4)

– visit each vertex in sequence
  - four cases



- Clipping of a triangle and a convex quadrilateral?
  – simpler algorithm?