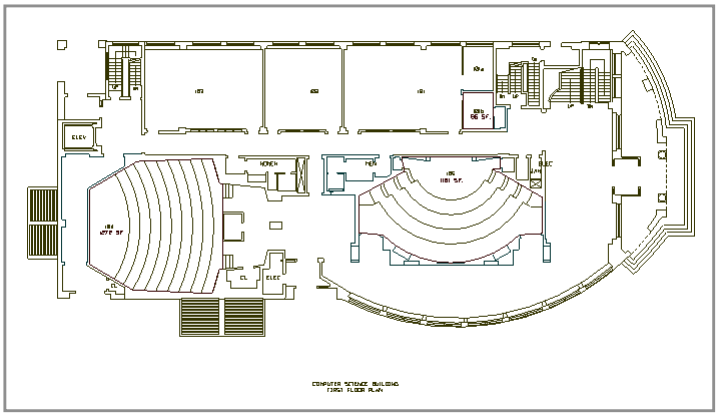
2D Rendering Pipeline

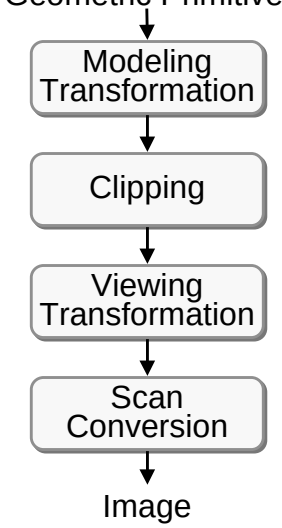
Thomas Funkhouser
Princeton University
C0S 426, Fall 1999

2D Rendering Example

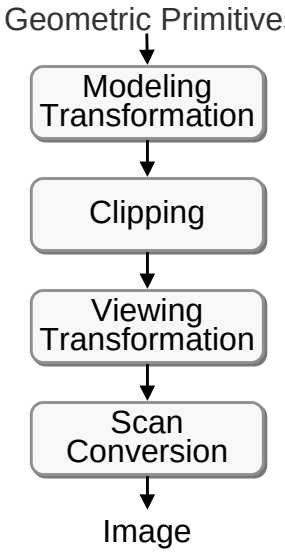


What issues must be addressed by a
2D rendering system?

2D Rendering Pipeline

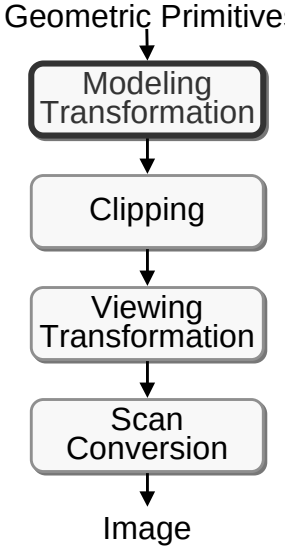


2D Rendering Pipeline



Start with a set of 2D primitives

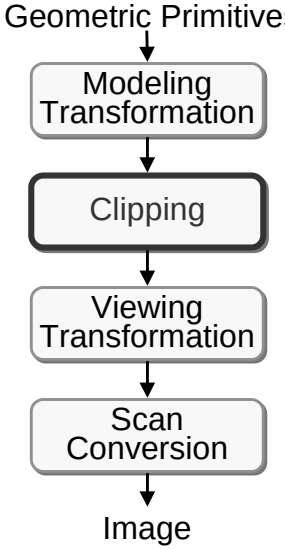
2D Rendering Pipeline



Start with a set of 2D primitives

Transform geometric primitives into world coordinate system

2D Rendering Pipeline

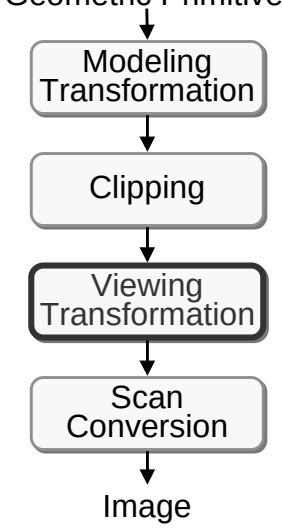


Start with a set of 2D primitives

Transform geometric primitives into world coordinate system

Clip portions of geometric primitives residing outside the window

2D Rendering Pipeline



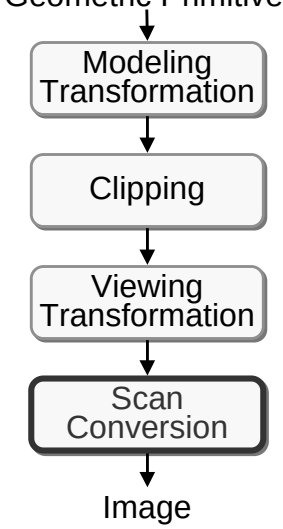
Start with a set of 2D primitives

Transform geometric primitives into world coordinate system

Clip portions of geometric primitives residing outside the window

Transform the clipped primitives from world to screen coordinates

2D Rendering Pipeline



Start with a set of 2D primitives

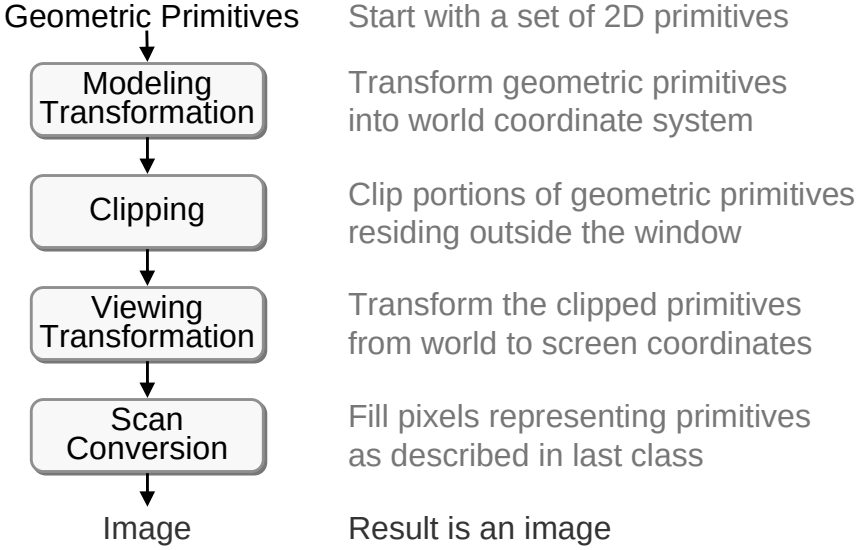
Transform geometric primitives into world coordinate system

Clip portions of geometric primitives residing outside the window

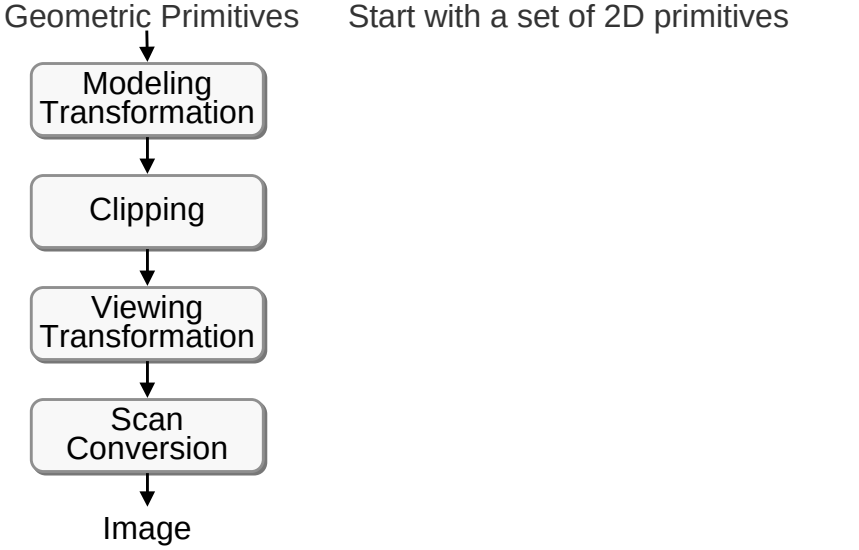
Transform the clipped primitives from world to screen coordinates

Fill pixels representing primitives as described in last class

2D Rendering Pipeline

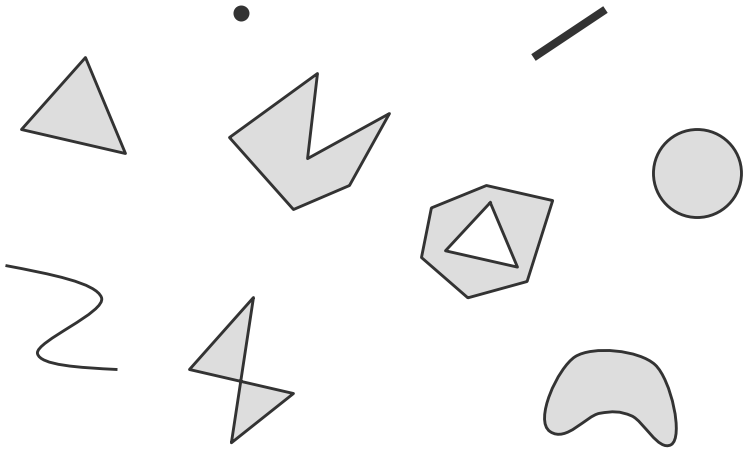


2D Rendering Pipeline



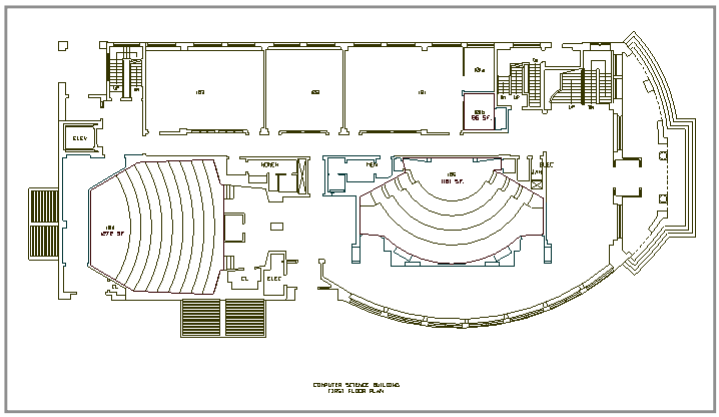
2D Geometric Primitives

- Lines, polygons, circles, splines, etc.



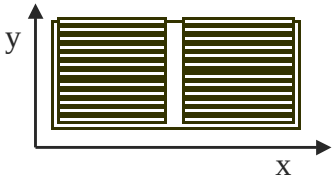
2D Geometric Primitives

- It would be hard to define this scene all in one coordinate system

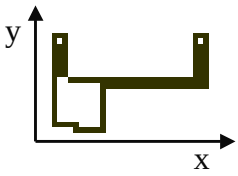


2D Geometric Primitives

- Some primitives may be grouped into “objects” and defined in their own coordinate systems

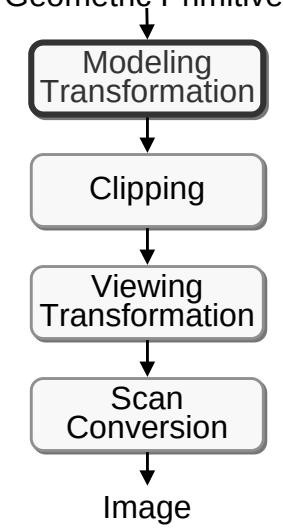


Coordinate System for Object #1



Coordinate System for Object #2

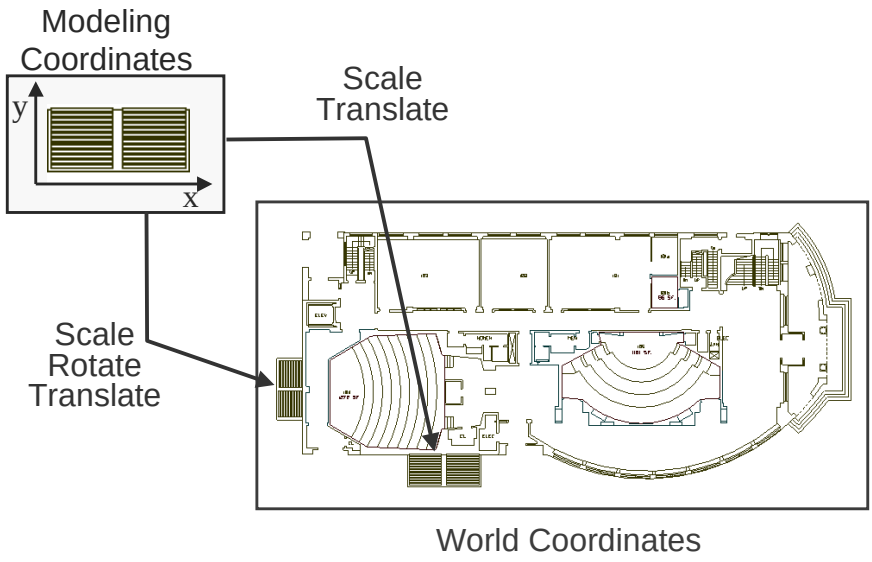
2D Rendering Pipeline



Start with a set of 2D primitives

Transform geometric primitives into world coordinate system

2D Modeling Transformations

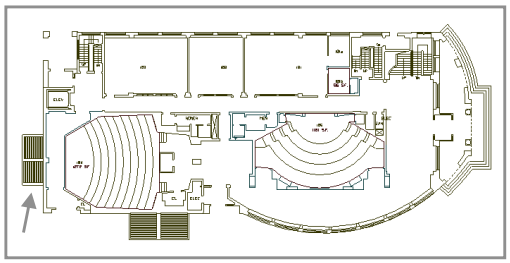


2D Modeling Transformations

- Translation:
  - x' = x + tx
  - y' = y + ty
- Scale:
  - x' = x * sx
  - y' = y * sy
- Rotation:
  - x = x*cosΘ - y*sinΘ
  - y = x*sinΘ + y*cosΘ
- Shear:
  - x' = x + hx*y
  - y' = y + hy*x

2D Modeling Transformations

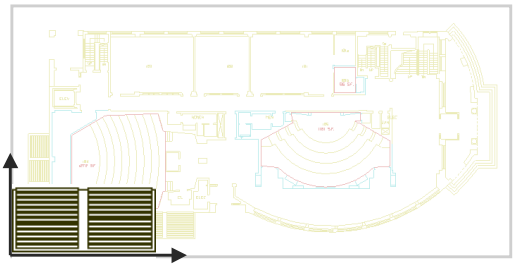
- Translation:
  - x’ = x + tx
  - y’ = y + ty
- Scale:
  - x’ = x * sx
  - y’ = y * sy



- Rotation:
  - x = x*cosΘ - y*sinΘ
  - y = x*sinΘ + y*cosΘ
- Shear:
  - x’ = x + hx*y
  - y’ = y + hy*x

2D Modeling Transformations

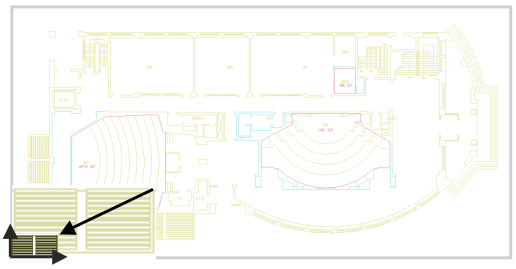
- Translation:
  - x’ = x + tx
  - y’ = y + ty
- Scale:
  - x’ = x * sx
  - y’ = y * sy



- Rotation:
  - x = x*cosΘ - y*sinΘ
  - y = x*sinΘ + y*cosΘ
- Shear:
  - x’ = x + hx*y
  - y’ = y + hy*x

2D Modeling Transformations

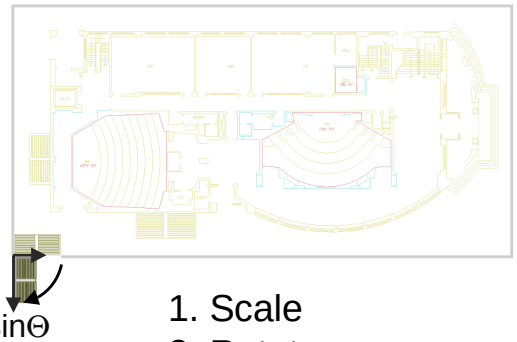
- Translation:
  - x’ = x + tx
  - y’ = y + ty
- Scale:
  - x’ = x * sx
  - y’ = y * sy
- Rotation:
  - x = x*cosΘ - y*sinΘ
  - y = x*sinΘ + y*cosΘ
- Shear:
  - x’ = x + hx*y
  - y’ = y + hy*x



1. Scale

2D Modeling Transformations

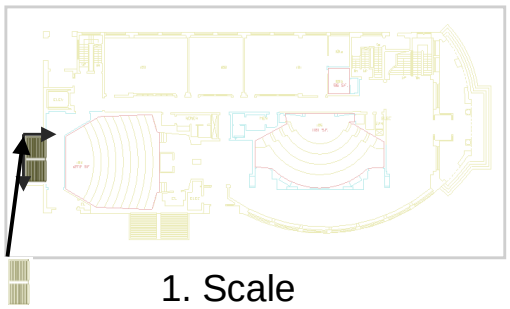
- Translation:
  - x’ = x + tx
  - y’ = y + ty
- Scale:
  - x’ = x * sx
  - y’ = y * sy
- Rotation:
  - x = x*cosΘ - y*sinΘ
  - y = x*sinΘ + y*cosΘ
- Shear:
  - x’ = x + hx*y
  - y’ = y + hy*x



1. Scale
2. Rotate

2D Modeling Transformations

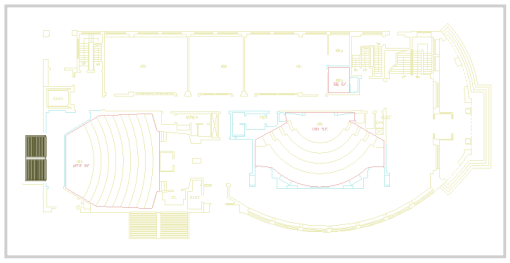
- Translation:
  - x' = x + tx
  - y' = y + ty
- Scale:
  - x' = x * sx
  - y' = y * sy
- Rotation:
  - x = x*cosΘ - y*sinΘ
  - y = x*sinΘ + y*cosΘ
- Shear:
  - x' = x + hx*y
  - y' = y + hy*x



1. Scale
2. Rotate
3. Translate

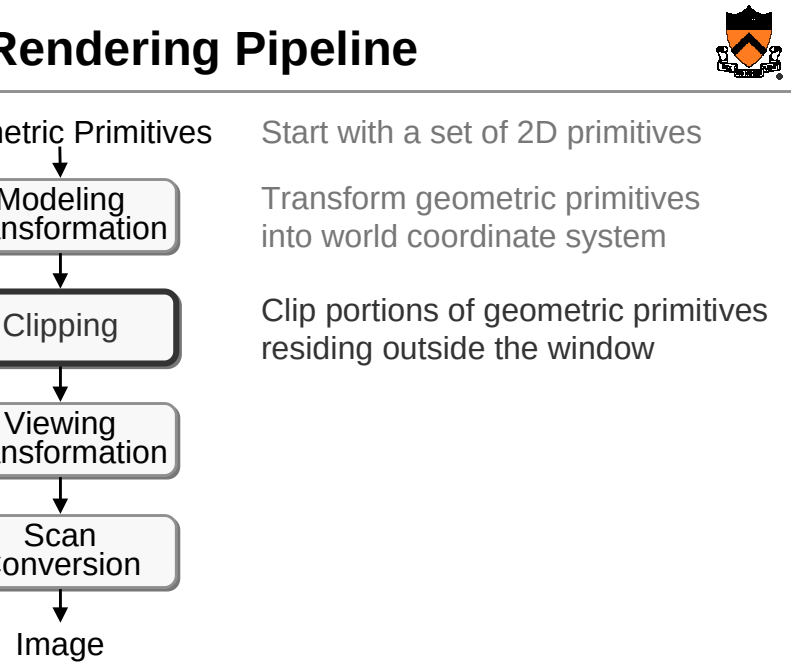
2D Modeling Transformations

- Translation:
  - x' = x + tx
  - y' = y + ty
- Scale:
  - x' = x * sx
  - y' = y * sy
- Rotation:
  - x = x*cosΘ - y*sinΘ
  - y = x*sinΘ + y*cosΘ
- Shear:
  - x' = x + hx*y
  - y' = y + hy*x



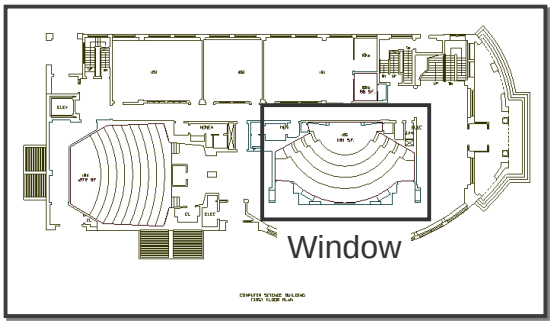
1. Scale
2. Rotate
3. Translate

2D Rendering Pipeline



Clipping

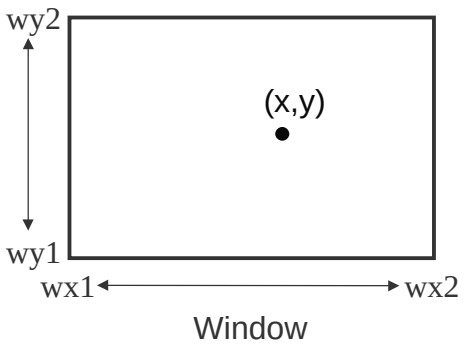
- Avoid drawing parts of primitives outside window
  - Window defines part of scene being viewed
  - Must draw geometric primitives only inside window



World Coordinates

Point Clipping

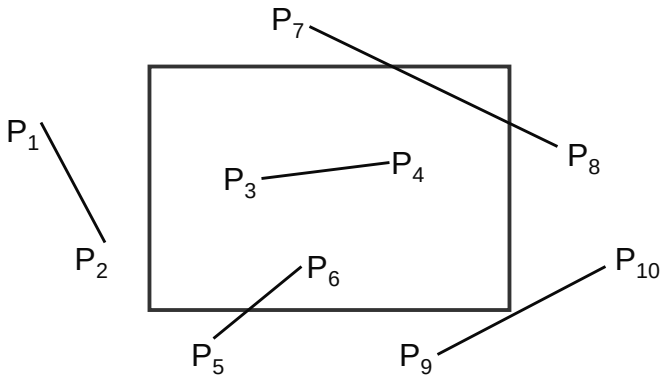
- Is point (x,y) inside the clip window?




```
inside =
  (x >= wx1) &&
  (x <= wx2) &&
  (y >= wy1) &&
  (y <= wy2);
```

Line Clipping

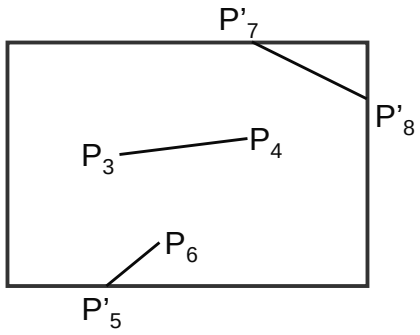
- Find the part of a line inside the clip window



Before Clipping

Line Clipping

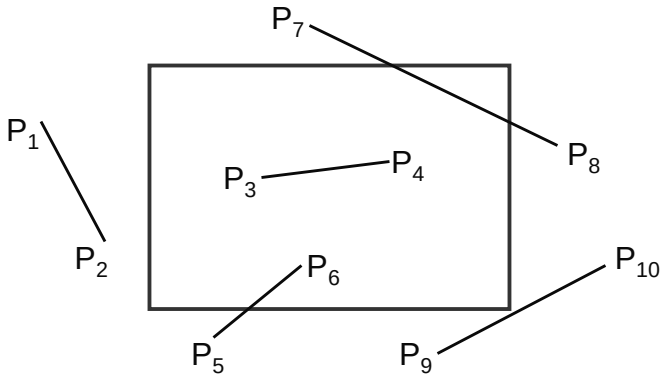
- Find the part of a line inside the clip window



After Clipping

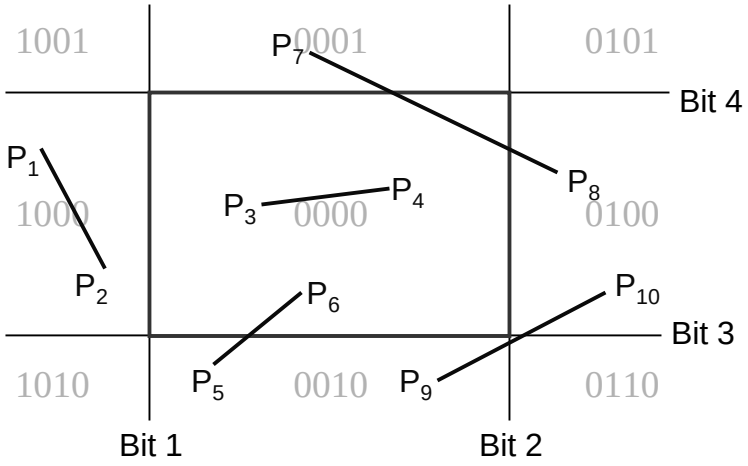
Cohen Sutherland Line Clipping

- Use simple tests to classify easy cases first



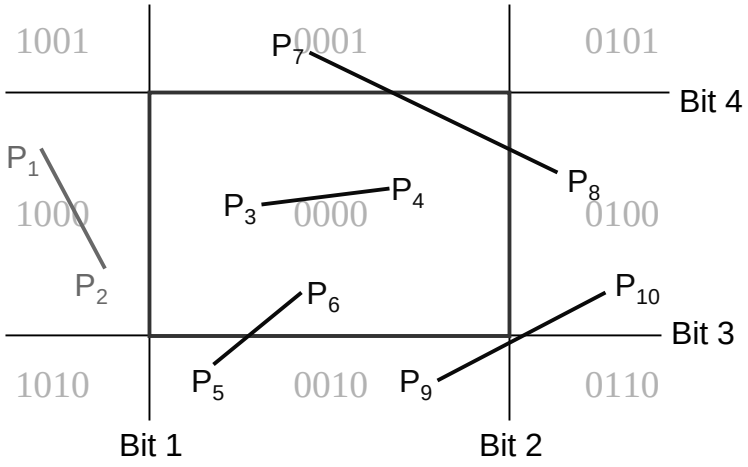
Cohen Sutherland Line Clipping

- Classify some lines quickly by AND of bit codes representing regions of two endpoints (must be 0)



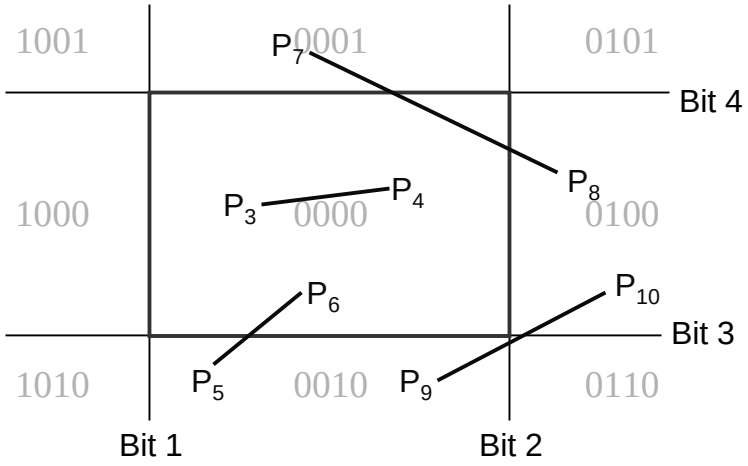
Cohen Sutherland Line Clipping

- Classify some lines quickly by AND of bit codes representing regions of two endpoints (must be 0)



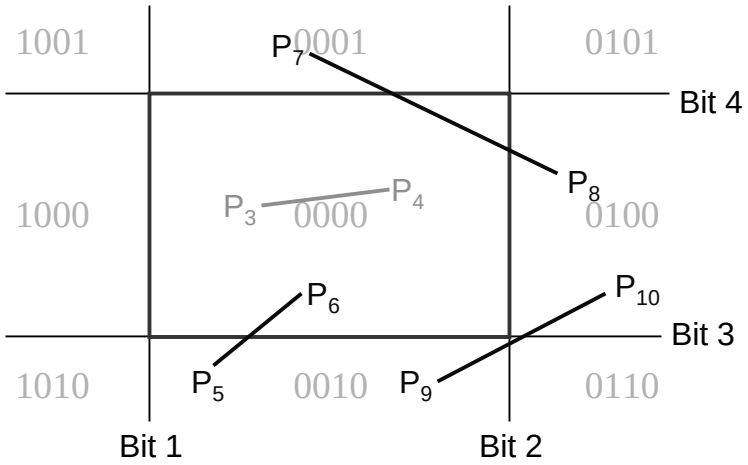
Cohen Sutherland Line Clipping

- Classify some lines quickly by AND of bit codes representing regions of two endpoints (must be 0)



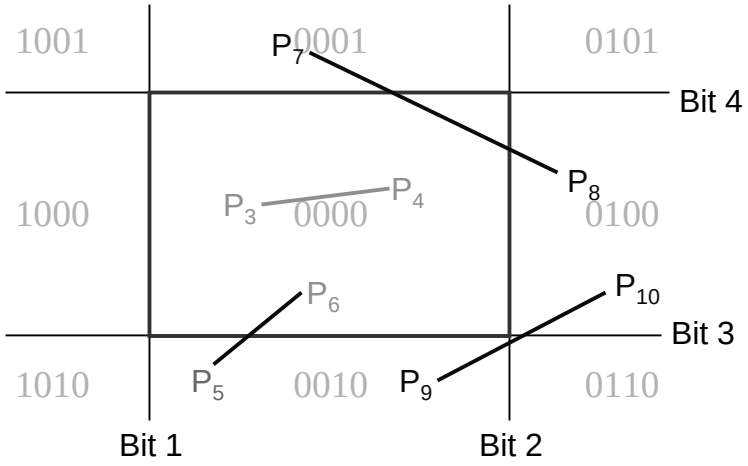
Cohen Sutherland Line Clipping

- Classify some lines quickly by AND of bit codes representing regions of two endpoints (must be 0)



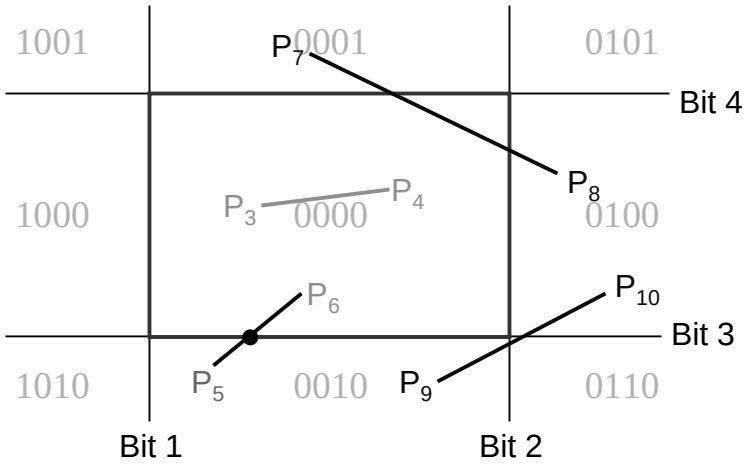
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



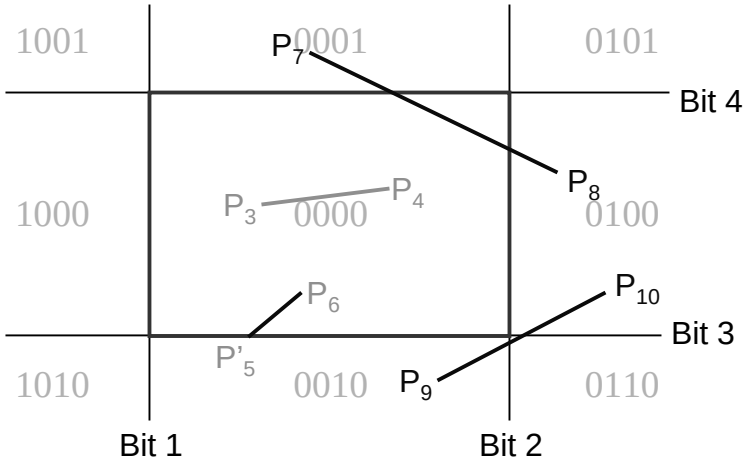
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



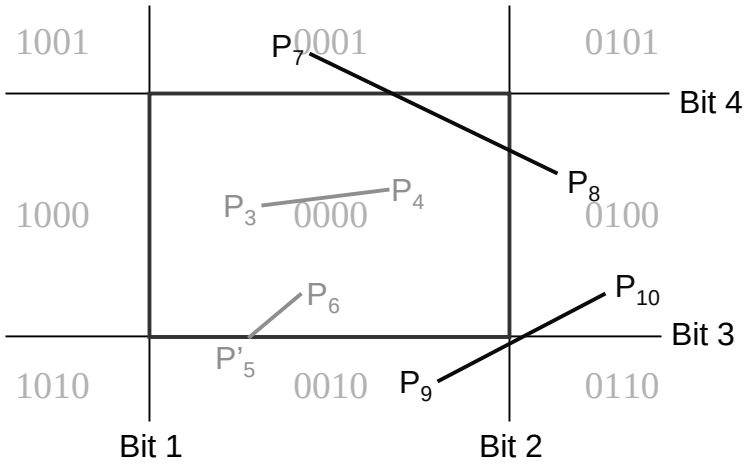
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



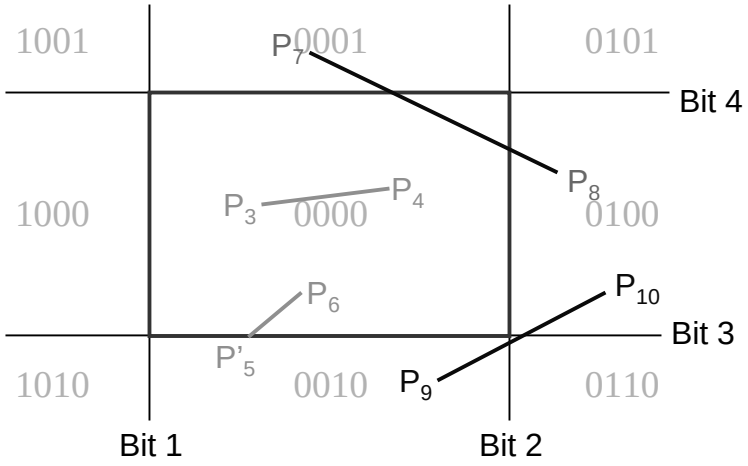
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



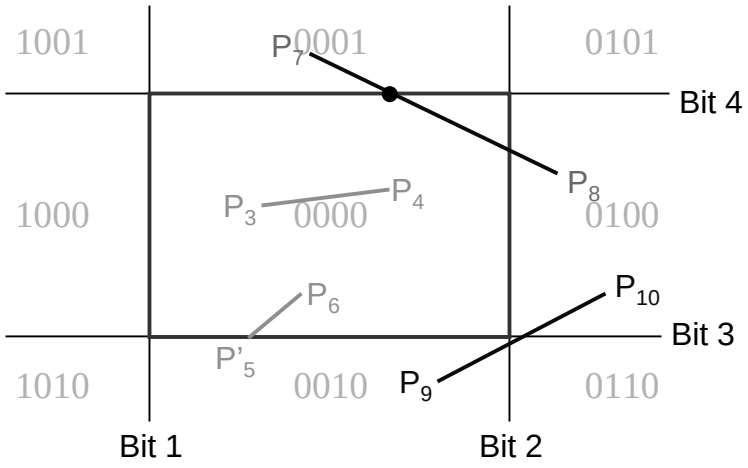
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



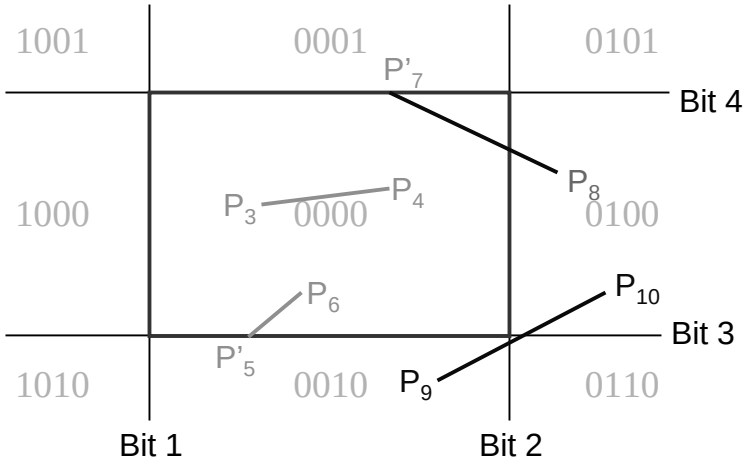
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



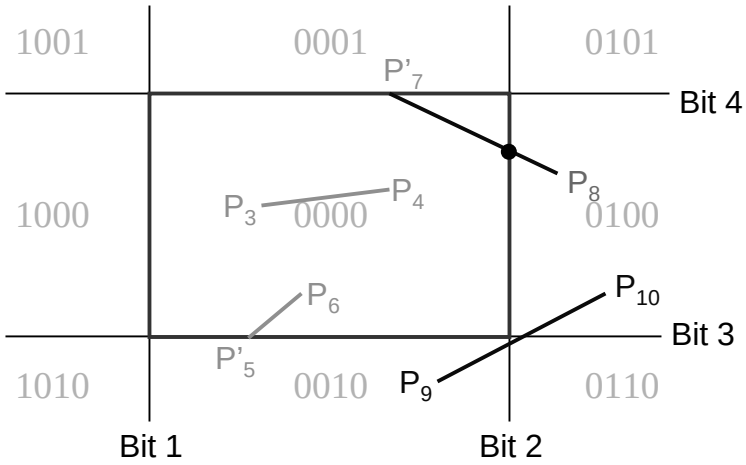
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



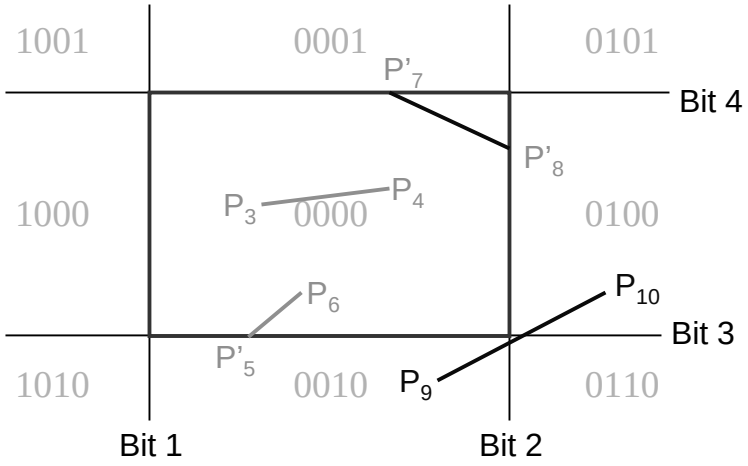
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



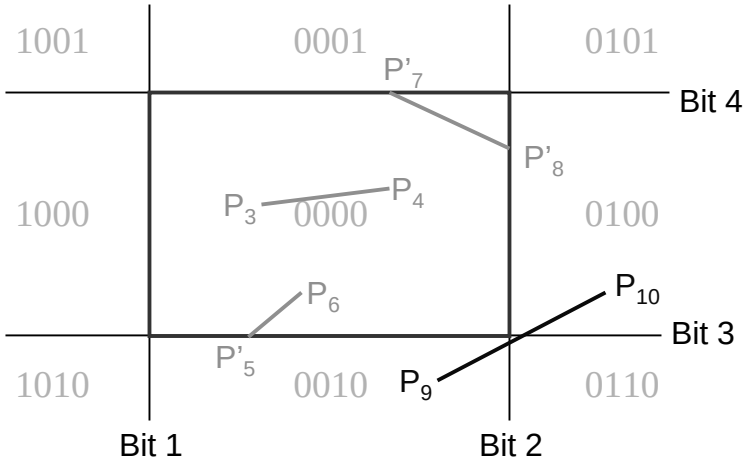
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



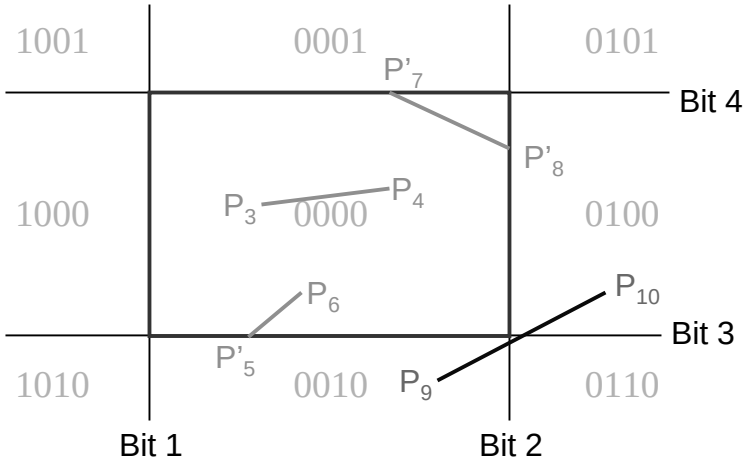
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



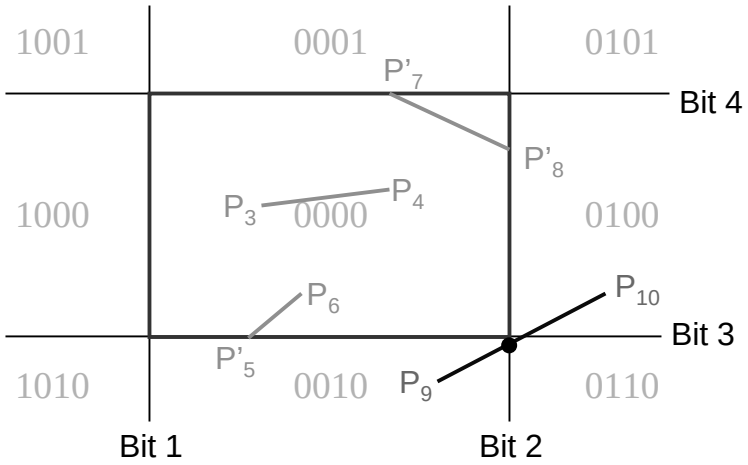
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



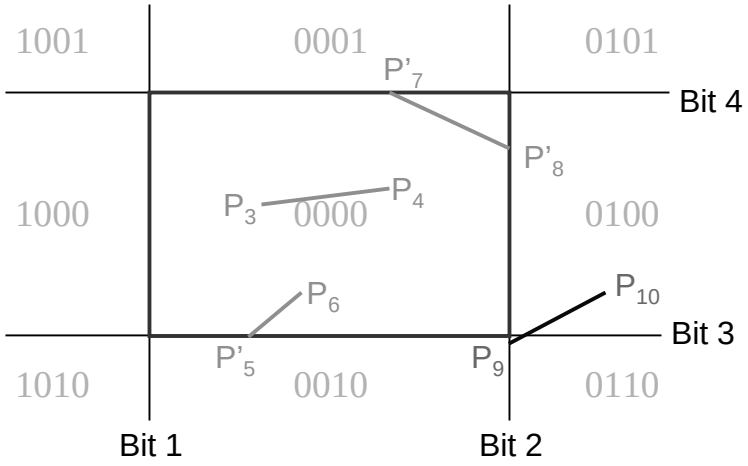
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



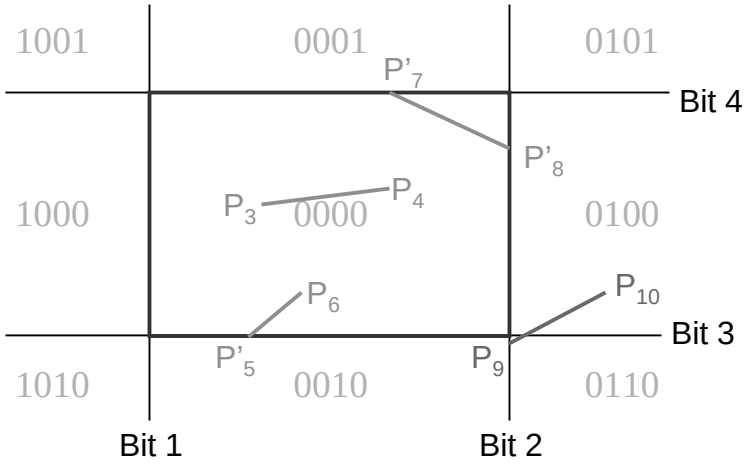
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



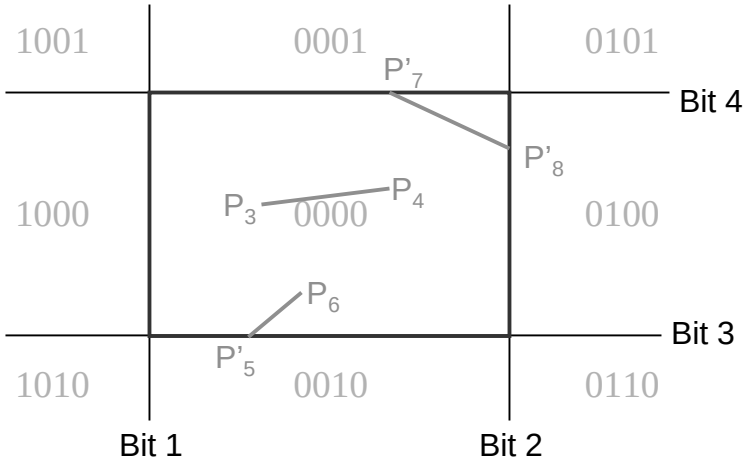
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



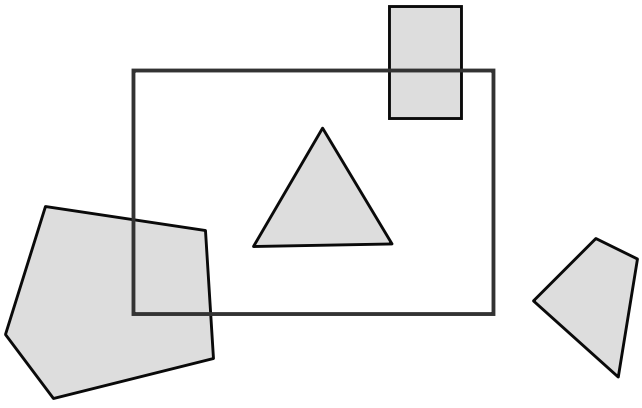
Cohen-Sutherland Line Clipping

- Compute interesections with window boundary for lines that can't be classified quickly



Polygon Clipping

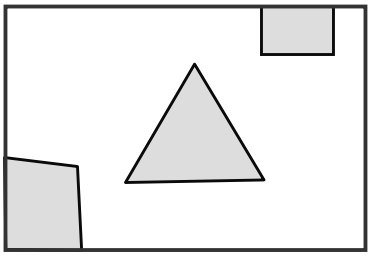
- Find the part of a polygon inside the clip window?



Before Clipping

Polygon Clipping

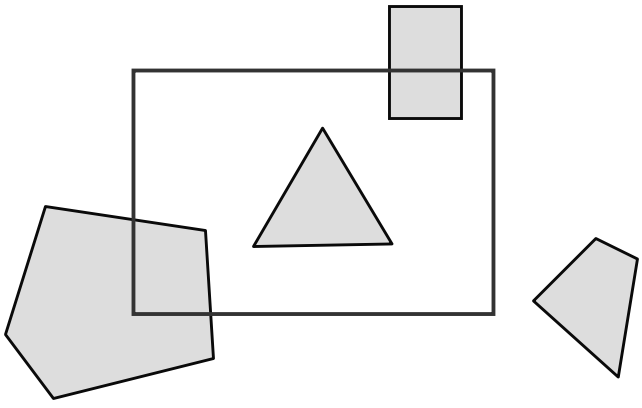
- Find the part of a polygon inside the clip window?



After Clipping

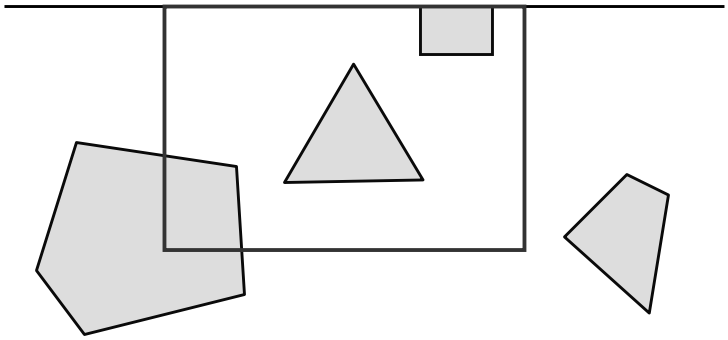
Sutherland Hodgeman Clipping

- Clip to each window boundary one at a time



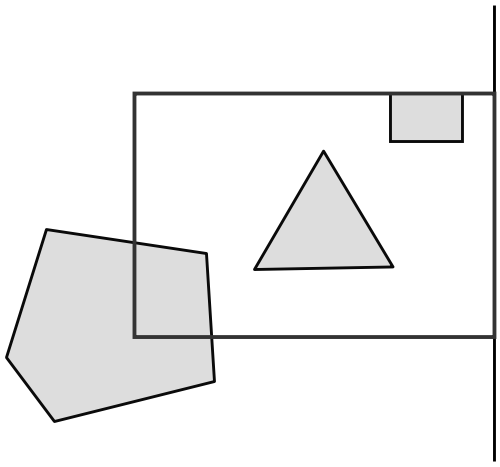
Sutherland Hodgeman Clipping

- Clip to each window boundary one at a time



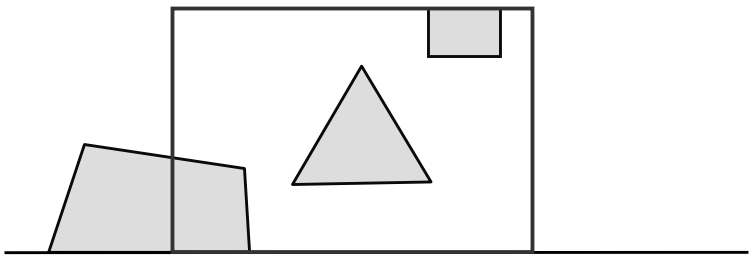
Sutherland Hodgeman Clipping

- Clip to each window boundary one at a time



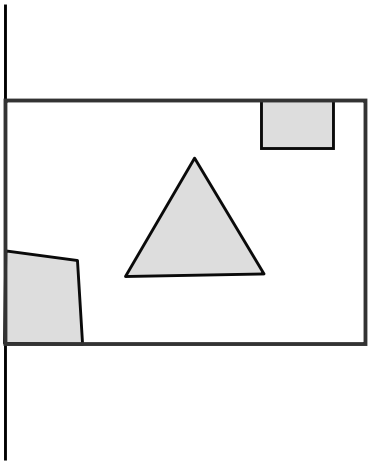
Sutherland Hodgeman Clipping

- Clip to each window boundary one at a time



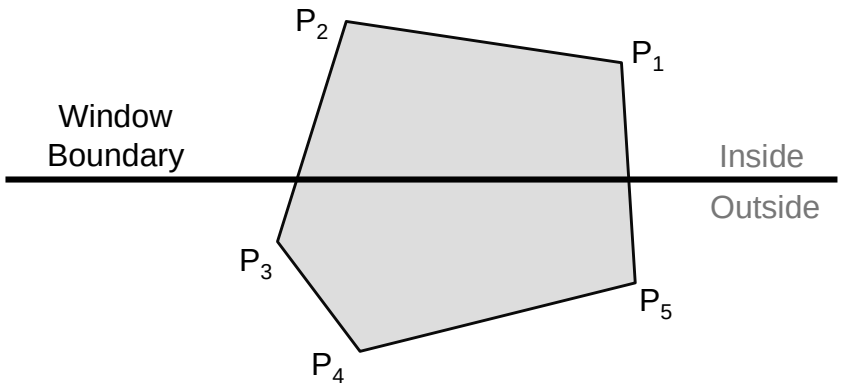
Sutherland Hodgeman Clipping

- Clip to each window boundary one at a time



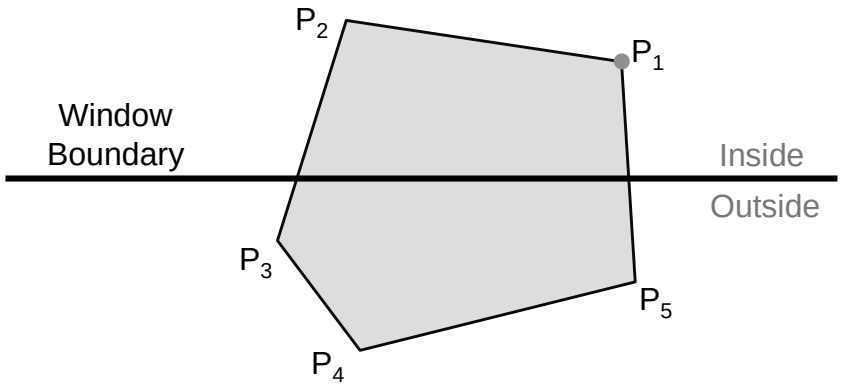
Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary



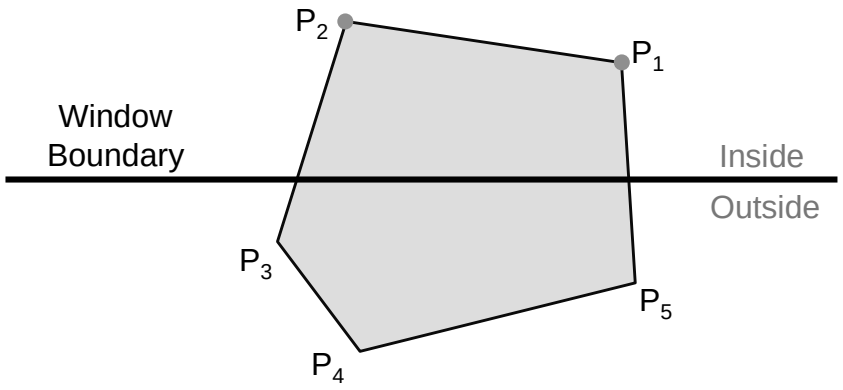
Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary



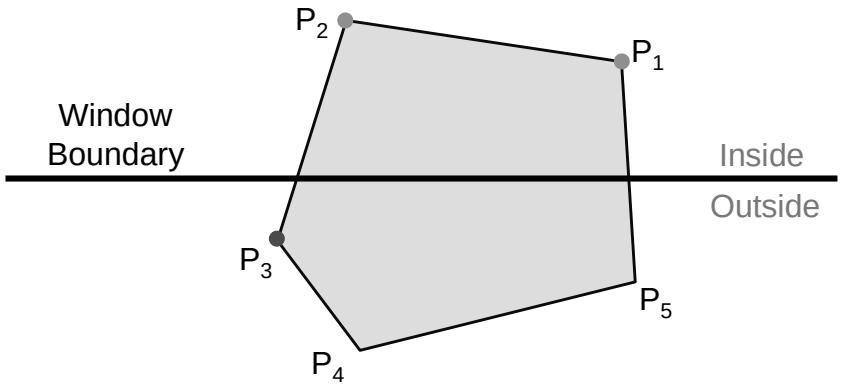
Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary



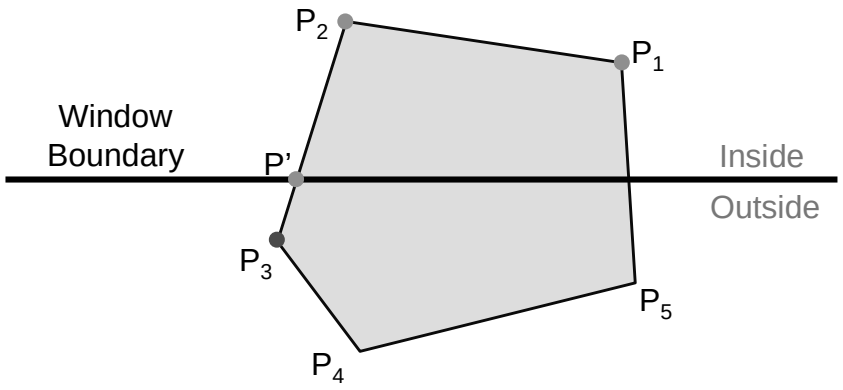
Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary



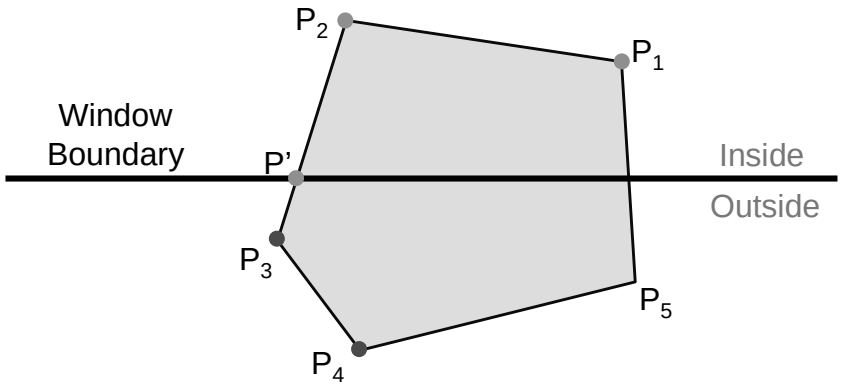
Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary



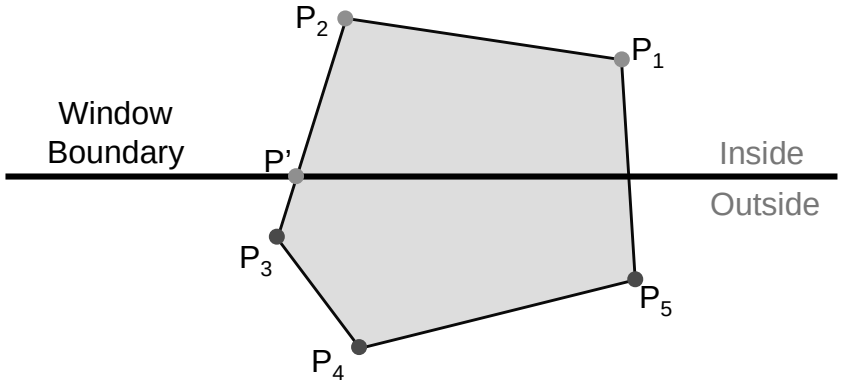
Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary



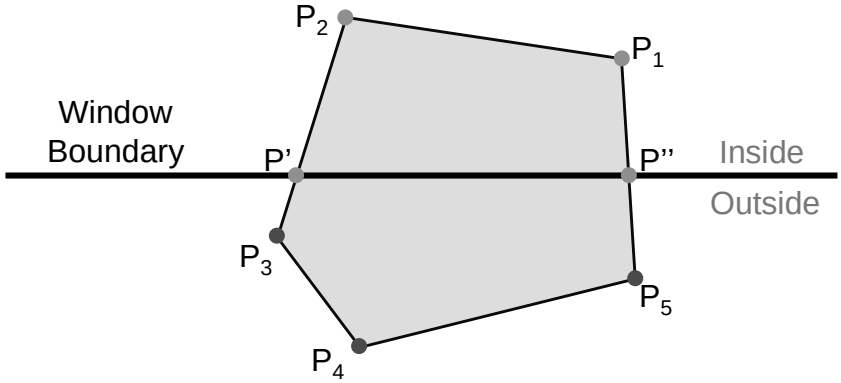
Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary



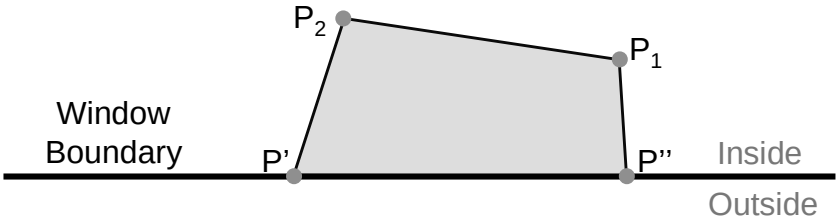
Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary

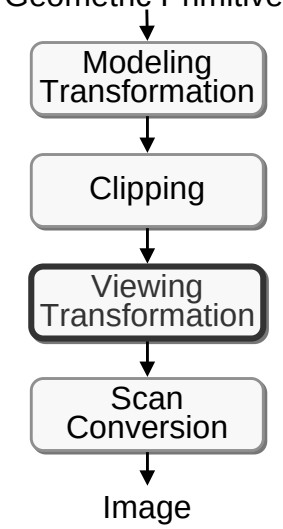


Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary



2D Rendering Pipeline



Start with a set of 2D primitives

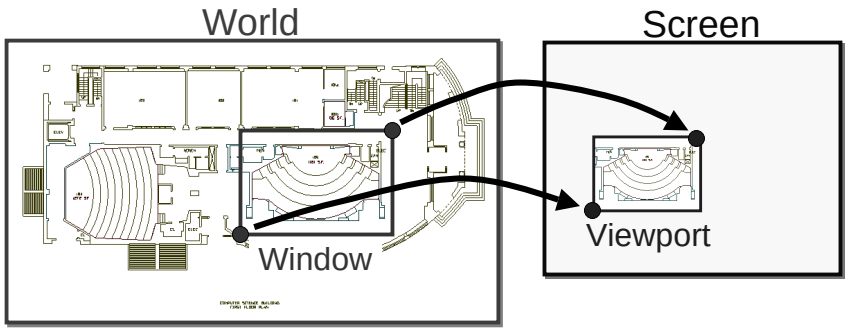
Transform geometric primitives into world coordinate system

Clip portions of geometric primitives residing outside the window

Transform the clipped primitives from world to screen coordinates

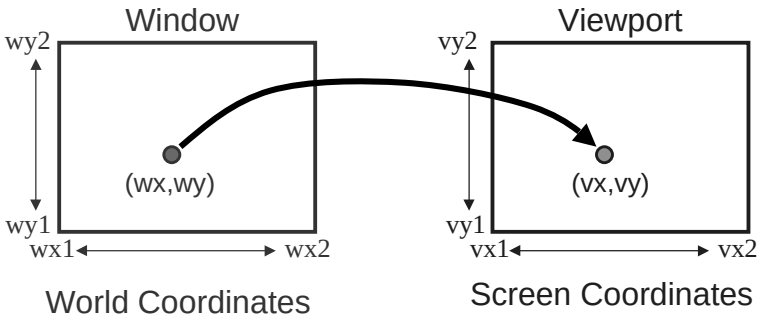
2D Viewing Transformation

- Transform 2D geometric primitives from application's world coordinate system to device's screen coordinate system



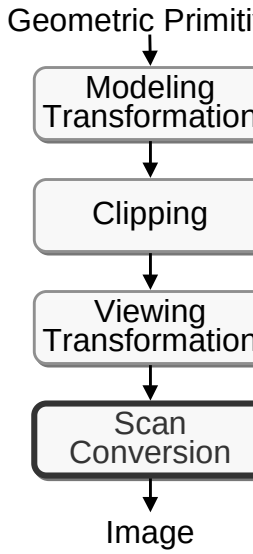
2D Viewing Transformation

- Window-to-viewport mapping




```
vx = vx1 + (wx - wx1) * (vx2 - vx1) / (wx2 - wx1);
vy = vy1 + (wy - wy1) * (vy2 - vy1) / (wy2 - wy1);
```

2D Rendering Pipeline



Start with a set of 2D primitives

Transform geometric primitives into world coordinate system

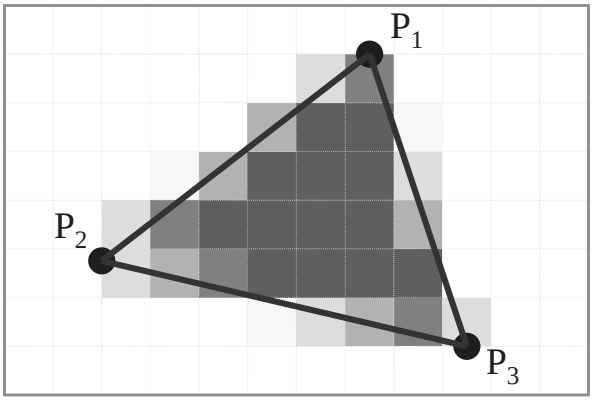
Clip portions of geometric primitives residing outside the window

Transform the clipped primitives from world to screen coordinates

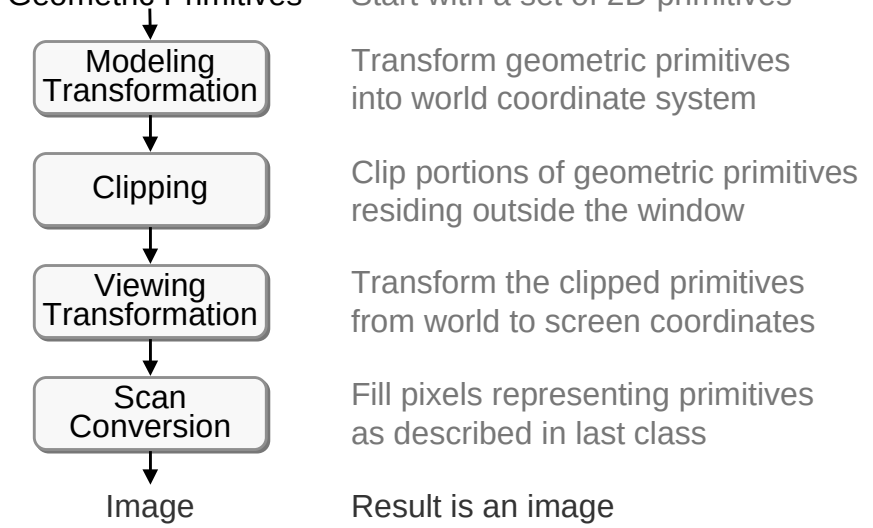
Fill pixels representing primitives as described in last class

Scan Conversion

- Fill pixels representing geometric primitive
  - Been there, done that … last class



2D Rendering Pipeline



Summary

- 2D Rendering Pipeline
  - Modeling transformation
  - Clipping
  - Viewing transformation
  - Scan conversion
- Transformations change coordinate systems
  - Modeling-to-world mapping
  - Window-to-viewport mapping