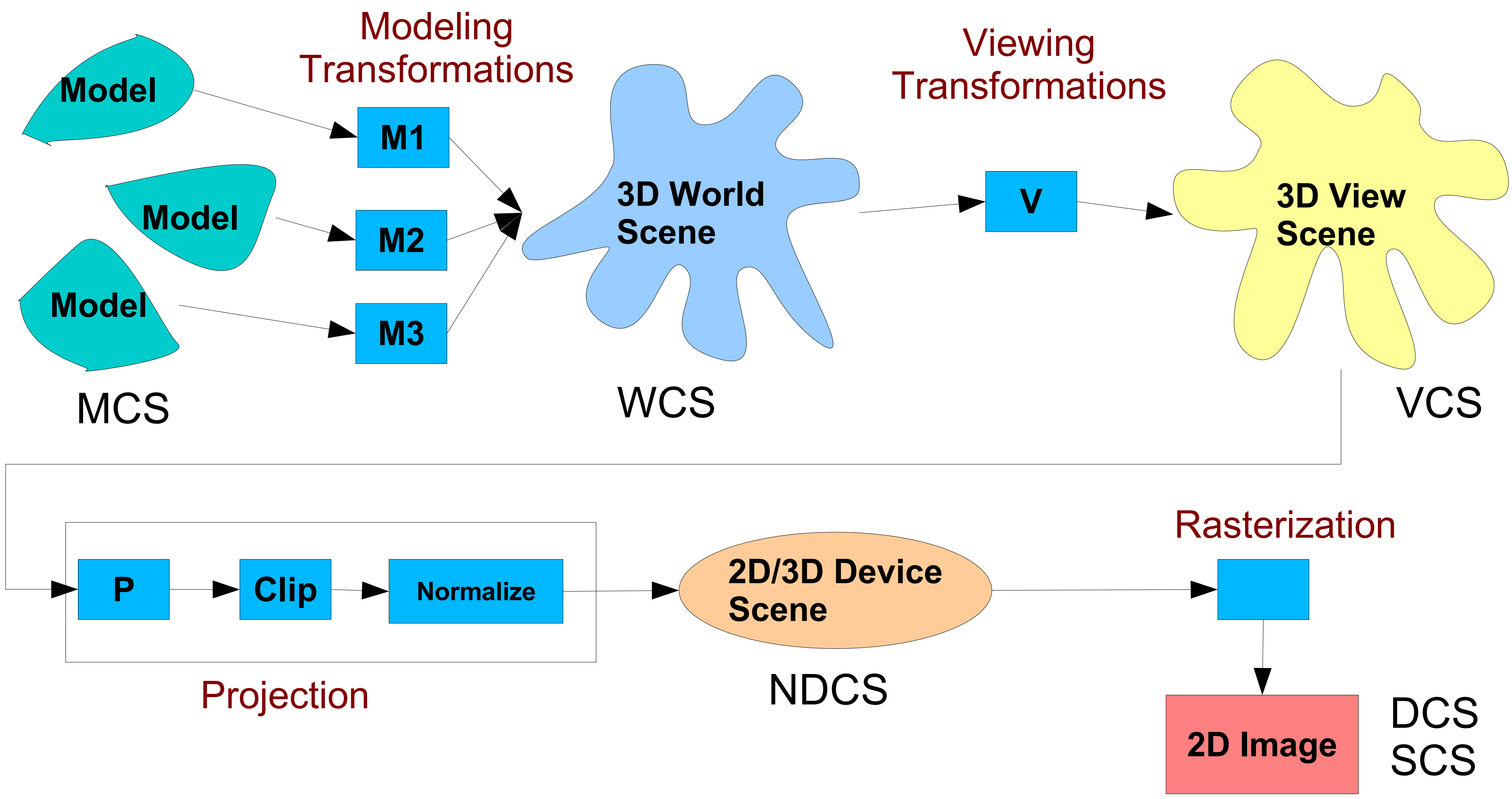
2 DIMENSIONAL VIEWING

Ceng 477 Computer Engineering
Lecture notes
METU

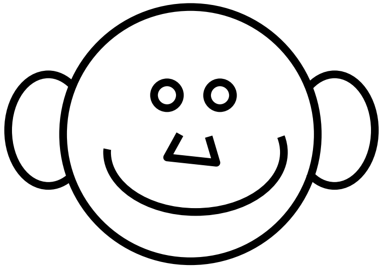
Viewing Pipeline Revisited



- Model coordinates to World coordinates: Modelling transformations

Model coordinates:
1 circle (head),
2 circles (eyes),
1 line group (nose),
1 arc (mouth),
2 arcs (ears).
With their relative
coordinates and sizes

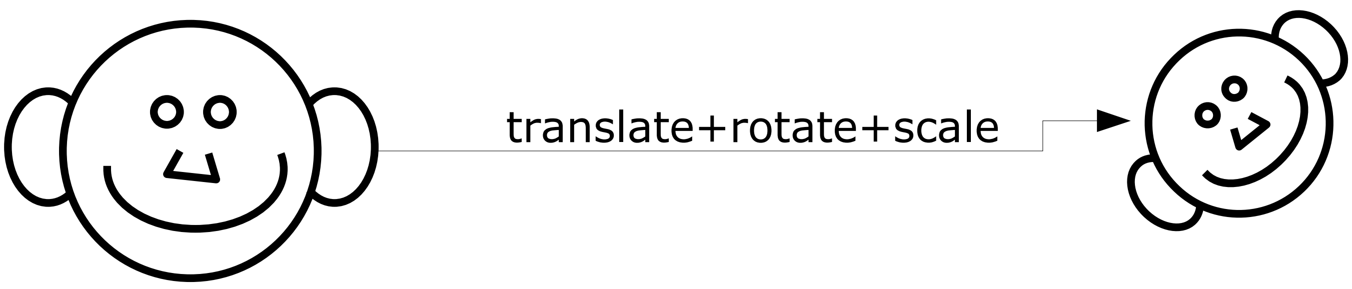
World coordinates:
All shapes with their
absolute coordinates and sizes.
circle(0,0,2)
circle(-.6,.8,.3) circle(.6,.8,.3)
lines[(-.4,0),(-.5,-.3),(.5,.3),(.4,0)]
arc(-.6,0,.6,0,1.8,180,360)
arc(-2.2,.2,-2.2,-.2,.8,45,315)
arc(2.2,.2,2.2,-.2,.8,225,135)



- World coordinates to Viewing coordinates: Viewing transformations

World coordinates

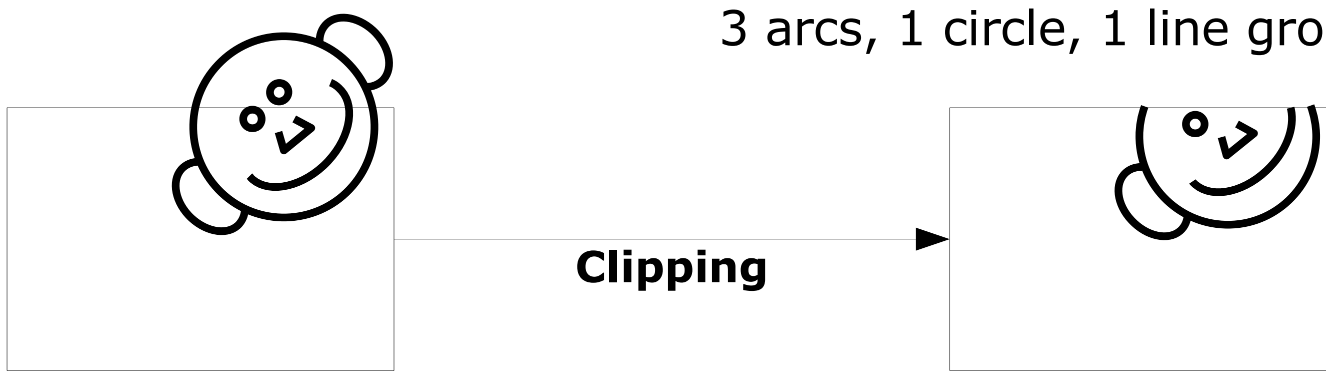
Viewing coordinates:
Viewers position and view angle. i.e. rotated/translated



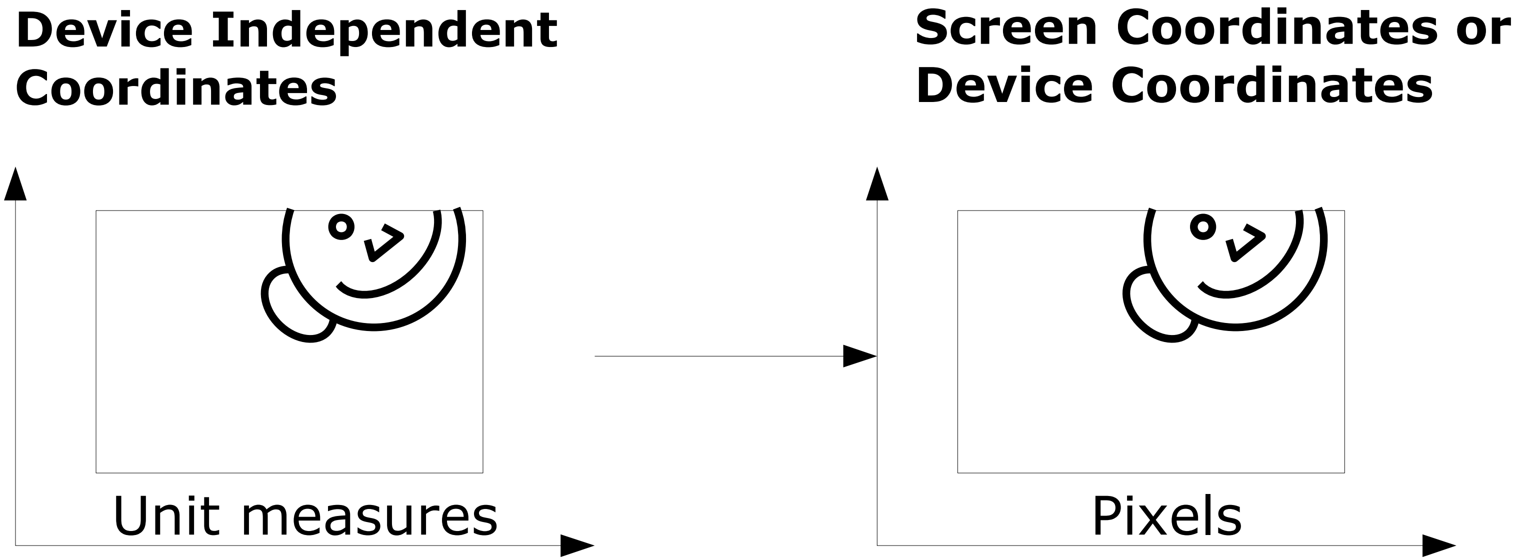
- Projection: 3D to 2D. Clipping depends on viewing frame/volume. Normalization: device independent coordinates (ie. cm.)

Viewing coordinates:

Device Independent Coordinates:
Invisible shapes deleted, others reduced to visible parts.
3 arcs, 1 circle, 1 line group



- Device Independent Coordinates to Device Coordinates. Rasterization



Basic Geometric Transformations

- Given the shape, transform all the points of the shape? Transform the points and/or vectors describing it.
- For example:
  Polygon: corner points
  Circle, Ellipse: center point(s), point at angle 0
- Some transformations preserves some of the attributes like sizes, angles, ratios of the shape.

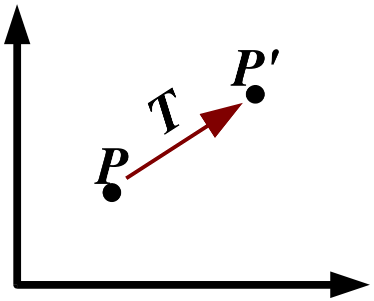
Translation

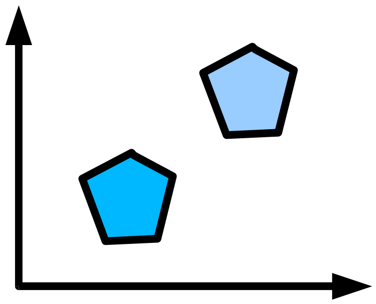
- Simply move the object to a relative position.

$$x' = x + t_x \quad y' = y + t_y$$

$$\boldsymbol{P} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \boldsymbol{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad \boldsymbol{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$\boldsymbol{P}' = \boldsymbol{P} + \boldsymbol{T}$$

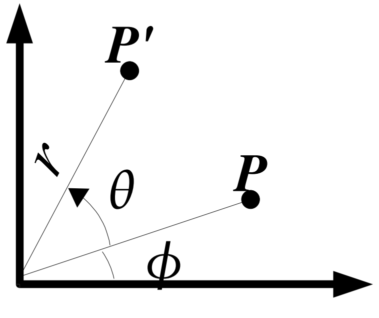




Rotation

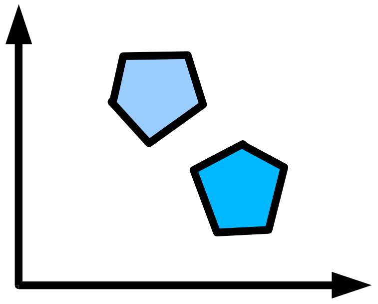
- Reposition the object in a circular path arround a point.

$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$



$$x = r\cos\phi \qquad y = r\sin\phi$$

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$



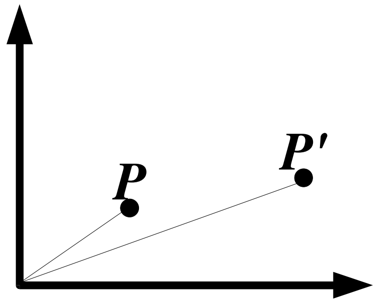
$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\boldsymbol{P}' = \boldsymbol{R} \cdot \boldsymbol{P}$$

Scaling

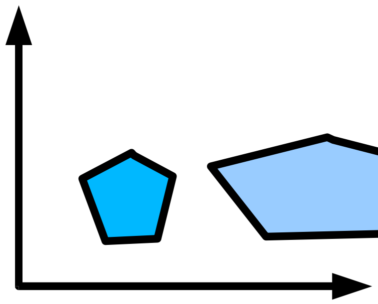
- Change the sizes of the object.

$$x' = x\, s_x \qquad y' = y\, s_y$$



$$\boldsymbol{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

$$\boldsymbol{P}' = \boldsymbol{S} \cdot \boldsymbol{P}$$



Homogenous Coordinates

- All transformations can be represented by matrix operations.
- Translation is additive, rotation and scaling is multiplicative; making the operations complicated.
- Adding another dimension to transformations make translation also representable by multiplication. Cartesian coordinates vs homogenous coordinates.

$$x = \frac{x_h}{h} \quad y = \frac{y_h}{h} \qquad \boldsymbol{P} = \begin{bmatrix} x_h \\ y_h \\ h \end{bmatrix} = \begin{bmatrix} h \cdot x \\ h \cdot y \\ h \end{bmatrix}$$

- Many points in homogenous coordinates can represent same point in cartesian coordinates.
- In homogenous coordinates, all transformations can be written as multiplications.

Transformations in Homogenous C.

- Translation

$$\boldsymbol{T}(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{P}' = \boldsymbol{T}(t_x, t_y) \cdot \boldsymbol{P}$$

- Rotation

$$\boldsymbol{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{P}' = \boldsymbol{R}(\theta) \cdot \boldsymbol{P}$$

- Scaling

$$\boldsymbol{S}(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{P}' = \boldsymbol{S}(s_x, s_y) \cdot \boldsymbol{P}$$

Composite Transformations

- Composition of similar type transformations

$$\boldsymbol{T}(t_{x1}, t_{y1}) \cdot \boldsymbol{T}(t_{x2}, t_{y2}) = \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x2}+t_{x1} \\ 0 & 1 & t_{y2}+t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{T}(t_{x1}+t_{x2}, t_{y1}+t_{y2})$$

$$\boldsymbol{R}(\theta) \cdot \boldsymbol{R}(\phi) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

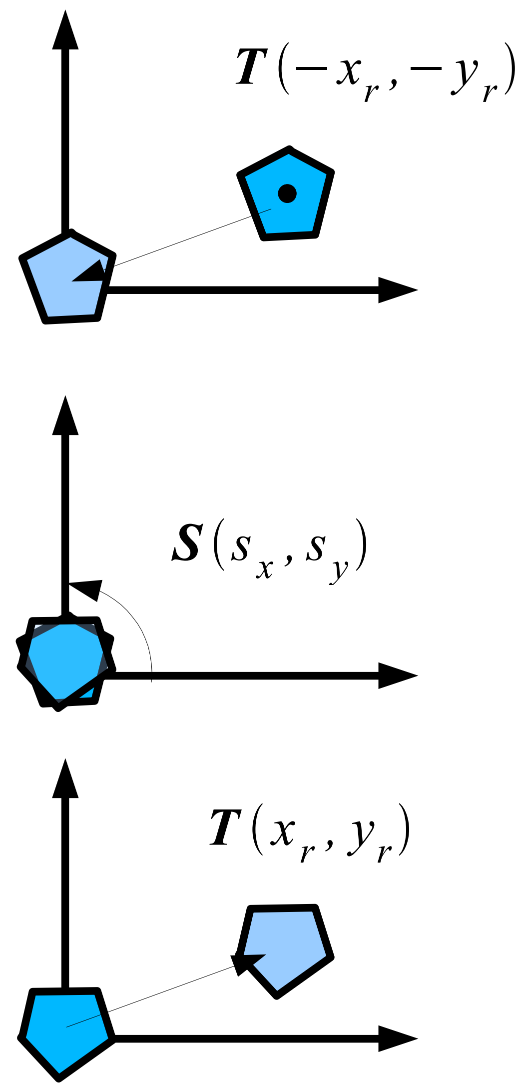
$$\begin{bmatrix} \cos\theta\cos\phi - \sin\theta\sin\phi & -\cos\theta\sin\phi - \sin\theta\cos\phi & 0 \\ \sin\theta\cos\phi + \cos\theta\sin\phi & -\sin\theta\sin\phi + \cos\theta\cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta+\phi & -\sin\theta+\phi & 0 \\ \sin\theta+\phi & \cos\theta+\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{R}(\theta+\phi)$$

$$\boldsymbol{S}(s_{x1}, s_{y1}) \cdot \boldsymbol{S}(s_{x2}, s_{y2}) = \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1} s_{x2} & 0 & 0 \\ 0 & s_{y1} s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{S}(s_{x1} \cdot s_{x2}, s_{y1} \cdot s_{y2})$$

- Rotation on a pivot point
  - Translate to origin
  - Rotate
  - Translate back
- $\boldsymbol{T}(x_r, y_r) \cdot \boldsymbol{R}(\theta) \cdot \boldsymbol{T}(-x_r, -y_r) =$

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix} =$$

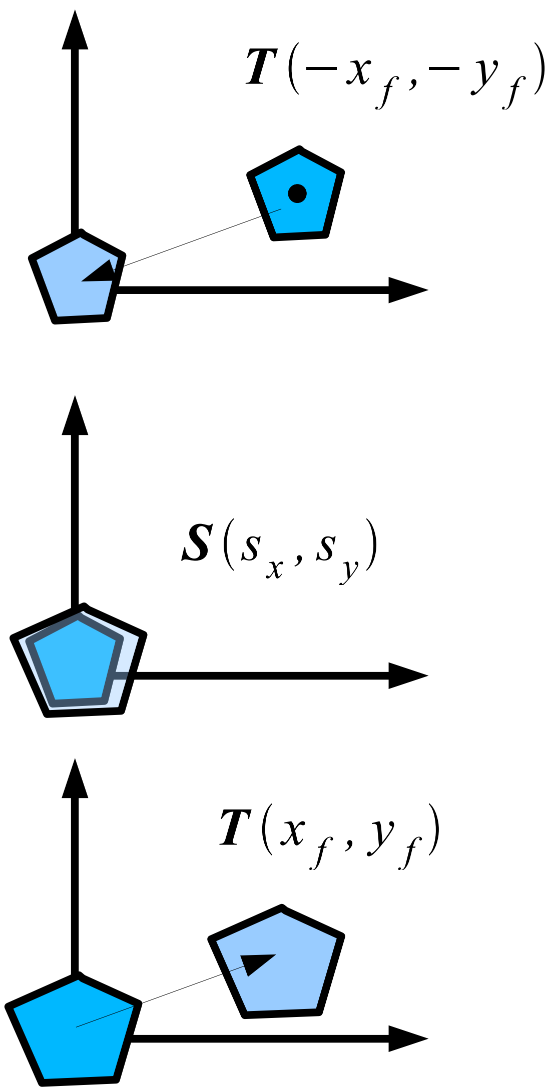
$$\begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta)+y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta)-x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$



- Scaling on a fixed point
  - Translate to origin
  - Scale
  - Translate back
- $\boldsymbol{T}(x_f, y_f) \cdot \boldsymbol{S}(s_x, s_y) \cdot \boldsymbol{T}(-x_f, -y_f) =$

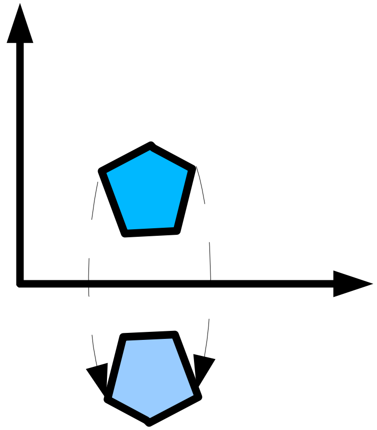
$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

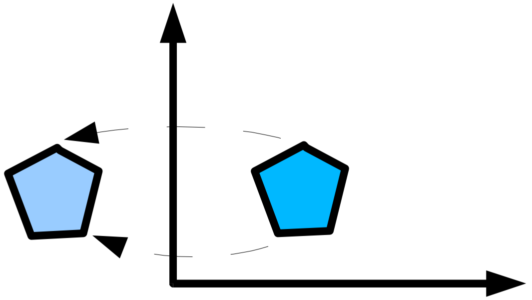


Other Transformations

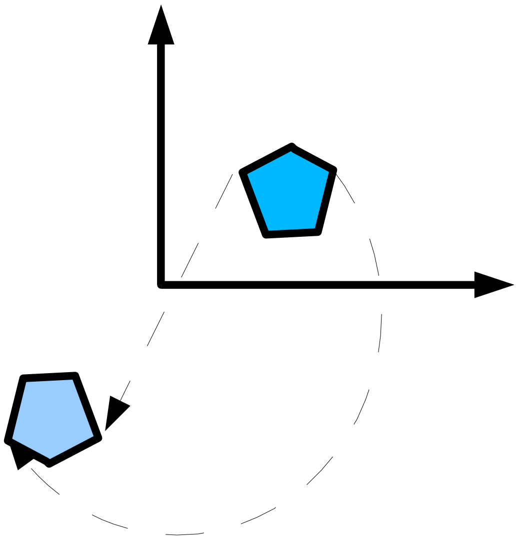
- Reflection



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

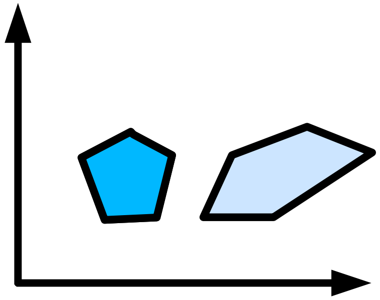


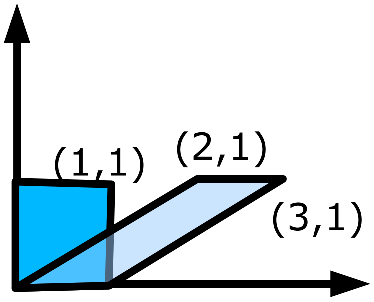
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



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

- Shear: Deform the shape like shifted slices.



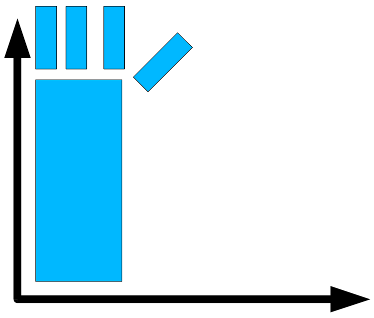


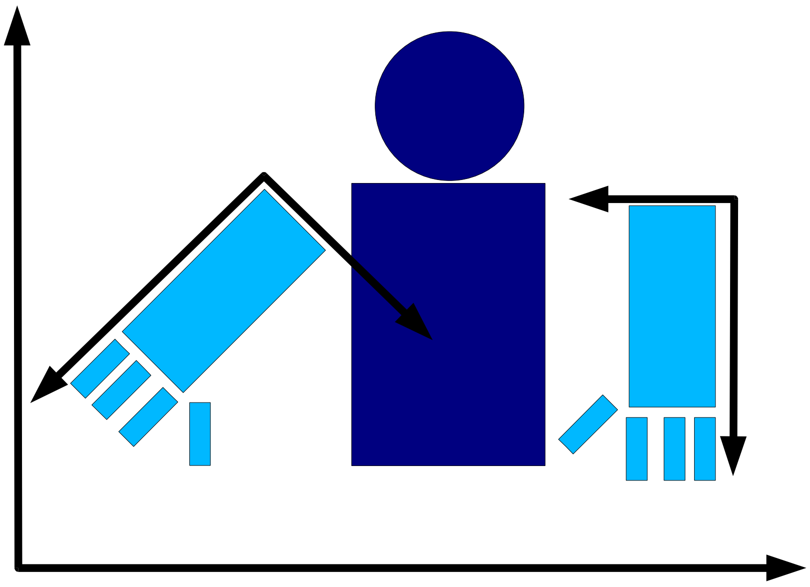
$$x' = x + sh_x \cdot y \qquad y' = y$$

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

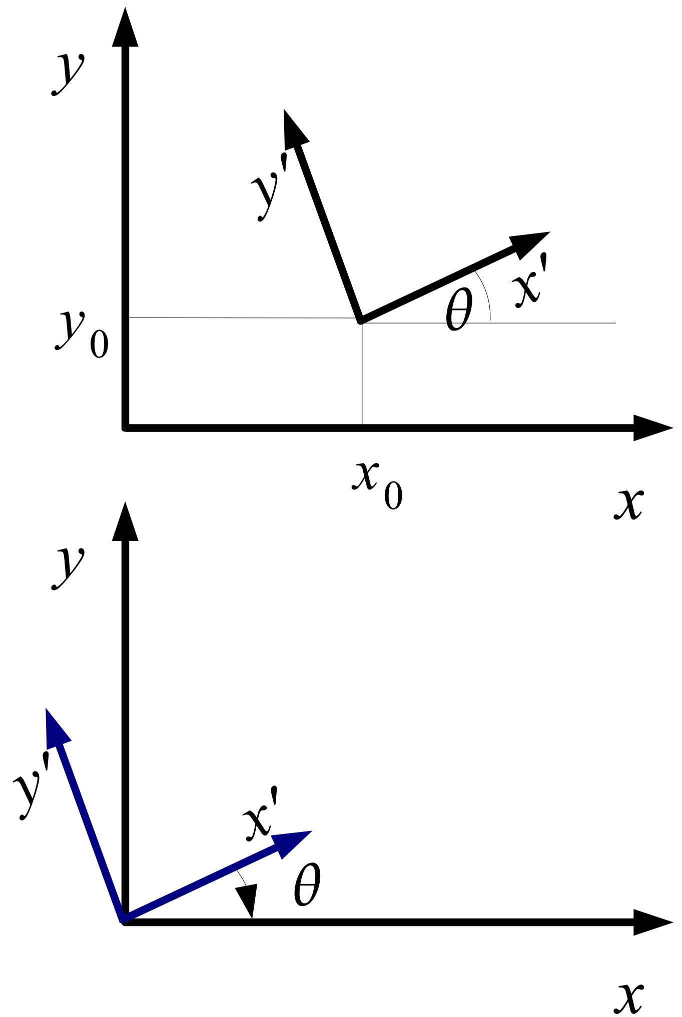
Transformations Between the Coordinate Systems

- Between different systems: Polar coordinates to cartesian coordinates
- Between two cartesian coordinate systems. For example relative coordinates or window to viewport transformation.





- How to transform from *x*,*y* to *x′*,*y′* ?
- Superimpose *x′*,*y′* to *x*,*y*
- Transformation:
  - Translate so that (x_0, y_0) moves to (0,0) of *x*,*y*
  - Rotate *x′* axis to *x* axis
- $\boldsymbol{R}(-\theta)\cdot\boldsymbol{T}(-x_{0,}-y_0)$



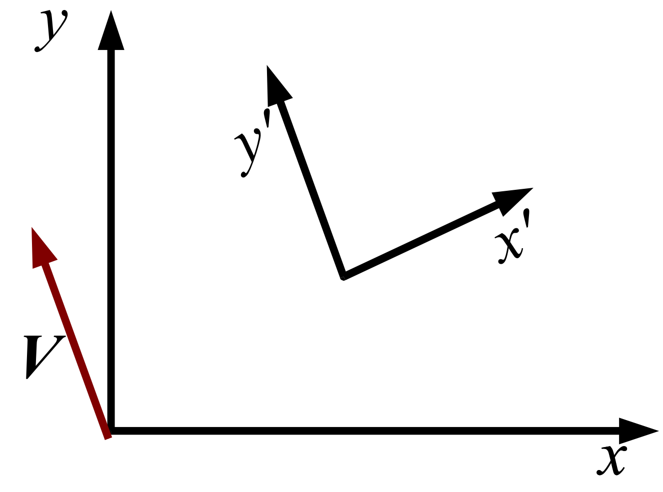
- Alternate method:
  Specify a vector V for positive y' axis:

unit vector in the y' *direction:*

$$\vec{v} = \frac{\vec{V}}{|\vec{V}|} = (v_x, v_y)$$

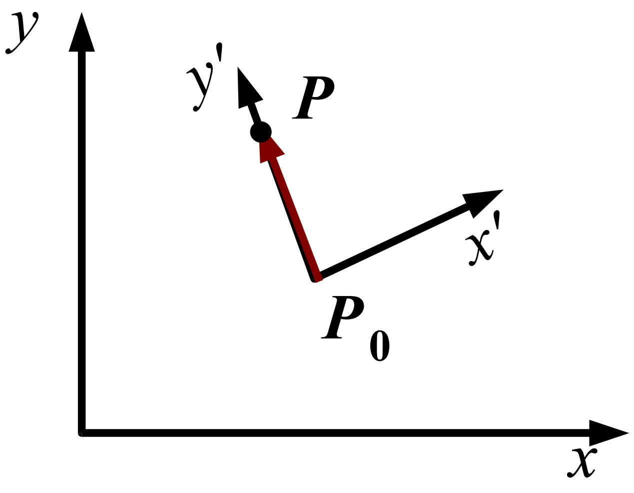
unit vector in the x' *direction, rotate* $\vec{v}$ *90*

$$\vec{u} = (v_y, -v_x) = (u_x, u_y)$$



- Elements of any rotation matrix can be expressed as elements of a set of orthogonal unit vectors:

$$R = \begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} v_y & -v_x & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\vec{V} = \frac{\boldsymbol{P} - \boldsymbol{P_0}}{|\boldsymbol{P} - \boldsymbol{P_0}|}$$

- Example:

$P_0 = (2,1) \quad P = (3.5, 3)$

$\vec{v} = \frac{P - P_0}{|P - P_0|} = \frac{(1.5, 2)}{\sqrt{1.5^2 + 2^2}} = (\frac{1.5}{2.5}, \frac{2}{2.5}) = (0.6, 0.8)$

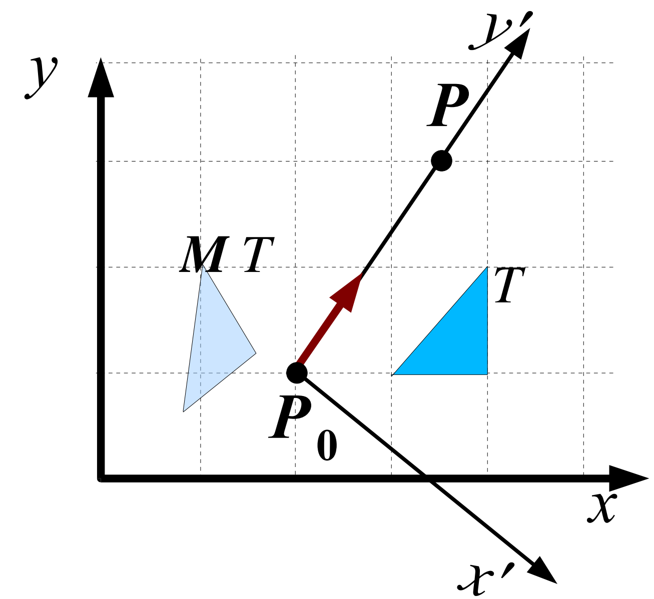
$\vec{u} = (0.8, -0.6)$

$\boldsymbol{M} = \boldsymbol{R}(\vec{u}, \vec{v}) \cdot \boldsymbol{T}(-2, -1) =$

$$\begin{bmatrix} 0.8 & -0.6 & 0 \\ 0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.8 & -0.6 & -1 \\ 0.6 & 0.8 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

Let triangle T *is defined as column vectors :* $\begin{bmatrix} 3 & 4 & 4 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

$$\boldsymbol{M} \cdot T = \begin{bmatrix} 0.8 & 1 & 1.6 \\ 0.6 & 2 & 1.2 \\ 1 & 1 & 1 \end{bmatrix}$$



Affine Transformations

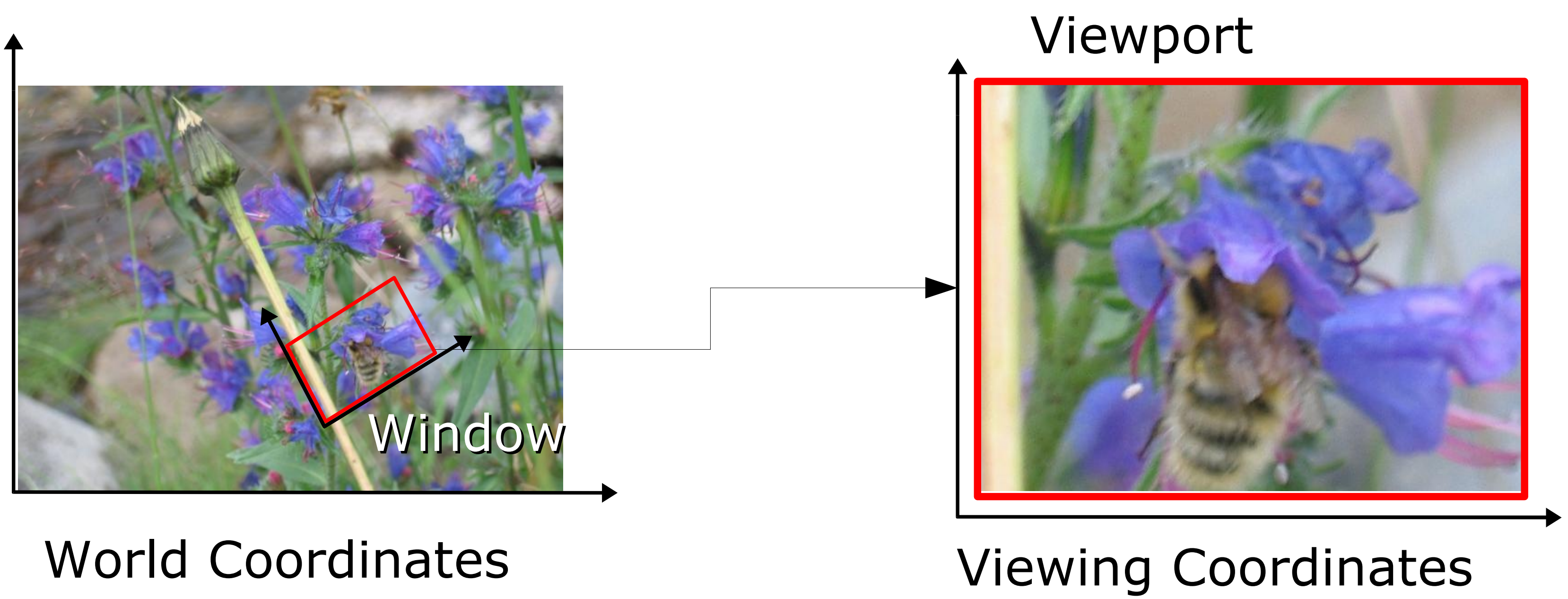
- Coordinations of the form:

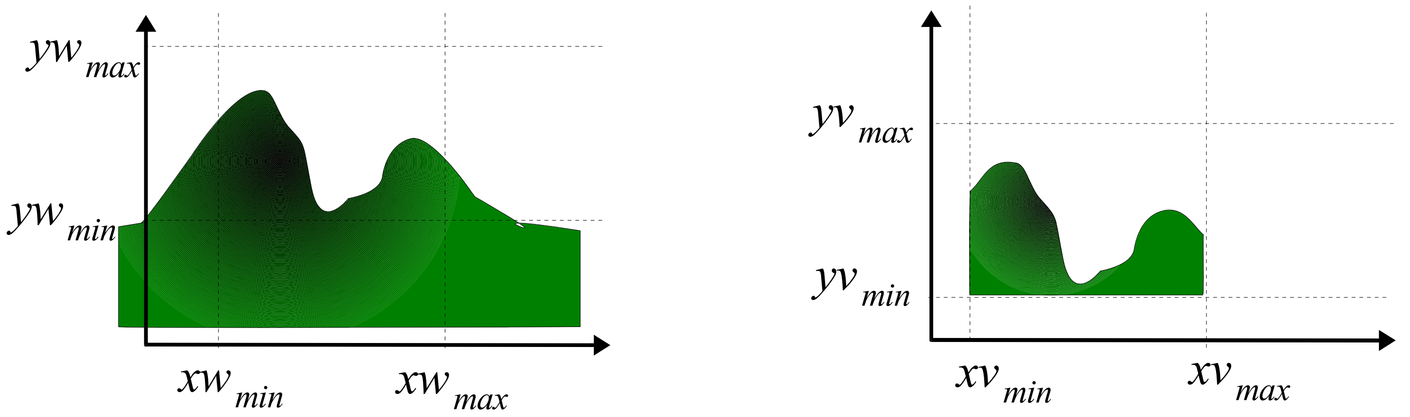
$$x' = a_{xx} x + a_{xy} y + b_x$$
$$y' = a_{yx} x + a_{yy} y + b_y$$

- Translation, rotation, scaling, reflection, shear. Any affine transformation can be expressed as the combination of these.
- Rotation, translation, reflection:
  preserve angles, lengths, parallel lines

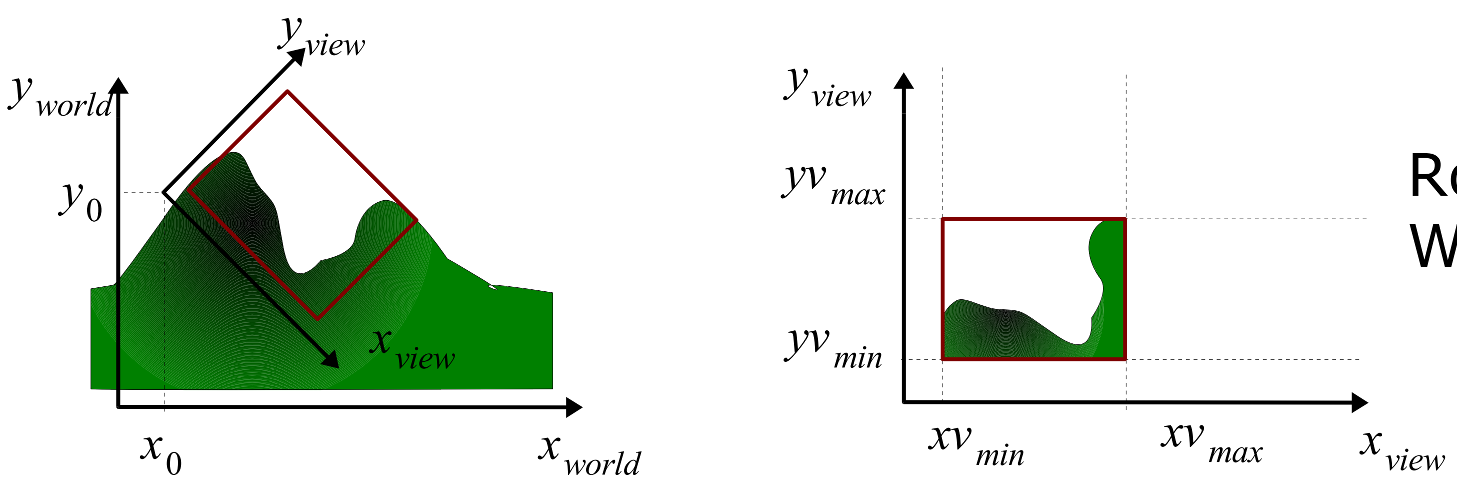
2D Viewing

- World coordinates to Viewing coordinates
- Window to Viewport.
  Window: A region of the scene selected for viewing
  Viewport: A region on display device for mapping to window



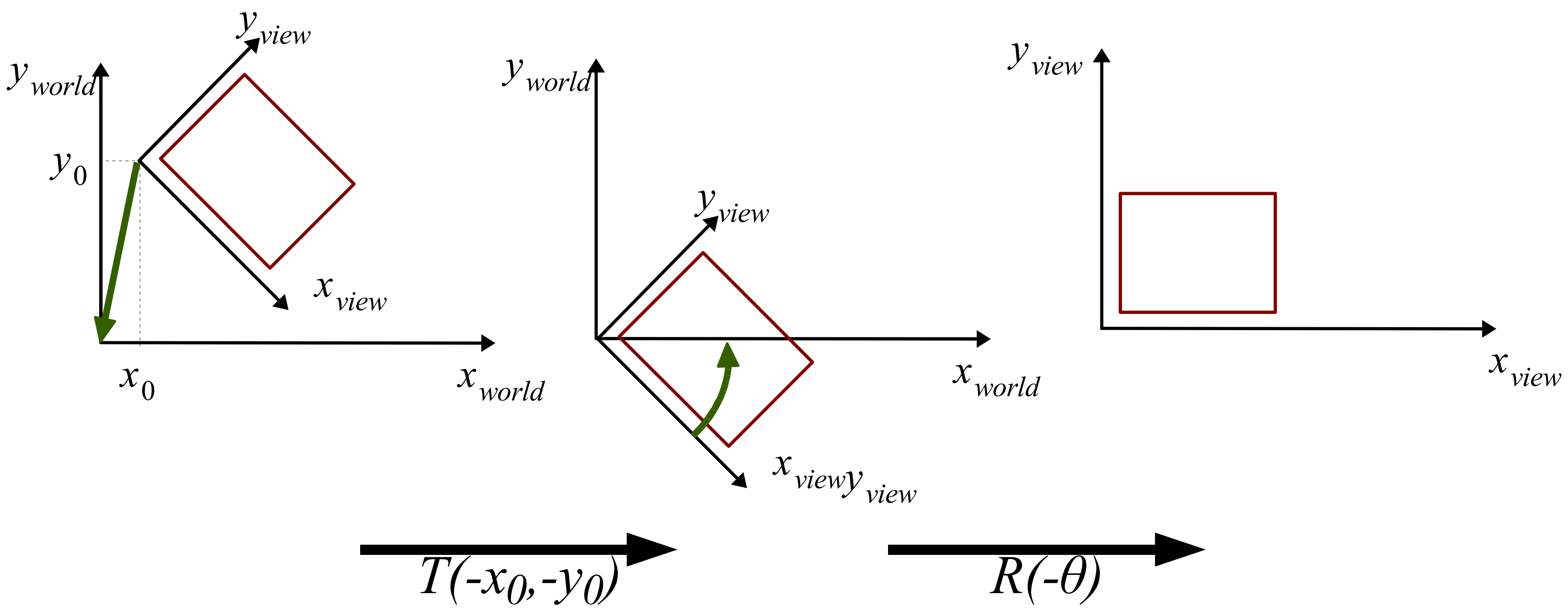


Rectangular Window



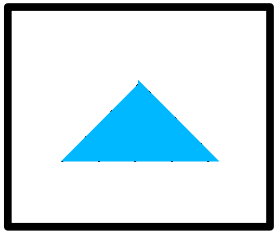
Rotated Window

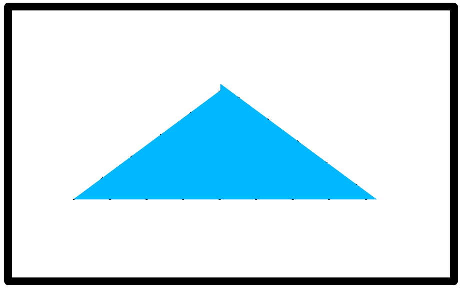
- Window coordinates to viewing coordinates:



- $M_{wc,vc} = R \cdot T$

- Coordinate transformation: Different sizes and/or height width ratios?





- For any point:

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

should hold.

$$xv = xv_{min} + (xw - xw_{min})\frac{(xv_{max} - xv_{min})}{xw_{max} - xw_{min}} \qquad s_x = \frac{(xv_{max} - xv_{min})}{xw_{max} - xw_{min}}$$

$$yv = yv_{min} + (yw - yw_{min})\frac{(yv_{max} - yv_{min})}{yw_{max} - yw_{min}} \qquad s_y = \frac{(yv_{max} - yv_{min})}{yw_{max} - yw_{min}}$$

- Scaling over the fixed point:

  $$\boldsymbol{S}(xw_{min}, yw_{min}, s_x, s_y)$$

- Translate window minimum to viewing minimum

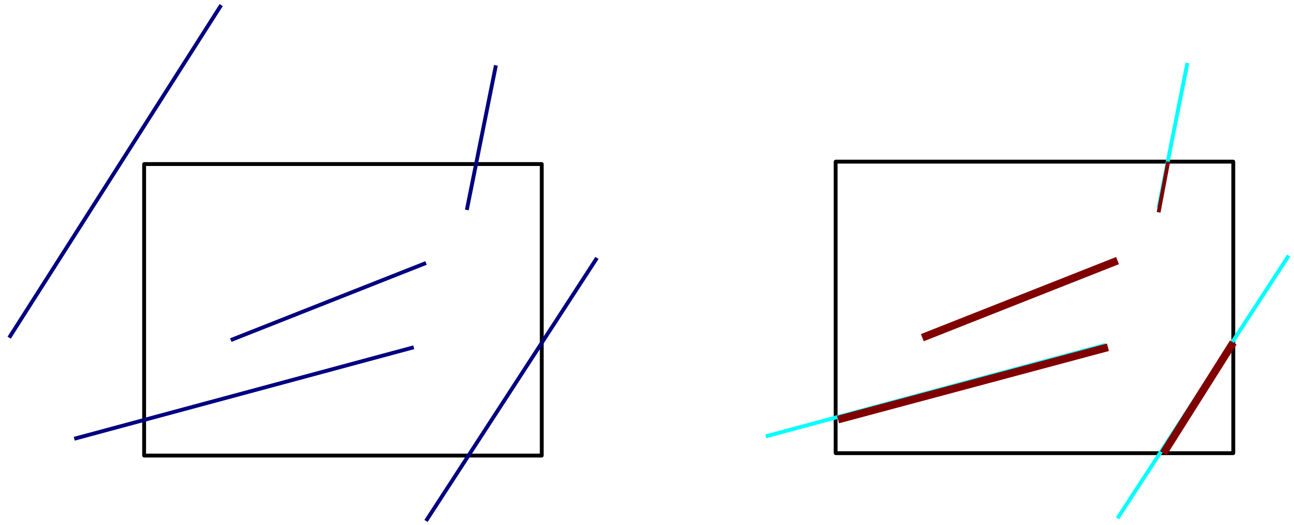
  $$\boldsymbol{T}(xv_{min} - xw_{min}, yv_{min} - yw_{min})$$

Clipping

- Clipping: identifying the parts of the objects that will be inside of the window.
- Scan converting all pixels and ignoring the pixels outside of the window is the easiest way. Is it a brilliant choice?
- Point clipping:

$$xw_{min} \leqslant x \leqslant xw_{max}$$
$$yw_{min} \leqslant y \leqslant yw_{max}$$

Line Clipping

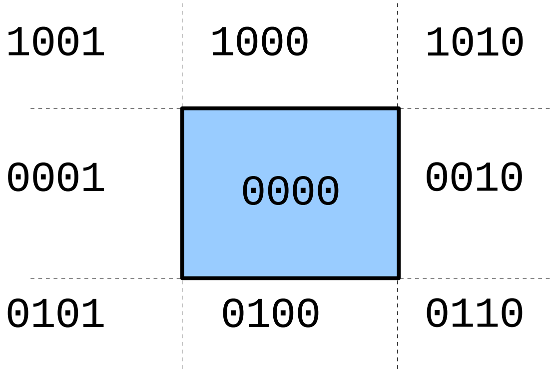


- For each line determine the intersection with boundaries. Parametric line equations:

$$x = x_1 + u(x_2 - x_1),$$
$$y = y_1 + u(y_2 - y_1), \qquad 0 \leqslant u \leqslant 1$$

Find u for all boundary lines. Expensive.

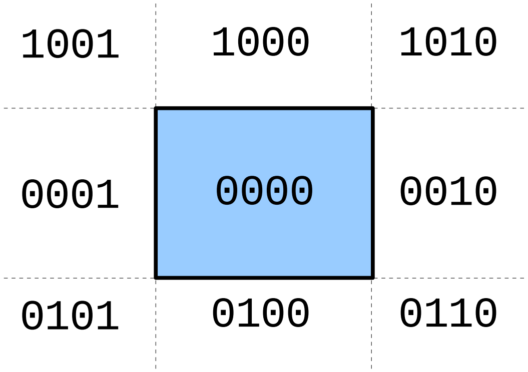
- Cohen-Shutherland Line Clipping:
  Based on determination of completely invisible line segments.
- Test bits for the point:
  bit 1: left of the left border
  bit 2: right of the right border
  bit 3: below the bottom border
  bit 4: above the top border



- When bits are defined for start and end point of the line segment, a single bitwise operation defines the visibility of the line segment. How?

- ```
  #define bits(x,y) ((x<xmin)|(x>xmax)<<1|(y<ymin)<<2|(y>ymax)<<3)

  b1=bits(x1,y1) ; b2=bits(x2,y2);
  if (b1 == 0 && b2==0) {
      /* both end points inside
         trivial accept          */
  } else if (b1 & b2) {
      /* line is completely outside
         ignore it               */
  } else {
      /* needs further calculation*/
  }
  ```



- For all borders find the transition from outside to inside
- Use the bits calculated before for outside tests
  ```

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

repeat

for $border = \{$*LEFT, RIGHT, BOTTOM, TOP*$\}$

if $\neg(bits(x_{1,} y_1) \vee bits(x_{2,} y_2))$

Both inside, terminate accept

else if $bits(x_{1,} y_1) \wedge bits(x_{2,} y_2)$

Same region, terminate reject

if $border = LEFT \vee border = RIGHT$

$$y_p = y_1 + m(x_{border} - x_1),\ x_p = x_{border}$$

else

$$x_p = x_1 + \frac{(y_{border} - y_1)}{m},\ y_p = y_{border}$$

if $bordertest(x_{1,} y_1)$

$$x_1 = x_p;\ y_1 = y_p$$

else if $bordertest(x_{2,} y_2)$

$$x_2 = x_p;\ y_2 = y_p$$

- Example:

$$bits(0,0)=0101 \qquad bits(8,5)=0010$$

$$bits(0,0)\wedge bits(8,5)=0000$$

$$m=\frac{(5-0)}{(8-0)}=\frac{5}{8}$$

LEFT: $y_p=y_1+\frac{5}{8}(1-0)=\frac{5}{8}$

$$LEFT(P_1)\rightarrow P_1{}'=(1,5/8)$$

RIGHT: $y_p=y_1+\frac{5}{8}(7-1)=\frac{5}{8}+\frac{5}{8}6=\frac{35}{8}$

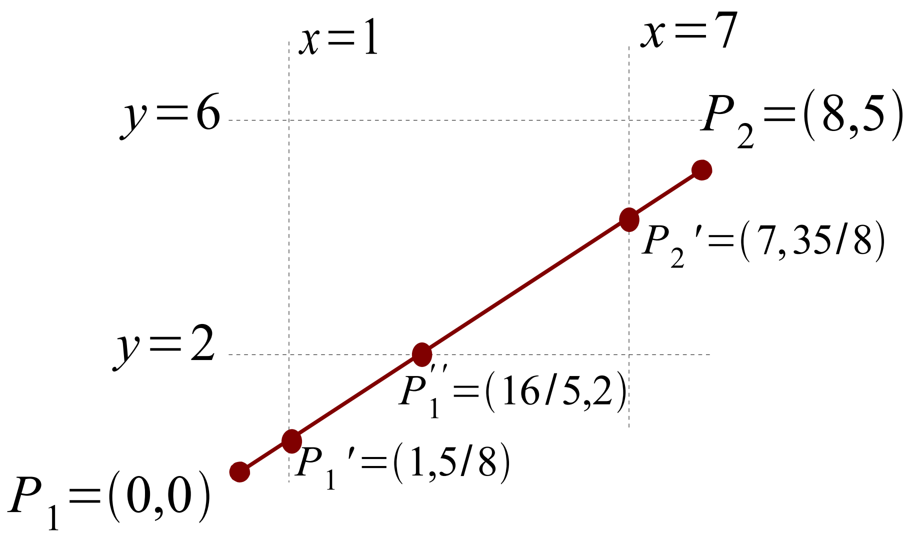
$$RIGHT(P_2)\rightarrow P_2{}'=(7,35/8)$$

BOTTOM: $x_p=1+(2-\frac{5}{8})\frac{8}{5}=1+\frac{16}{5}-1=\frac{16}{5}$

$$BOTTOM(P_1{}')\rightarrow P_1{}''=(16/5,2)$$

$P_1{}''$ *inside,* $P_2{}'$ *inside, so terminate*

$$L=(16/5,2) \text{ to } (7,35/8)$$



- Liang-Barsky Line clipping
  - accelerate evaluation of parametric equations

$$xw_{min} \leqslant x_1 + u\,\Delta x \leqslant xw_{max}$$
$$yw_{min} \leqslant y_1 + u\,\Delta y \leqslant yw_{max}, \qquad 0 \leqslant u \leqslant 1$$

rewite these inequalities in the form:

$$u\,p_k \leqslant q_k, \qquad k = 1, 2, 3, 4$$

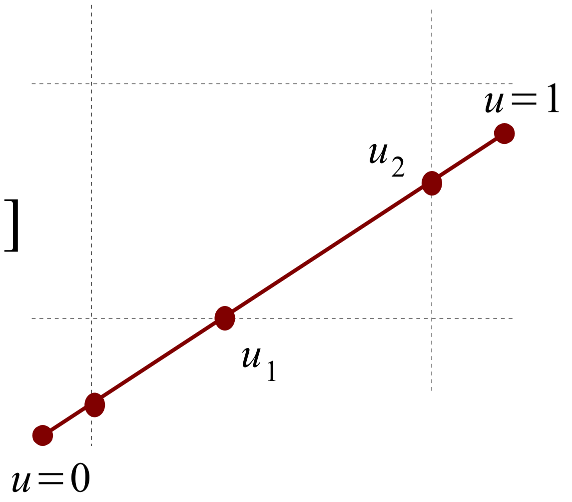
$$p_1 = -\Delta x, \quad q_1 = x_1 - xw_{min}$$
$$p_2 = \Delta x, \quad q_2 = xw_{max} - x_1$$
$$p_3 = -\Delta y, \quad q_3 = y_1 - yw_{min}$$
$$p_4 = \Delta y, \quad q_4 = yw_{max} - y_1$$

$$u = \frac{q_k}{p_k}$$

$$u\, p_k \leqslant q_k, \qquad 0 \leqslant u \leqslant 1$$

if $p_k < 0$, *outside to inside transition, candidate for* u_1

if $p_k > 0$, *inside to outside transition, candidate for* u_2



- Problem: find the clipping interval $[u_1, u_2]$
- If $p_k<0$ u_1 = max($u_1, p_k/q_k$)
  If $p_k>0$ u_2 = min($u_2, p_k/q_k$)

$u_1=0, u_2=1$

for $k=1,2,3,4$

find p_k

if $p_k<0$ $\quad r_k=\frac{q_k}{p_k}$

$u_1=max(u_{1,} r_k)$

else if $p_k>0$ $\quad r_k=\frac{q_k}{p_k}$

$u_2=min(u_{2,} r_k)$

else *parallel line, treat specially*

if $u_1>u_2$

line is outside, totally reject

$x_2=x_1+u_2 \Delta x, y_2=y_1+u_2 \Delta y$

$x_1=x_1+u_1 \Delta x, y_1=y_1+u_1 \Delta y$

- Example:

$\Delta x=8, \Delta y=5$

$k=1$

$$p_1=-8,\ \ q_1=0-1=-1,\quad r_1=\frac{1}{8}$$

$$u_1=max\{\frac{1}{8},0\}=\frac{1}{8}$$

$k=2$

$$p_2=8,\ \ q_2=7-0=7,\quad r_2=\frac{7}{8}$$

$$u_2=min\{\frac{7}{8},1\}=\frac{7}{8}$$

$k=3$

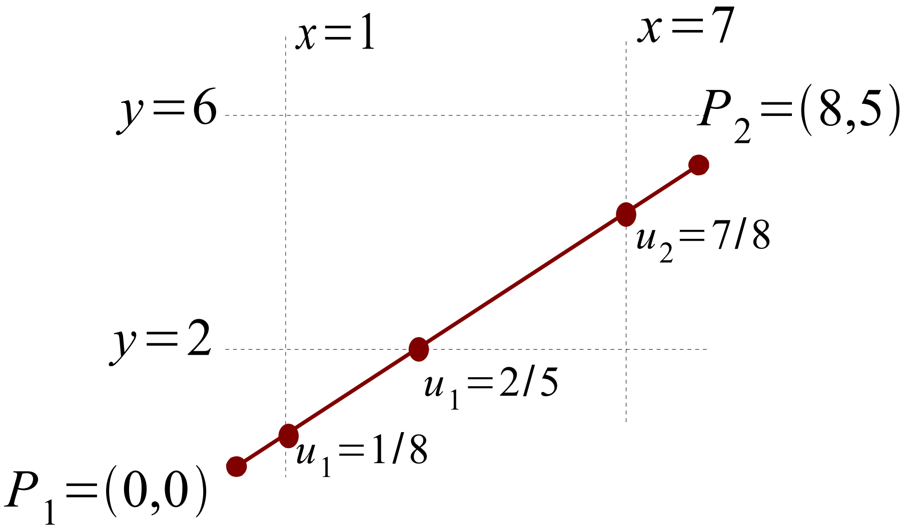
$$p_3=-5,\ \ q_3=0-2=-2,\quad r_3=\frac{-2}{-5}$$

$$u_1=max\{\frac{1}{8},\frac{2}{5}\}=\frac{2}{5}$$

$k=4$

$$p_4=5,\ \ q_4=6-0=6,\quad r_4=\frac{6}{5}$$

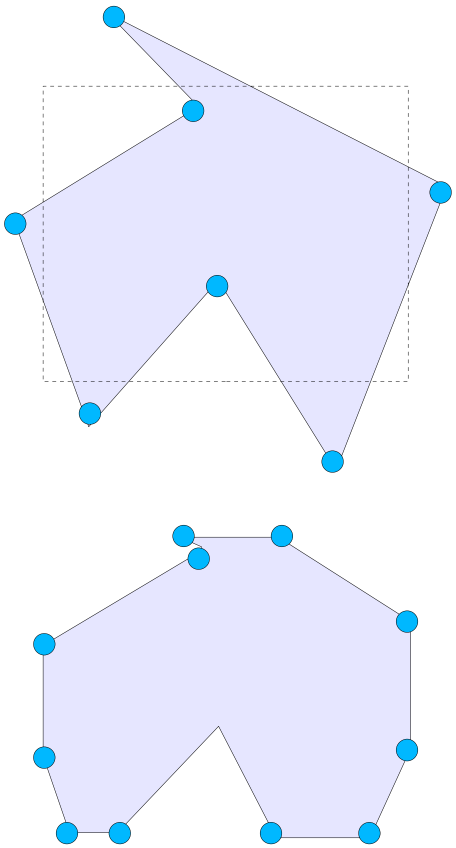
$$u_2=min\{\frac{6}{5},\frac{7}{8}\}=\frac{7}{8}$$



$$P_1=(0+2/5\cdot 8,0+2/5\cdot 5)=(16/5,2)\qquad P_2=(0+7/8\cdot 8,0+7/8\cdot 5)=(7,35/8)$$

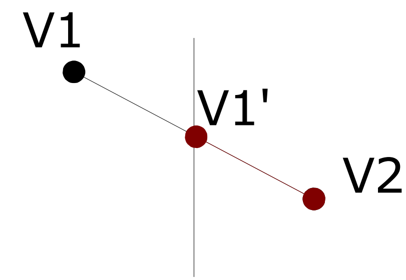
Polygon Clipping

- Find the vertices of the new polygon(s) inside the window.
- Sutherland-Hodgeman Polygon Clipping:
  Check each edge of the polygon against all window boundaries.
  Modify the vertices based on transitions.

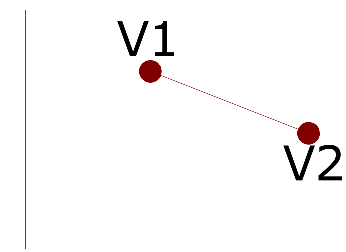


Shutherland-Hodgeman Polygon Clipping

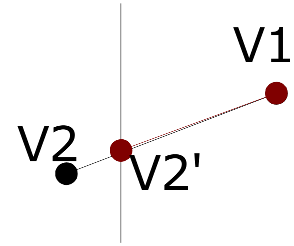
- Traverse edges for borders; 4 cases:
  - V1 outside, V2 inside: take V1' and V2



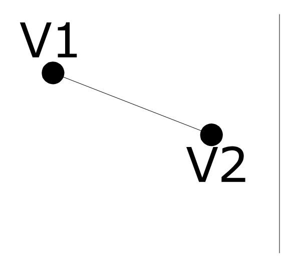
  - V1 inside, V2 inside: take V1 and V2

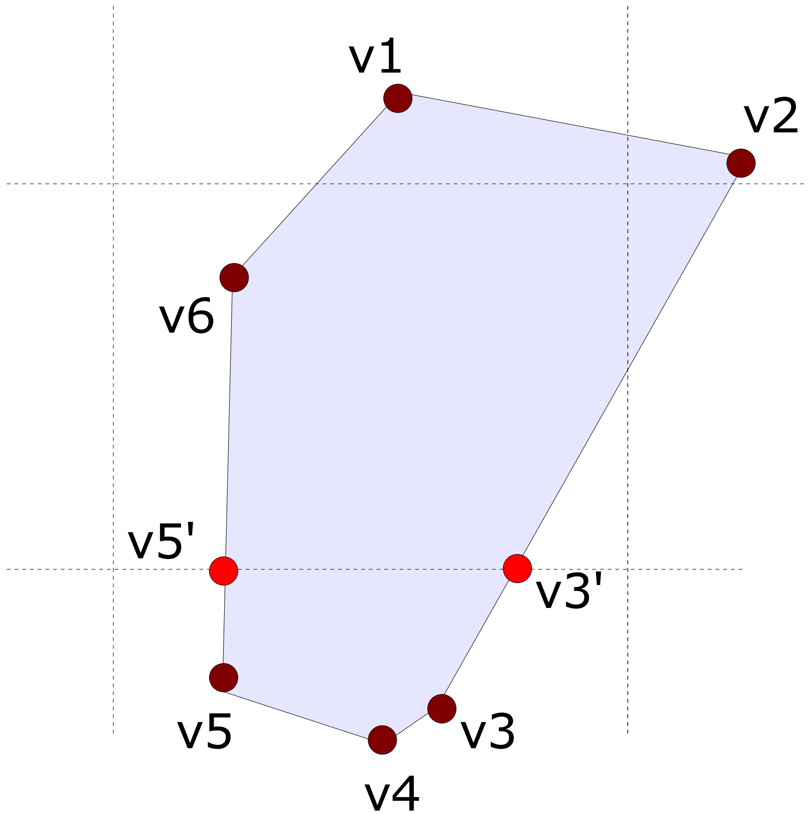


  - V1 inside, V2 outside: take V1 and V2'



  - V1 outside, V2 outside: take none





- Left border:

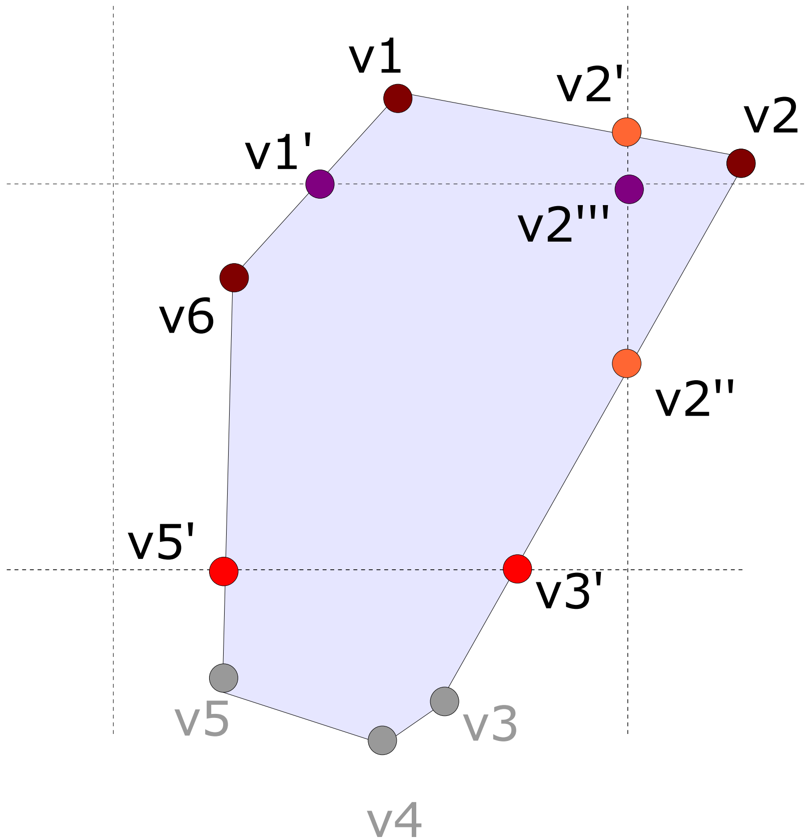
v1 v2	both inside	v1 v2
v2 v3	both inside	v2 v3
.....	" "	....

  v1,v2,v3,v4v5,v6,v1

- Bottom Border:

v1 v2	both inside	v1 v2
v2 v3	v2 i, v3 o	v2 v3'
v3 v4	both outside	none
v4 v5	both outside	none
v5 v6	v5 o, v6 i	v5' v6
v6 v1	both inside	v6 v1

  v1,v2,v3',v5',v6,v1



- v1,v2,v3',v5',v6,v1
- Right border:

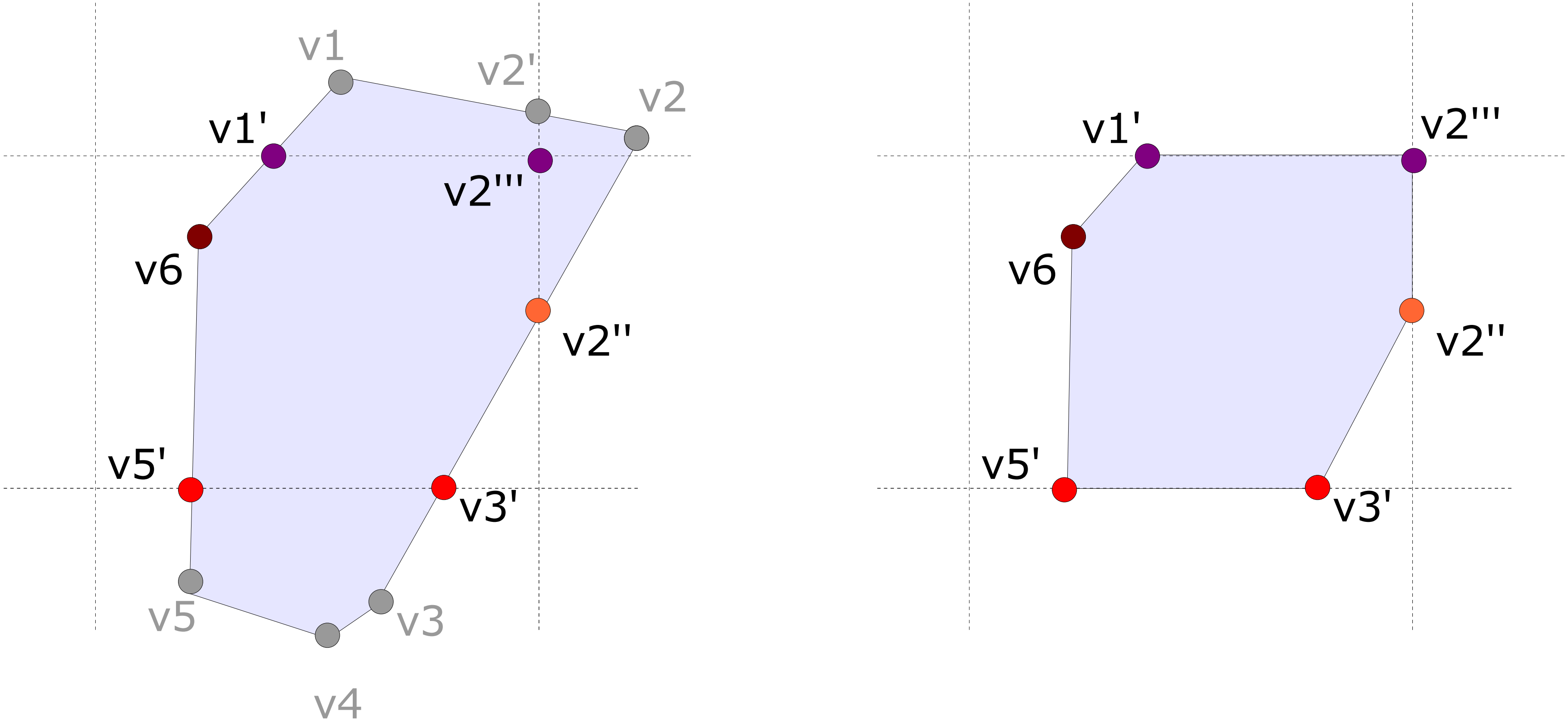
v1 v2	v1 i, v2 o	v1 v2'
v2 v3'	v2 o, v3'i	v2'' v3'
v3' v5'	both inside	v3' v5'
v5' v6	both inside	v5' v6
v6 v1	both inside	v6 v1

v1,v2',v2'',v3',v5',v6,v1

- Top Border:

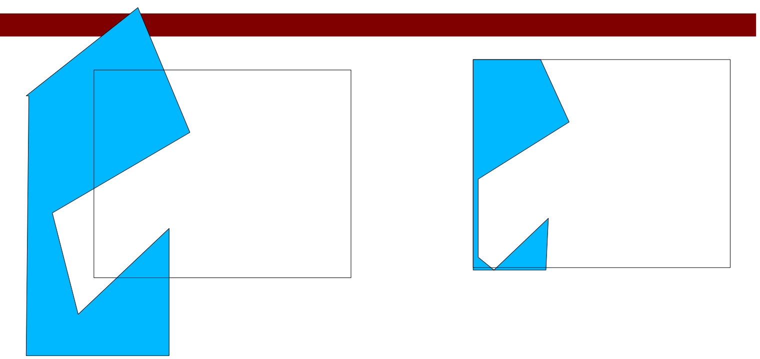
v1 v2'	both outside	none
v2' v2''	v2' o, v2'' i	v2''' v2''
v2'' v3'	both inside	v2'' v3'
v3' v5'	both inside	v3' v5'
v5' v6	both inside	v5' v6
v6 v1	v6 i, v1 o	v6 v1'

v2''',v2'',v3',v5',v6,v1'


v1
v2'
v2
v1'
v2'''
v6
v2''
v5'
v3'
v5
v3
v4
v1'
v2'''
v6
v2''
v5'
v3'

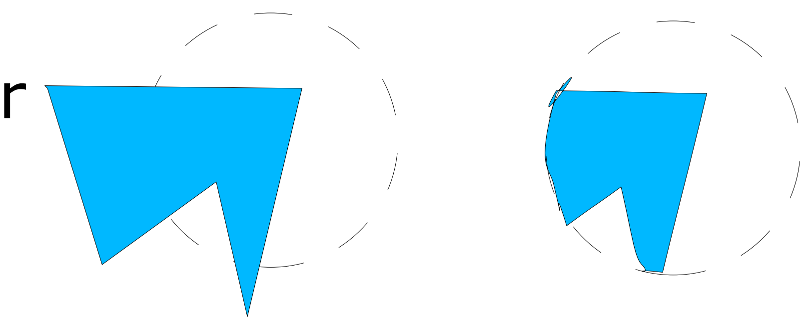
Other Issues in Clipping

- Problem in Shutherland-Hodges. Weiler-Atherton has a solution



- Clipping other shapes: Circle, Ellipse, Curves.

- Clipping a shape against another shape



- Clipping the exteriors.