Lecture 3: 11 September 1997

Administrative:

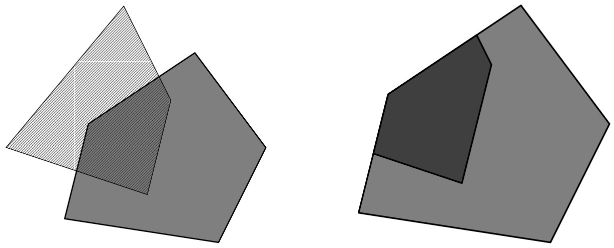
Web signup due tomorrow, 5pm
  Required for registration!
Lecture notes now online & printable
Asst 1 due tomorrow, 5pm
  First octant lines only !
  PC's available in 38-344, 371955*
  All hours, except M-F 2-5, T 5-7
Asst 2 due one week from tomorrow
  Analogous to Asst 1; posted online
TA Office Hours:
  Prof. T.: T 4-on (NE43 208)
  David T 4-6 (NE43 2^{nd} floor)
  Jon W 12-on, **R 4-on** (NE43 2^{nd} floor)
  Luke W 2-4 (NE43 2^{nd} floor)

Today:

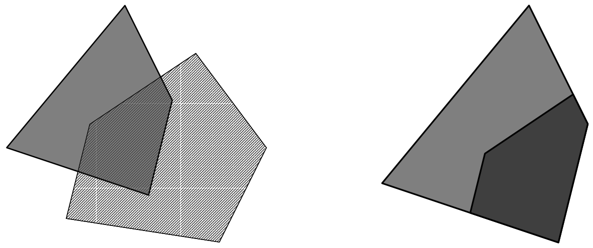
Clipping filled 2D polygons (H&B S6.8)
2D coordinate transformations (H&B S5.1 – 5.6)

Clipping Filled Polygons

Clip "subject polygon" to "clip polygon"

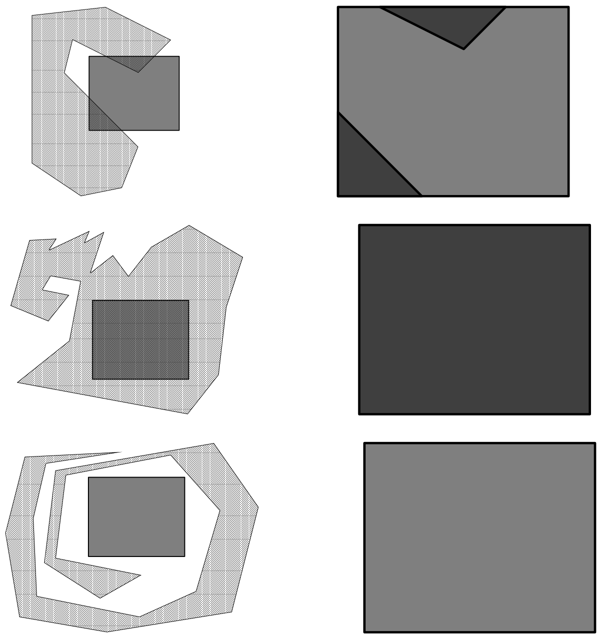


Roles interchangeable; really computing *intersection*:



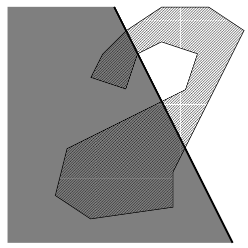
Clipping Challenges

Many complications even for convex clip regions:

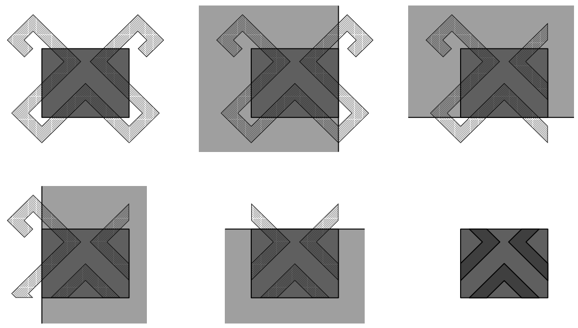


Sutherland-Hodgman Algorithm

Clips subject polygon to any convex polygon
Idea: employ half-plane clipping as primitive:

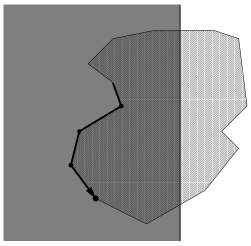


Simply clip to each bounding half-plane in turn:

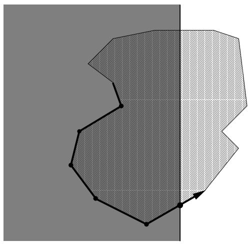


Sutherland-Hodgman Algorithm

Input: ordered list of subject polygon vertices
"Walk" around polygon, processing each edge in turn
Produces output for each edge **AB** (0, 1, or 2 vertices).
1) Edge entirely in half-plane: output **B**

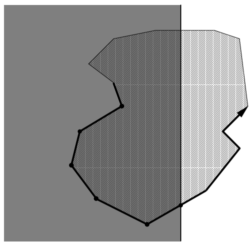


2) **A** inside, **B outside**: output exit point

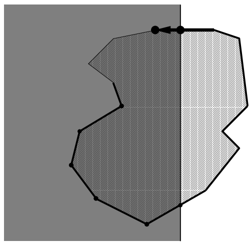


Sutherland-Hodgman Algorithm: Cases

3) **A** outside, **B outside**: output nothing



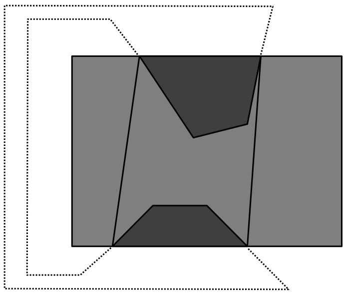
4) **A** outside, **B inside**:
output entry point, then **B** (in order)



Is the algorithm complete?

Sutherland-Hodgman Problem

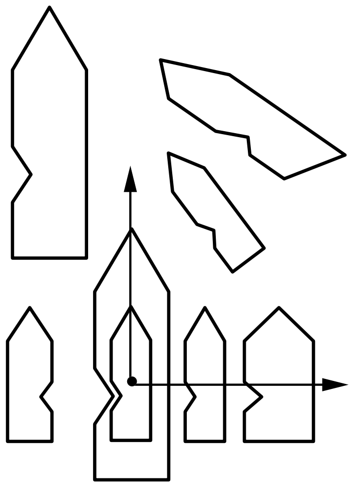
Algorithm cannot represent disconnected output!



Remove by postprocessing, or...
Handle in scan conversion
Implementation notes:
How to handle input, clipped vertices?

2D Coordinate Transformations

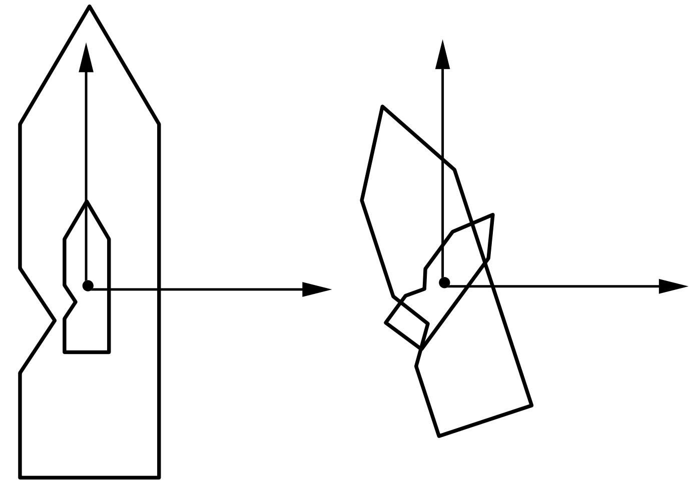
Transformation: operator defined on geometric object
Abstractly: $\mathbf{M}$ maps points $\mathbf{p} = (\mathbf{x}, \mathbf{y})$ to
points $\mathbf{M}(\mathbf{p}) = \mathbf{p'} = (x', y')$



Simplitudes

Rotation, uniform (pos.) scaling, or any combination

Preserves angles, ratios of lengths

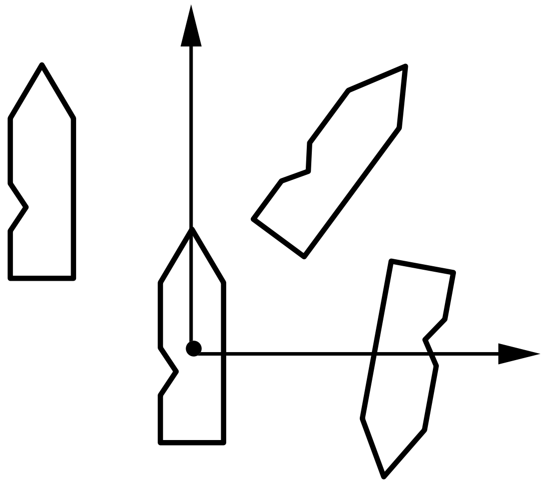


DOFs: 1 of rotation, 1 of scaling

Rigid Body

Rigid-Body: translation, rotation (or any combination)

Preserves $\boxed{\quad\quad}$ and $\boxed{\quad\quad}$

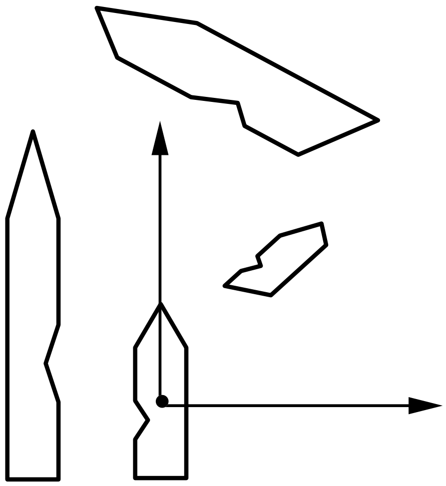


DOFs: 1 of rotation, 2 of translation

Affine

Affine: rotation, scaling, reflection, shear, translation, or any combination of these (linear mappings plus a translation)

Preserves $\boxed{\quad\quad\quad}$



DOFs: 1 rot, 2 trans, 3 skew

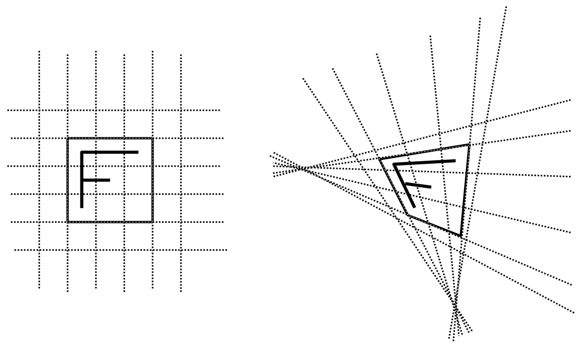
Projective

Projective: perspective projection

Aka *homogeneous* or *rational linear* mapping

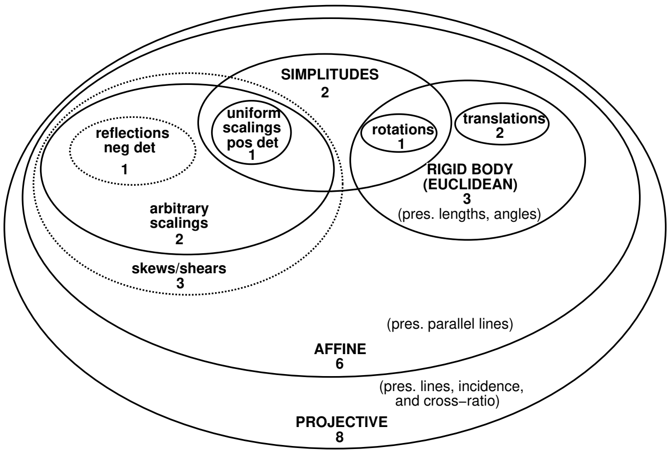
Preserves $\boxed{\quad\quad}$ and $\boxed{\quad\quad}$

(most general line-preserving transformation)



DOFs: 1 rot, 2 trans, 3 skew, 2 projective

Transformation Families



Matrix Representation

Traditional: Use a 2×2 matrix for 2D transformations:

Identity:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Rotation about origin by θ

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Scaling about origin by s_x, s_y:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Shearing parallel to x axis, by factor s_x:

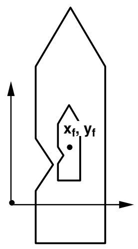
What points are left in place?

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & s_x \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Application Other Than at Origin

Example: Scaling w.r.t **fixed point** x_f, y_f:

$$\begin{aligned} x' &= x_f + s_x(x - x_f) \\ y' &= y_f + s_y(y - y_f) \end{aligned}$$



Rearrange to collect constant terms:

$$\begin{aligned} x' &= x \cdot s_x + x_f(1 - s_x) \\ y' &= y \cdot s_y + y_f(1 - s_y) \end{aligned}$$

Then write as sum:

$$\mathbf{P'} = \mathbf{M} \cdot \mathbf{p} + \mathbf{C}$$

Analogous strategies for rotation, shearing;
All expressible as $(\mathbf{M}, \mathbf{C})$

Composition

How to apply *sequence* of transformations?

Given $(\mathbf{M}, \mathbf{C})_\mathbf{1} .. (\mathbf{M}, \mathbf{C})_\mathbf{N}$

Compute each result in turn:

$$\begin{aligned} \mathbf{p_1} &= M_1 \cdot p + C_1 \\ \mathbf{p_2} &= M_2 \cdot p_1 + C_2 \\ &\ldots \\ \mathbf{p_N} &= M_N \cdot p_{N-1} + C_N \end{aligned}$$

But: translation inexpressible as 2×2 matrix:

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

How can we represent translations in matrix form?

Notational Extension

Scheme allows matrix representation for translation:

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

And in combination with other transformations:

$$\begin{aligned} x' &= Ax + By + t_x \\ y' &= Cx + Dy + t_y \end{aligned}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} A & B & t_x \\ C & D & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

At the cost of extra storage, manipulation
(Later, we'll see other uses, formalism)

Consistent Representation

Rotation about origin by θ

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Scaling about origin by s_x, s_y:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Shearing parallel to x axis, by factor s_x:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & s_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Translation:

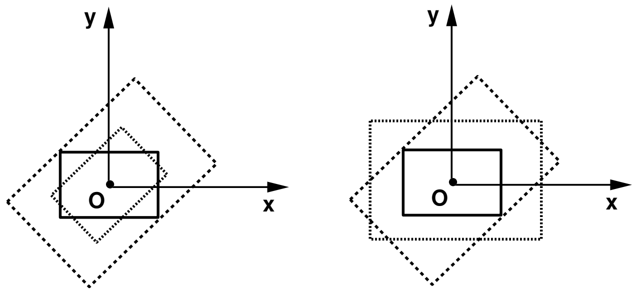
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Composition of Transformations

How to apply sequence of transformations $\mathbf{A}, \mathbf{B}, \mathbf{C}$, etc.?

Easy: since $\mathbf{p}' = \mathbf{M} \cdot \mathbf{p}$,

$$\begin{aligned} \mathbf{p}' &= A \cdot (B \cdot (C \cdot p))) \\ &= A \cdot (B \cdot (C \cdot p))) \\ &= (ABC) \cdot p \text{ (by associativity)} \\ &= M \cdot p \end{aligned}$$



Rotate 45°, then scale 2: $\mathbf{M} = \mathbf{SRp}$
Scale 2, then rotate 45°: $\mathbf{M} = \mathbf{RSp}$

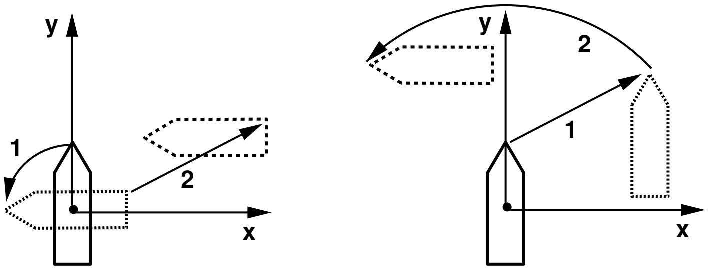
Commutative Transformations

Same result. These transformations *commute*:

The identity, with anything (of course)
Any sequence of scalings only
Any sequence of rotations only
Any sequence of translations only
Any rotation, any *uniform* scaling

Non-commutative Transformations

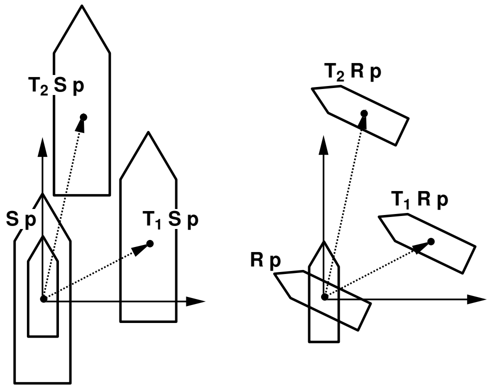
Sad fact of life: most transformations don't commute
Example: translation and rotation



Rotate 90°, then translate t_x: $\mathbf{M} = \mathbf{TRp}$
Translate t_x, then rotate 90°: $\mathbf{M} = \mathbf{RTp}$
Something weird: why *say*, "Rotate, then Translate"
But *write* in other order? $\mathbf{M} = \mathbf{TRp}$

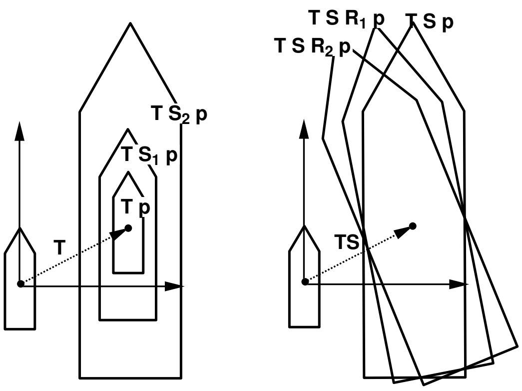
World Space Transformation

$\mathbf{T}\ \mathbf{R}\ \mathbf{p} = \mathbf{T}\ (\mathbf{R}\ \mathbf{p})$; $\mathbf{T}$ acting on rotated $\mathbf{p}$
Think of $\mathbf{M}$ as a *string* of transformations, $\mathbf{p}' = \mathbf{Mp}$
Now apply $\mathbf{N}$ to the *left* of $\mathbf{M}$: $\mathbf{p}'' = \mathbf{NMp} = \mathbf{N}(\mathbf{Mp})$
We say that $\mathbf{N}$ is applied in "world space."



Object Space Transformation

Apply $\mathbf{N}$ to the *right* of $\mathbf{M}$: $\mathbf{p}'' = \mathbf{MNp} = M(\mathbf{Np})$
We say that $\mathbf{N}$ is applied in "object space."
This makes sense – it's the first thing to "act upon" $\mathbf{p}$
So it must happen in $\mathbf{p}$'s local coordinate system



More Composite Transformations

Scale about fixed point F:
Translate F to origin
Apply scaling
Translate origin back to F
Reflect about line $y = x$:
Rotate counter-clockwise 45°
Reflect about x axis
Rotate clockwise 45°
M = Rot · Refl · Rot
Reflect about line $Ax + By + C = 0$:
Translate line to origin (how?)
Rotate down to x axis
Reflect about x axis
Rotate x axis back to line
Translate line back (how?)
M = Trans · Rot · Refl · Rot · Trans