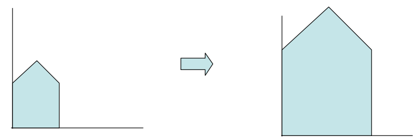
Transformation examples in 2D

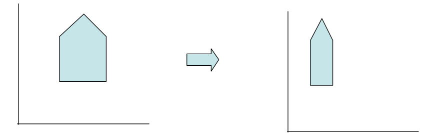
- Scale (stretch) by a factor of k



$$M = \begin{vmatrix} k & 0 \\ 0 & k \end{vmatrix}$$ (k = 2 in the example)

Transformation examples in 2D

- Scale by a factor of (S_x, S_y)



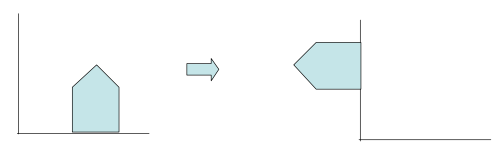
$$M = \begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix}$$ (Above, $S_x = 1/2$, $S_y = 1$)

Orthogonal Transformations

- Orthogonal transformations are defined by $O^TO=I$
- Also have $|\det(O)|=1$
- Rigid body rotations and flips

Transformation examples in 2D

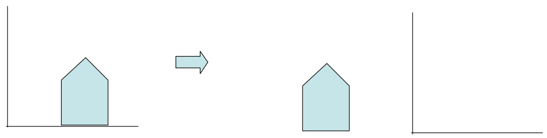
- Rotate around origin by θ (Orthogonal)



$$M = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$$ (Above, $\theta=90^o$)

Transformation examples in 2D

- Flip over y axis (Orthogonal)

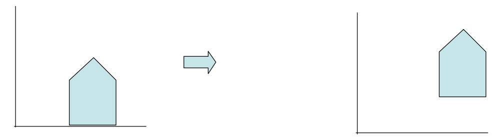


$$M = \begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$$

Flip over x axis is ?

2D Transformations

- Translation ($\mathbf{P}_{new} = \mathbf{P} + \mathbf{T}$)



M =?

Homogenous Coordinates

- Represent 2D points by 3D vectors
- (x,y)-->(x,y,1)
- Now a multitude of 3D points (x,y,W) represent the same 2D point, (x/W, y/W, 1)
- Represent 2D transforms with 3 by 3 matrices
- Can now represent translations by matrix multiplications

2D Scale in H.C.

$$M = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

2D Rotation in H.C.

$$M = \begin{vmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

2D Translation in H.C.

- $\mathbf{P}_{new} = \mathbf{P} + \mathbf{T}$
- $(x', y') = (x, y) + (t_x, t_y)$

$$M = \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix}$$

Transformations in 3D

- Homogeneous coordinates now have four components - traditionally, (x, y, z, w)
  - ordinary to homogeneous: (x, y, z) -> (x, y, z, 1)
  - homogeneous to ordinary: (x, y, z, w) -> (x/w, y/w, z/w)
- Again, translation can be expressed as a multiplication.

Transformation examples in 3D

- Translation:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Anisotropic scaling:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformation examples in 3D

- Rotation about x-axis:

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

- See supplementary material for rotation about an arbitrary axis.
- A rotation matrix can be thought of as either a rotation about an axis, or a rigid transformation represented by an orthogonal matrix

Example

- Rewrite world coordinates as camera centric coordinates
- Problem is that the origins are not the same and the axis are not aligned.
- Note that our rotation matrices are about an axis.
- Hence we need to translate the world coordinates, and then rotate them.

Transform object expressed in world coords to camera coords

Step 1. Translate the camera at O_C to the world origin. Call this T_1.

Translation vector is simply negative O_C.

(We are changing the coordinate system of the world, which is the same thing mathematically as moving the camera. We want object world coordinates to **change** so that the camera location **becomes** the origin).

Transform object expressed in world coords to camera coords

Step 2. Rotate camera coordinate frame (in w.c.) so that so that **u** is **x**, **v** is **y**, and **n** is **z**. The matrix is ?

(We are changing the coordinate system of the world, which is the same thing mathematically as moving the camera. We want object world coordinates to **change** so that the camera axis **becomes** the standard axis—e.g, **u** becomes (1,0,0), **v** becomes (0,1,0) and **n** becomes (0,0,1)).

Transform object expressed in world coords to camera coords

Step 2. Rotate camera coordinate frame (in w.c.) so that so that **u** is **x**, **v** is **y**, and **n** is **z**. The matrix is:

$$\begin{bmatrix} \mathbf{u}^T & 0 \\ \mathbf{v}^T & 0 \\ \mathbf{n}^T & 0 \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

(why?)

Transform object expressed in world coords to camera coords

$$\begin{bmatrix} \mathbf{u}^T & 0 \\ \mathbf{v}^T & 0 \\ \mathbf{n}^T & 0 \\ 0\ 0\ 0 & 1 \end{bmatrix} \mathbf{u} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

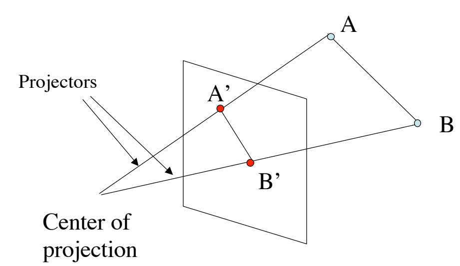
In the current coords (world shifted so that VPR is at origin): **u** maps into the X-axis unit vector (1,0,0,0) which is what we want.

(Similarly, **v**-->Y-axis unit vector, **n**-->Z-axis unit vector)

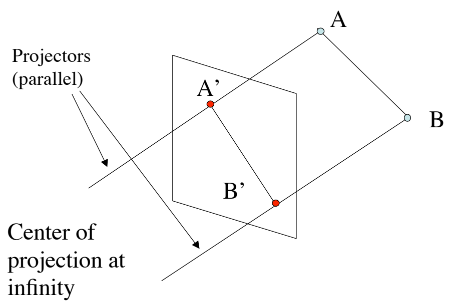
Projections

- Want to think about geometric image formation as a mathematical transformation taking points in the 3D world and mapping them into an image plane.
- Mathematical definition of a projection: PP=P
- (Doing it a second time has no effect).
- Generally rank deficient (non-invertable)--exception is P=I
- Transformation looses information (e.g., depth)
- Given a 2D image, there are many 3D worlds that could have lead to it.

Projections



Parallel Projection

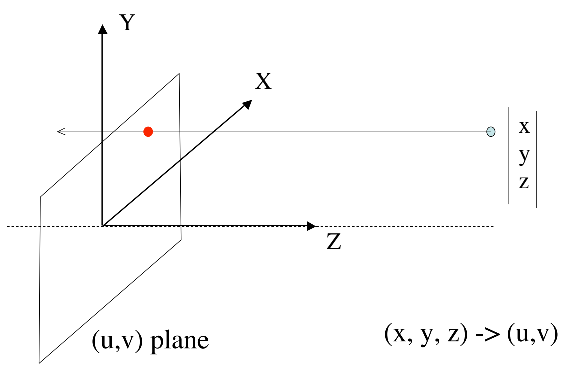


Parallel Projection

Parallel lines remain parallel, some 3D measurements can be made using 2D picture

If projection plane is perpendicular to projectors the projection is orthographic

Orthographic example (onto z=0)



The equation of projection (orthographic, onto z=0)

- In homogeneous coordinates

  $$(x,\ y,\ z,\ 1) \Rightarrow (x,\ y,\ 1)$$

- Graphics course survivors: You will notice slight changes in style to be consistent with the book. Perhaps most notably we will explicitly, rather than implicitly, ignore the third projected coordinate, so projection matrices will be 3 by 4 not 4 by 4.

The projection matrix

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