Lecture 3
Transformation and GLUT Setup

Xin (Shane) Li

Sep. 1, 2009

http://www.ece.lsu.edu/xinli/teaching/EE4700Fall2009.htm

Last class

- Shape Representation Overview (cont.)
  - Quadtree, octree rep.
  - Skeleton rep.
  - GC rep.
- Half-Edge data structure
  - Why we need it – efficient local traversal, storage
  - Data Structure
  - Using Half-Edge data structure in a few examples
  - Resources
  - ➢ We will see a few more examples on using half-edge data structure in the future (and show more its usages in codes).

Today

1. Write your first OpenGL program

2. Points, Vectors, and Transformations

OpenGL Setup

1). Check: http://www.ece.lsu.edu/xinli/OpenGL/GLUTSetup.htm to download the precompiled libraries you need.

2). Download the "HelloWorld" program from: http://www.ece.lsu.edu/xinli/OpenGL/program1.cpp

3). Create a Win32 console project, include this "program1.cpp", then compile and run it.

4). If you get linker errors or run-time errors, your system environment might not be compatible with the precompiled libraries. You might need to go back to 1) and download the source codes, compile them in your system. Then use the libraries (glut.h, glut32.lib, glut32.dll) newly generated.

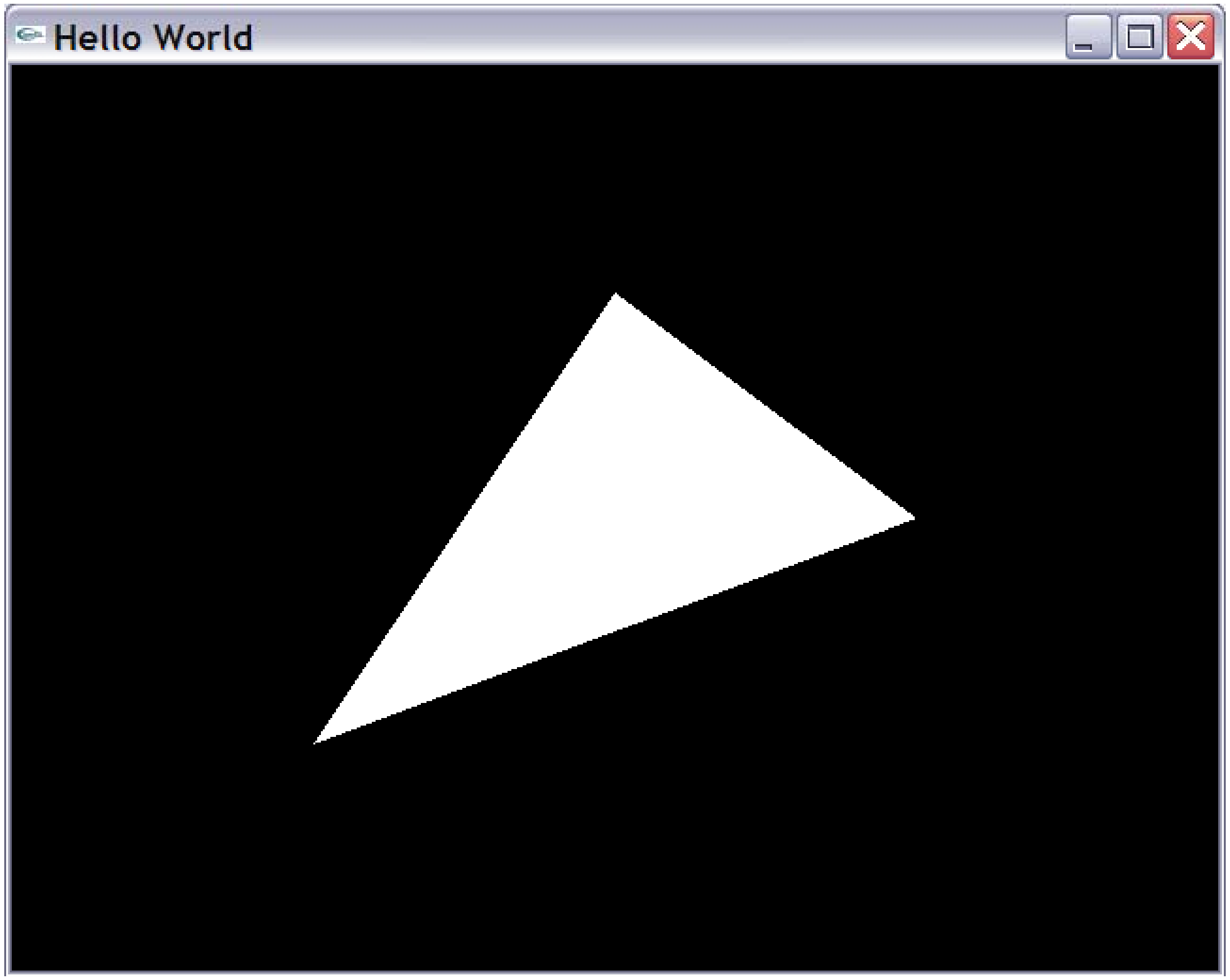
Now let's go through "program1.cpp"

Hello World

Go through program1.cpp line by line…

Run it.

You should see something like this:



Resources

We will keep talking about transformation so late later we can make the display interactively.

There are many online resources about OpenGL:

1. The OpenGL official website http://www.opengl.org/
find coding resources, documentation, tutorials…
2. Nate Robins OpenGL website:
http://www.xmission.com/~nate/index.html
3. OpenGL Tutorials at NeHe
http://nehe.gamedev.net/
4. And so on…

1.1 Points and Vectors

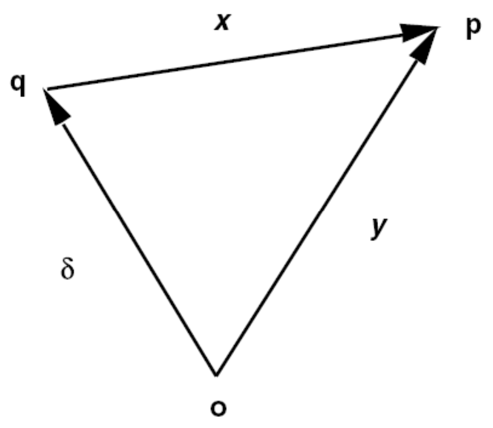
- Points
  - ❑ A solid object with infinitely small size → a mathematical abstraction
  - ❑ A set whose elements, called points, satisfying certain axioms → a Euclidean space
  - ❑ Use points to define locations, to describe trajectories of objects, and model geometric shapes…
- Vectors
  - Length (magnitude) + direction (e.g. velocity)
  - Add two vectors → parallelogram rule of analytical geometry
  - Multiply a vector by a scalar → change magnitude not direction
  - A set of elements (vectors), with two operations (addition and scalar multiplication) → a vector space
  - Vectors also abstract polynomials, splines, periodic functions, …

1.1 Points and Vectors

- Points and Vectors
  - Pick an origin o, each point p corresponds to a vector x

    (since each p and o defines a length and a direction)

    → simply define point difference as an operation that produces vectors
  - The correspondence depends on the origin
    - But, simple relationship between x and y
    - $y = x + \delta$, $\delta = q - o$
  - In geometric modeling, it is customary not to distinguish between points and vectors (e.g. point addition)
  - Basis, components
    - Can always select a minimal set $\{e_1, e_2, \ldots, e_n\}$, s.t. $x = x_1e_1+x_2e_2+\ldots+x_ne_n$



Change of origin

1.1 Points and Vectors

- Points and Vectors (cont.)
  - For a fixed basis E, there is a one-to-one correspondence between vectors {x} and the arrays of components {X}
  - Often write x = EX

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad E = \begin{bmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_2 & \dots & \boldsymbol{e}_n \end{bmatrix}$$

  - Matrix notation preserves rules of vector sum z=x+y and scalar multiplication z=ax :

$$Z = X + Y = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}. \qquad Z = \begin{bmatrix} ax_1 \\ ax_2 \\ \vdots \\ ax_n \end{bmatrix}$$

1.1 Points and Vectors

- Points and Vectors (cont.)
  - inner product (dot product): xy
  - norm (length) of vector: $\|x\| = \sqrt{x.x}$
  - a unit vector ← length equals unity
  - orthogonal vectors ← dot product is zero
  - orthonormal bases (unit vectors, pairwise orthogonal)
    - In such a basis, the inner product:

$$\boldsymbol{x}.\boldsymbol{y} = X^t Y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

and the length becomes:
$$\|\boldsymbol{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

1.2 frame

- Points correspond to Vectors, given a fixed origin
- Vectors correspond to column matrices, given a fixed basis

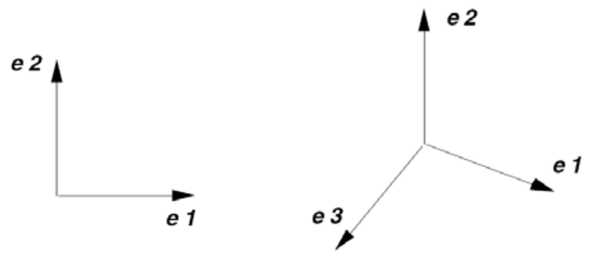
➔ Points correspond to column matrices

- A pair (origin, basis) is called a <u>frame</u>, or <u>coordinate system</u>
    - For a fixed frame, points → column matrices (elements of the matrix are called the coordinates of the point in that frame)
    - Correspondences between points, vectors, and column matrices → provide computational tools to represent and manipulate these entities (matrices are easy to represent and manipulate in programming)

1.3 cross product

- Useful especially in 3D, defined in a right-handed, orthonormal , 3-D basis:

$$\boldsymbol{x} \times \boldsymbol{y} = (x_2 y_3 - x_3 y_2)\boldsymbol{e}_1 + (x_3 y_1 - x_1 y_3)\boldsymbol{e}_2 + (x_1 y_2 - x_2 y_1)\boldsymbol{e}_3$$



$$\boldsymbol{e}_1 \times \boldsymbol{e}_2 = \boldsymbol{e}_3$$
$$\boldsymbol{e}_2 \times \boldsymbol{e}_3 = \boldsymbol{e}_1$$
$$\boldsymbol{e}_3 \times \boldsymbol{e}_1 = \boldsymbol{e}_2$$

- Cross product of two parallel vectors is zero
- Otherwise, its magnitude = the area of the parallelogram, its direction is perpendicular to both vectors
- For completing a 3D orthonormal basis when two of its vectors are known

2. Transformations

- Moving, scaling, and deforming objects are fundamental operations in geometric modeling.
- If objects are considered as sets of points, what we need are transformations that map points onto other points

…Start with some basic transformations…

Computing Transformation is important because we want:

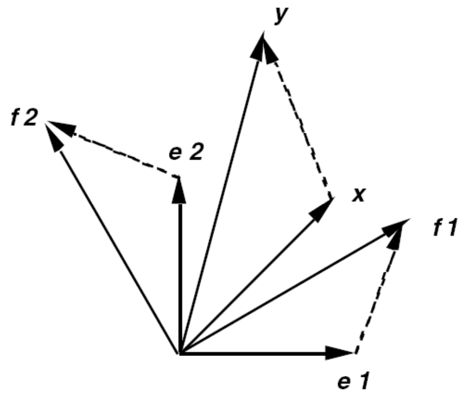
- To compute/represent transformations when necessary
  - For rendering, interactive visualization
  - Other visual computing applications
- To analyze shapes under transformation
  - Find invariant shape properties under various transformations (applications: shape descriptors for shape retrieval, object recognition, object tracking/localization…)

geometry is the study of **invariants** under **transformations**

2.1 Linear Transformations

- A transformation is linear if it distributes over linear combinations, i.e. $T(a\mathbf{x} + b\mathbf{y}) = a\mathbf{T}(\mathbf{x}) + b\mathbf{T}(\mathbf{y})$
- Suppose we have two bases $E = [\mathbf{e}_1 \quad \cdots \quad \mathbf{e}_n]$, $F = [f_1 \quad \cdots \quad f_n]$
- And we want to find the linear transformation:

$$\mathbf{T}_{ef}(\mathbf{e}_i) = f_i, \quad i = 1, \ldots, n.$$



- What is the effect of such a transformation on an arbitrary vector?

$$\mathbf{y} = \mathbf{T}_{ef}(\mathbf{x}) \qquad \mathbf{x} = EX^e$$

- Following the linearity definition:

$$\mathbf{y} = \mathbf{T}_{ef}(\mathbf{x}) = \left[\mathbf{T}_{ef}(\mathbf{e}_1) \quad \cdots \quad \mathbf{T}_{ef}(\mathbf{e}_n)\right] X^e = \left[f_1 \quad \cdots \quad f_n\right] X^e$$

2.1 Linear Transformations (cont.)

$$\boldsymbol{y} = \boldsymbol{T}_{ef}(\boldsymbol{x}) = \left[\boldsymbol{T}_{ef}(\boldsymbol{e}_1) \quad \cdots \quad \boldsymbol{T}_{ef}(\boldsymbol{e}_n)\right] X^e = [\boldsymbol{f}_1 \quad \cdots \quad \boldsymbol{f}_n] X^e$$

$\Rightarrow$ $$\boldsymbol{y} = \left[EF_1^e \quad \cdots \quad EF_n^e\right] X^e = E\left[F_1^e \quad \cdots \quad F_n^e\right] X^e$$

- Meaning the components of y in basis E are $M^e = \left[F_1^e \quad \cdots \quad F_n^e\right]$

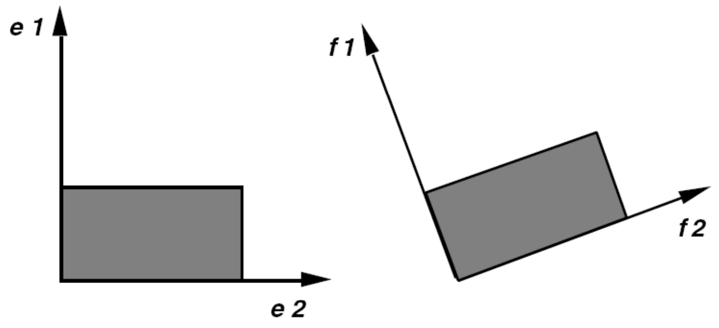
$$Y^e = M^e X^e$$

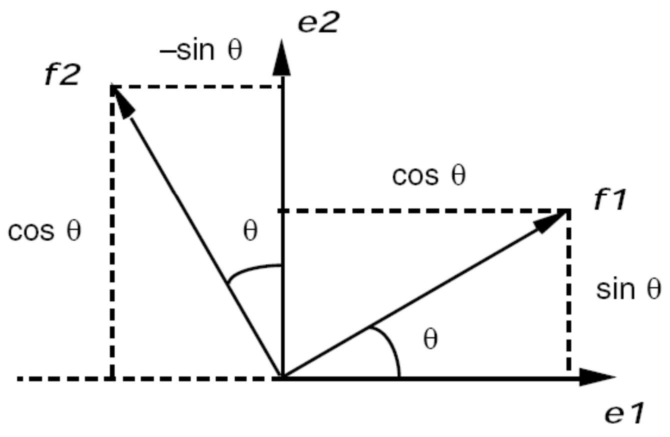
- These two equations show:
  - For a fixed basis E, <u>each transformation corresponds to a square matrix</u> (correspondence between linear transformation and square matrices)
  - They give us computational tools for evaluating transformation effect on a vector (simply multiply the corresponding matrix)
  - The way to construct this matrix mapping a basis to another basis is convenient

2.1 Linear Transformations (cont.)

Example.

- Coordinate frames are usually attached to objects, suppose we want to orient left rectangle such that it aligns with the right rectangle.





- Usually, just solve a linear system
- Here, based on elementary trigonometry (simple when it is just a planar rotation)

$$F_1^e = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \quad F_2^e = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

$$M^e = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

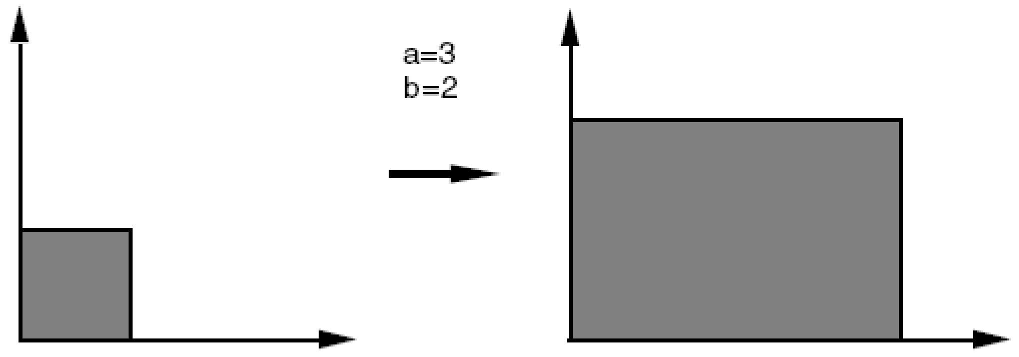
2.1 Specific Linear Transformations

- Several most common linear 2D transformations (results apply to 3D, we will see the modifications shortly)
  - Scaling
  - Rotation
  - Shear
  - Reflection -- scaling with negative factors
  - Orthographic projection

2.1.1 Scaling

- 2D Transformation matrix: $\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$
- Scaling a vector: $\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax \\ by \end{bmatrix}$
  - Scaling factors: a and b, along x and y axes
  - If a=b → uniform (isotropic) scaling

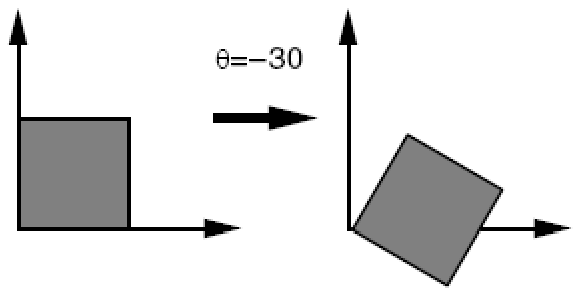
    shape preserved, only size changed
  - If a=b=1 → identity transform
- Directly extendible to 3D



2.1.2 Rotation and Shear

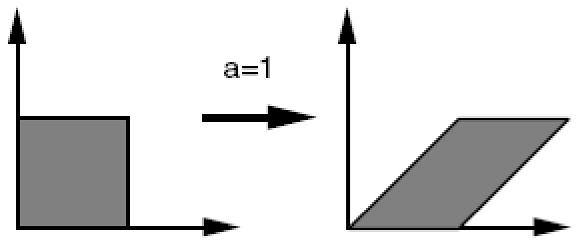
- Rotation in 2D:

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



- Shear: let one of the off-diagonal elements of the scaling transformation matrix be non-zero

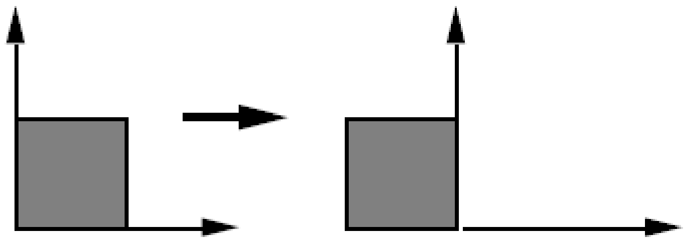
$$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + ay \\ y \end{bmatrix}$$



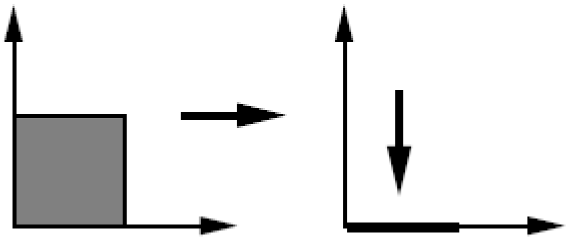
2.1.3 Reflection and Orthographic Projection

- Reflection = scaling with negative factors

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x \\ y \end{bmatrix}$$



- Orthographic Projection



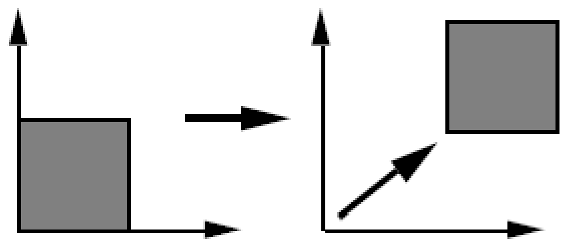
$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix}$$

  - Does not map a basis onto another basis
  - Singular transformation (can't be inverted, lost all depth information along a direction)

2.2 Translation

- simply

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} x+\delta_1 \\ y+\delta_2 \end{bmatrix}$$



- Not a linear transformation
  (can't be computed by 2*2 matrix multiplication)

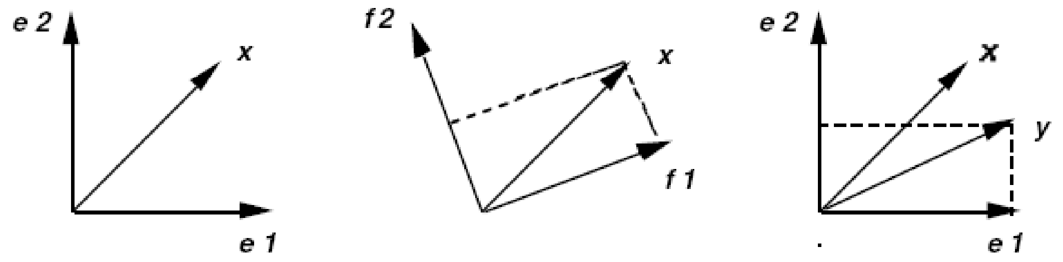
- <u>Affine Transformations</u>

  = translations + non-degenerate linear transformations

- <u>Rigid Transformations</u> = translations + rotations
  - Preserve distance, also called isometries

2.3 Change of Basis

- Let x be a vector with components X^E in basis E.
- Consider a new basis F, obtained from E by a transformation T_{ef} (corresponding matrix M^e_{ef} in basis E)
- So new basis: $f_i = EF_i^e$ or $F = \begin{bmatrix} f_1 & \cdots & f_n \end{bmatrix} = \begin{bmatrix} EF_1^e & \cdots & EF_n^e \end{bmatrix} = EM_{ef}^e$
- In the new basis, the components of x→ $x = EX^e = FX^f = EM_{ef}^e X^f$
- So $X^e = M_{ef}^e X^f$ or $X^f = {M_{ef}^e}^{-1} X^e$
- The same as that we apply a transformation T_{ef} to the vector x, and get a new vector y : $Y^e = M_{ef}^e X^e$
- First way →the same vector, the basis changes
- Second way → the basis fixed, the vector transformed
- Transform vectors = inversely transform basis



2.4 Homogeneous Coordinates

- Translations and linear transformations can be treated more uniformly if we introduce a different system of coordinates: homogeneous coordinates.
- Use the 2D example in the following, but can be generalized to n-D straightforwardly.
- Given a vector x with components X, we want to apply to it a linear transformation with matrix M, to get another vector y with components Y:

1. introduce an additional component and associate with the vector x the column matrix:

$$X^* = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ 1 \end{bmatrix}$$

X* are called homogeneous coordinates

2.4 Homogeneous Coordinates (cont.)

2. Also add a third row and column to the linear transformation matrix $M^* = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$ i.e. $M^* = \begin{bmatrix} M & 0 \\ 0 & 1 \end{bmatrix}$

3. Multiply this augmented matrices, and get:

$$M^* X^* = \begin{bmatrix} M & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = \begin{bmatrix} MX \\ 1 \end{bmatrix} = \begin{bmatrix} Y \\ 1 \end{bmatrix} = Y^*$$

Nothing changed so far.

- When elements of the third column become non-zero:

e.g. $M^* = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}$

- We get:

$$Y^* = M^* X^* = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \\ 1 \end{bmatrix}$$

2.4 Homogeneous Coordinates (cont.)

- Now we have uniform treatment of translation and rotation
  - only one procedure needed to process both
  - one matrix-multiplication hardware works for both
  - Also deal with projections (needed for displaying 3D objects, later)

2.4 Homogeneous Coordinates (cont.)

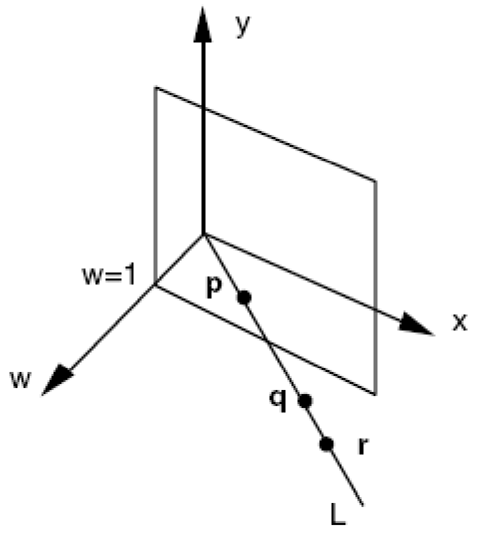
-- Geometric Interpretation

Not just a trick but with geometric intuition

(1) generalize the coordinates of an Euclidean point p:

$$P^* = \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- ❑ Increased the dimension of the original space by 1
- ❑ The original standard Euclidean plane is at w=1

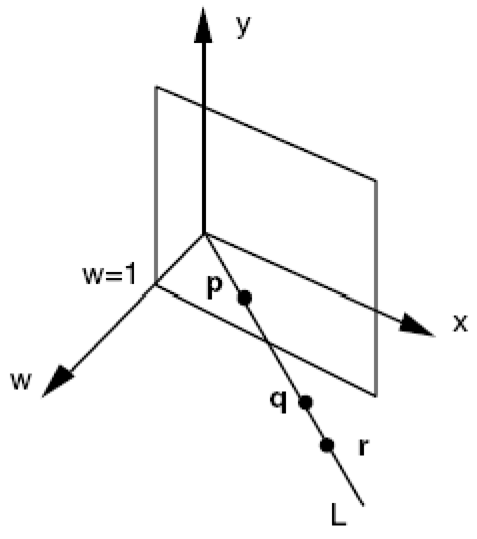


2.4 Homogeneous Coordinates (cont.)

-- Geometric Interpretation

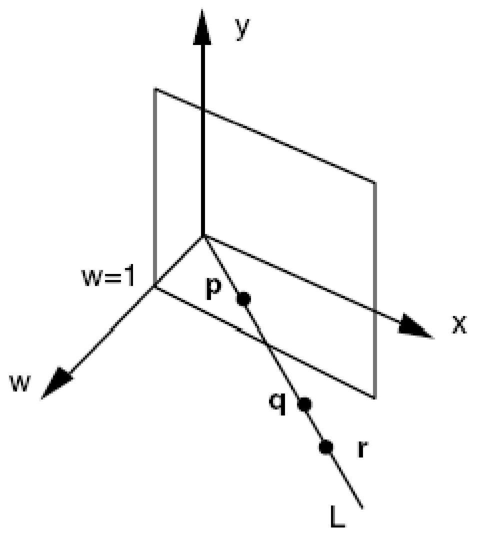
(2) Connect p with the origin (get line L)

- Each p corresponds to one line L, any point on L differ with p by a scaling
- We can always normalize the coordinate to $\begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$



2.4 Homogeneous Coordinates (cont.)

-- Geometric Interpretation



- The set of all lines through the origin of our auxiliary 3D space is called the <u>projective plane</u>.
- The elements of the projective plane are called <u>projective points</u>. (note they are actually lines)
- Each Euclidean point p has a corresponding line L and projective point p*
- Therefore, we can manipulate Euclidean points through operations on their projective counterparts.

  (see the diagram on the right:

an affine transform T

= Imbed + projective transform + normalize)

$$\begin{tikzcd}
p \arrow[r, "T"] \arrow[d, "\text{Imbed}"'] & q \\
p^{*} \arrow[r, "T^{*}"'] & q^{*} \arrow[u, "\text{Normalize}"']
\end{tikzcd}$$

2.5 Change of Frames

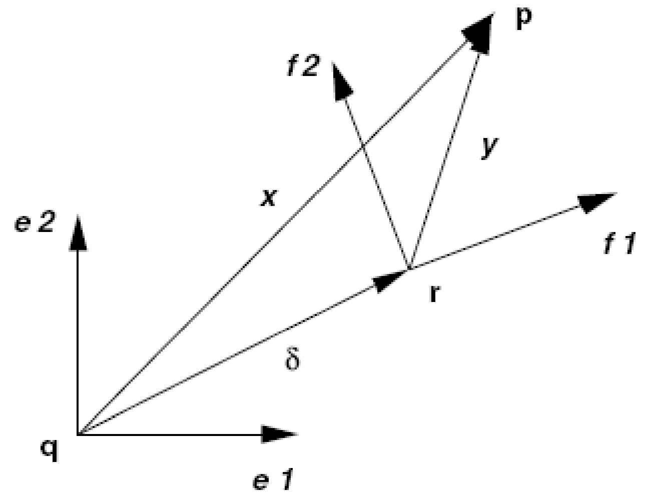
- Consider two frames E=(E, q) and F=(F, r)
- A point p corresponds two different vectors in E and F:

$$\mathrm{p} \overset{\mathbf{E}}{\longleftrightarrow} x$$

$$\mathrm{p} \overset{\mathbf{F}}{\longleftrightarrow} y$$

$$x = \delta + y$$

$$\delta = \mathrm{r} - \mathrm{q}$$

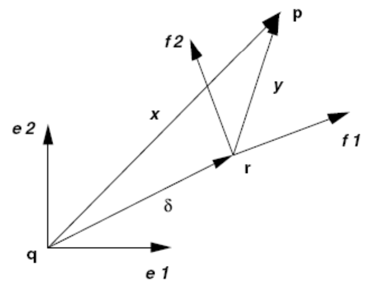


- The coordinates matrices of p in E and F are:

$$P^f = Y^f$$

$$P^e = X^e = Y^e + \underbrace{D^e}_{\text{Components of } \delta \text{ in E}} = M^e_{ef} Y^f + D^e = \underbrace{M^e_{ef}}_{\text{Transformations map basis E to F}} P^f + D^e$$

2.5 Change of Frames (cont.)



$$P^f = Y^f$$

$$P^e = X^e = Y^e + D^e = M^e_{ef} Y^f + D^e = M^e_{ef} P^f + D^e$$

- Now it can be written using homogeneous coordinates:

$$\begin{bmatrix} P^e \\ 1 \end{bmatrix} = \begin{bmatrix} M^e_{ef} & D^e \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P^f \\ 1 \end{bmatrix}$$

i.e. $\left(P^*\right)^e = \left(M^*_{ef}\right)^e \left(P^*\right)^f$ where $\left(M^*_{ef}\right)^e = \begin{bmatrix} M^e_{ef} & D^e \\ 0 & 1 \end{bmatrix}$.

Therefore, the effect of a change of frame on the homogeneous coordinates of a point ←→ the effect of a change of basis on the components of a vector

e.g. here frame F comes from an affine transformation on (q, E), specifically, a rotation followed by a translation

$$\begin{bmatrix} I & D^e \\ 0 & 1 \end{bmatrix} \begin{bmatrix} M^e_{ef} & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} M^e_{ef} & D^e \\ 0 & 1 \end{bmatrix} = \left(M^*_{ef}\right)^e$$