

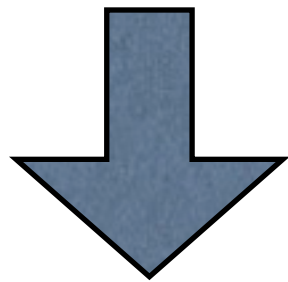


Introduction to Transformations & Matrices

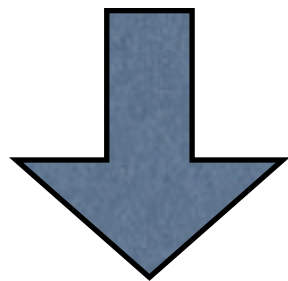
CS 355: Interactive Graphics and Image Processing

Drawing

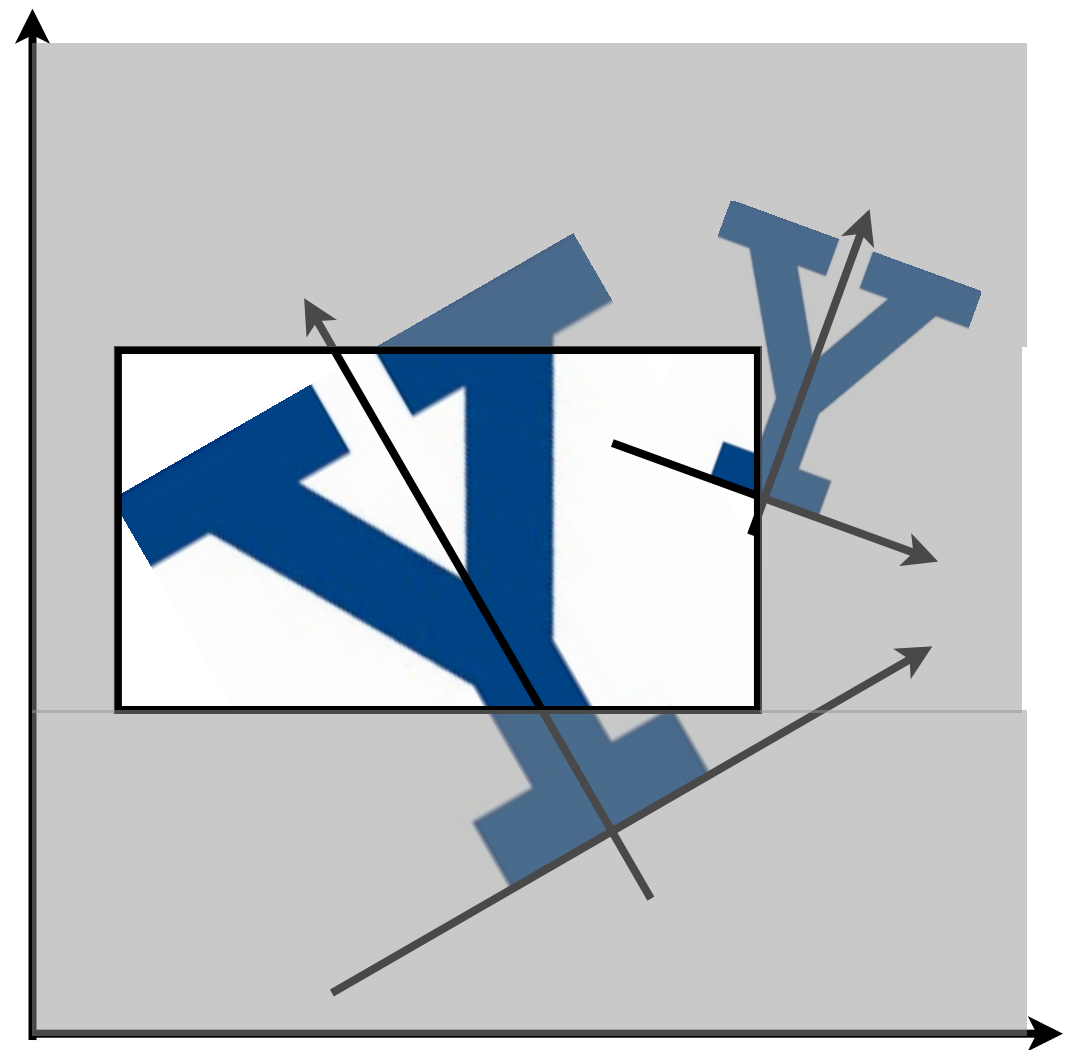
- Object Coordinates



- World Coordinates

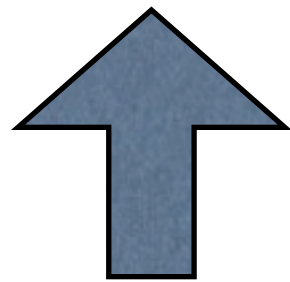


- Viewing Coordinates

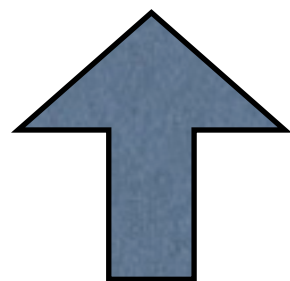


Selecting

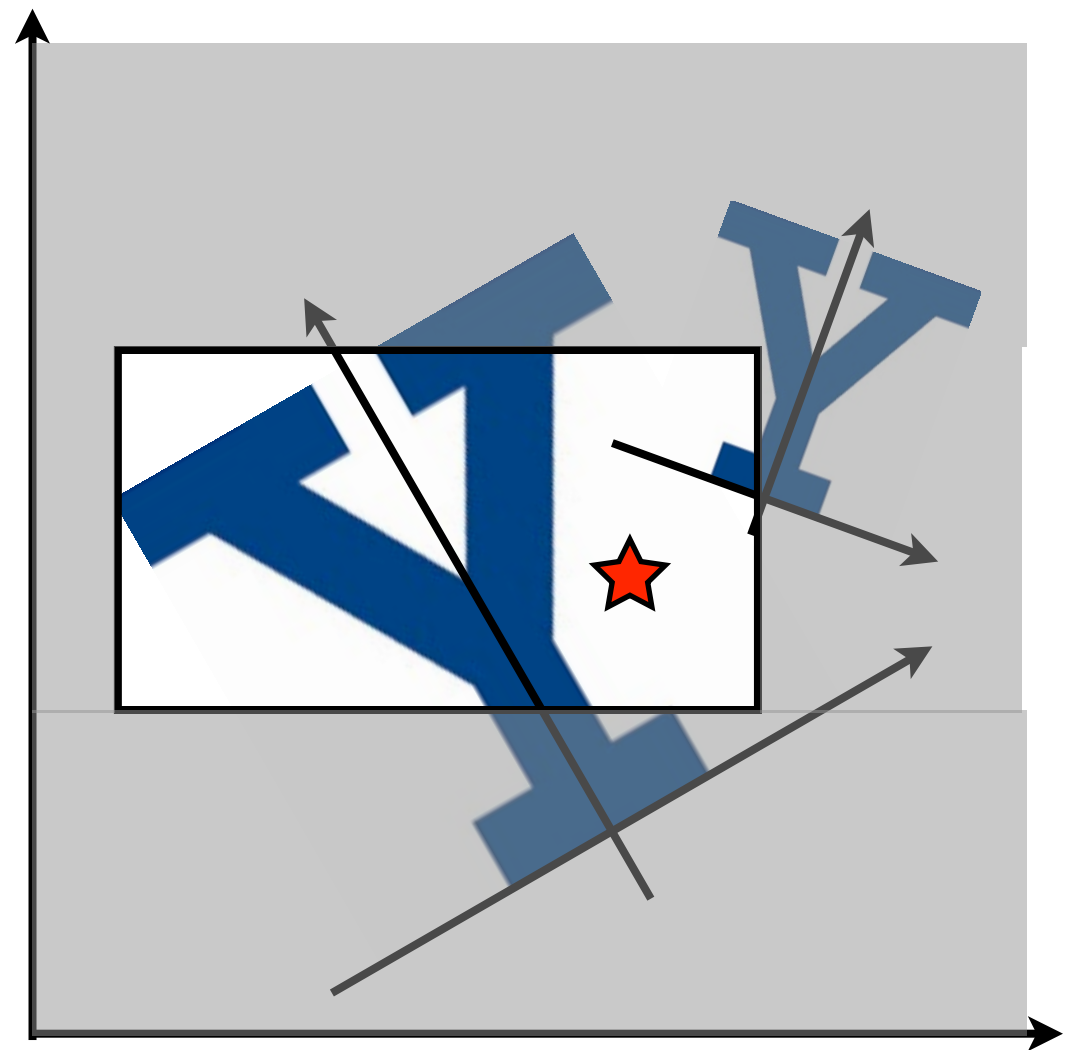
- Object Coordinates



- World Coordinates



- Viewing Coordinates

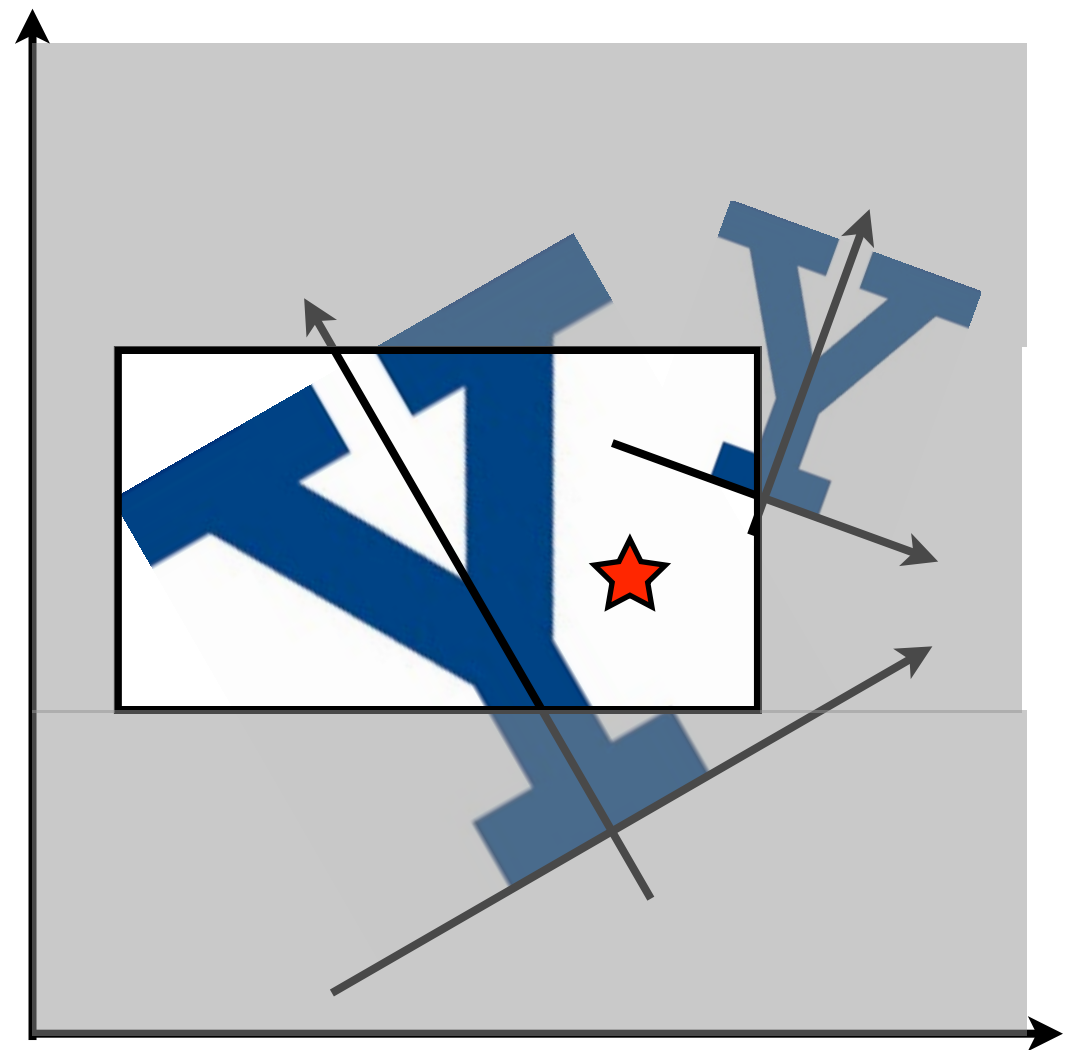


Coordinate Transformations

- To move between coordinate systems we *transform* the points
- A *transformation* is simply a mapping between points:

$$\mathbf{p}' = f(\mathbf{p})$$

- Usually 1 : 1
(but not always)



Translation

- Translating a coordinate system moves the origin
- Keeps the x and y directions the same

$$(x', y') = (x + t_x, y + t_y)$$

OR

$$\mathbf{p}' = \mathbf{p} + \mathbf{t}$$

$$\mathbf{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: Translation

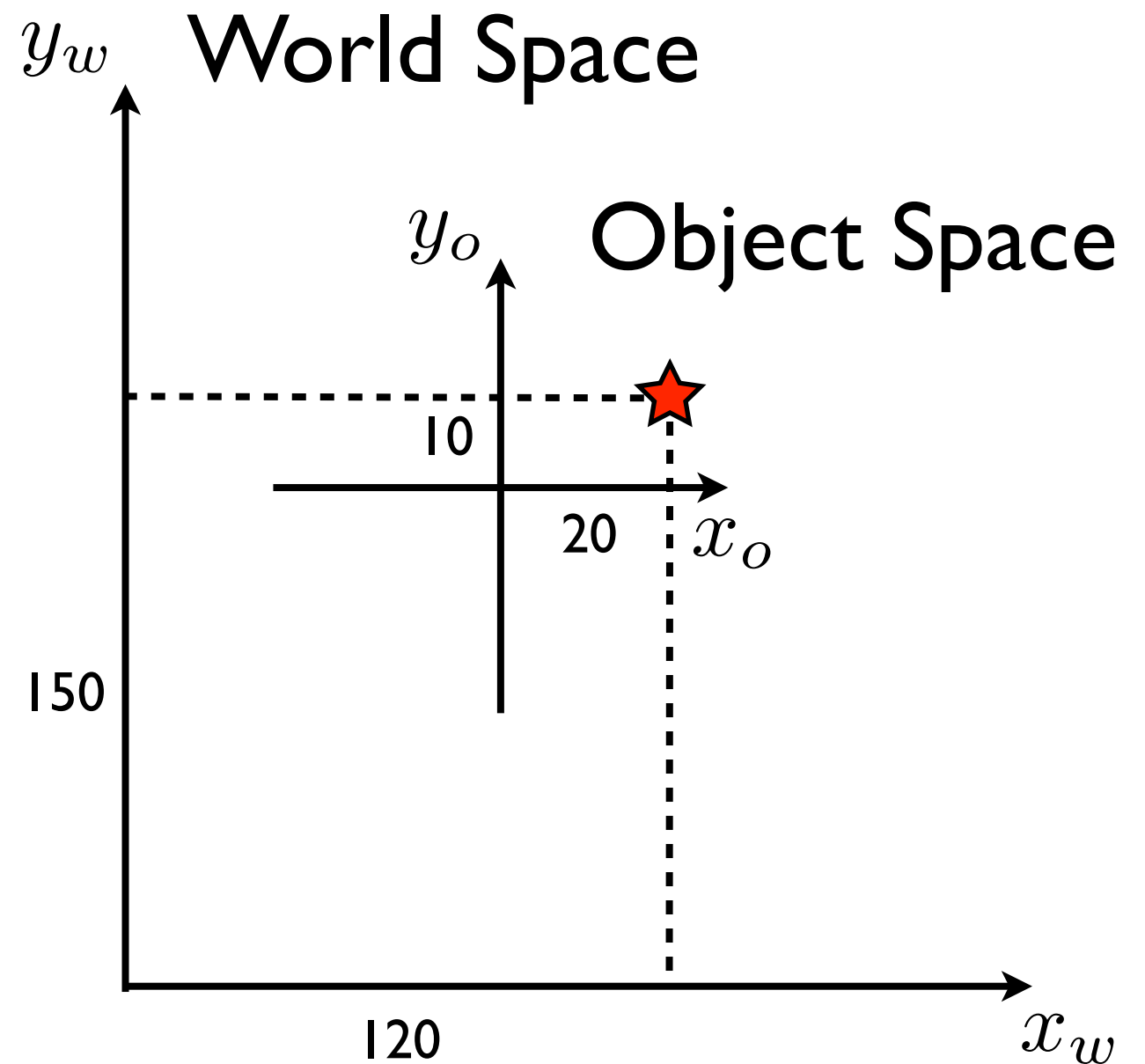
- Suppose that an object is centered at (100, 140)
- Where is the point (20, 10) in object space at in the world?

- Object to world space:

$$\mathbf{p}_w = \mathbf{p}_o + \begin{bmatrix} 100 \\ 140 \end{bmatrix}$$

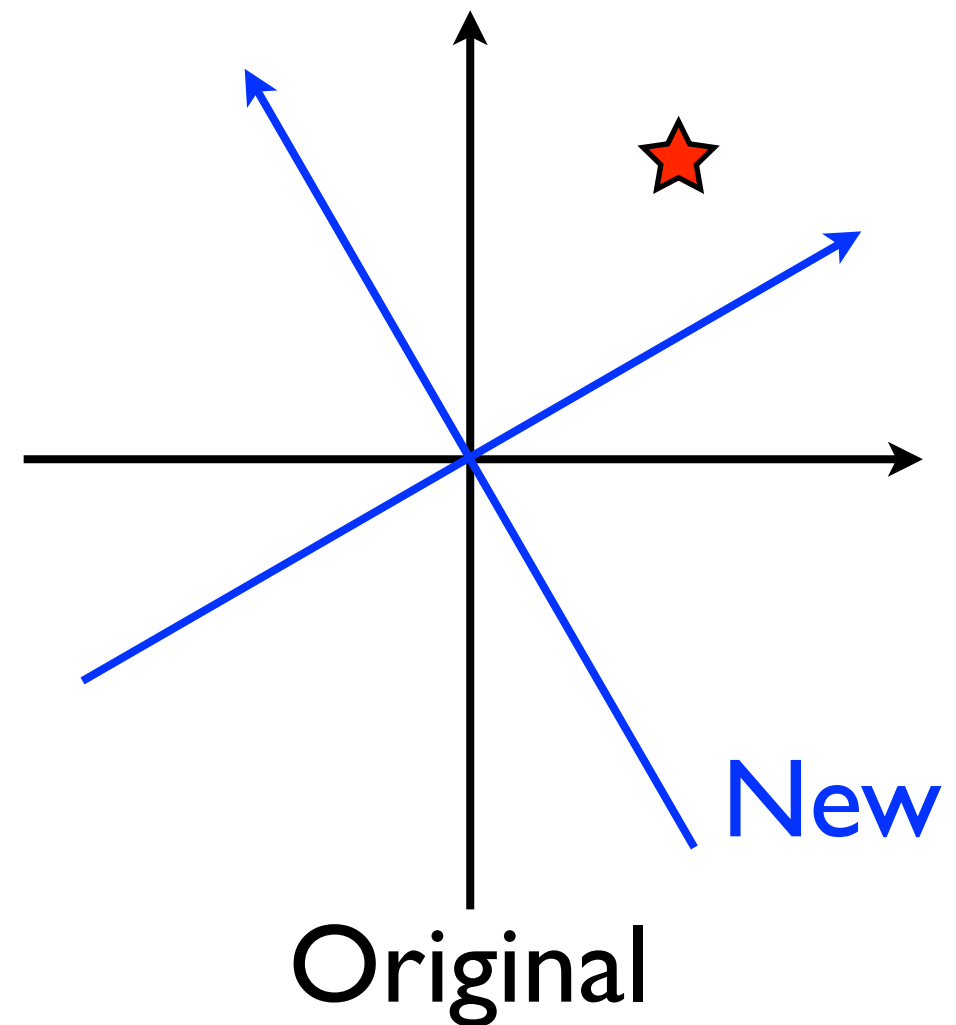
- World to object space:

$$\mathbf{p}_o = \mathbf{p}_w - \begin{bmatrix} 100 \\ 140 \end{bmatrix}$$



Rotation

- Rotating a coordinate system keeps the origin, turns the axis directions
- Conceptually,
 - The point stays the same and the axes change
 - The axes stay the same and the point rotates around the origin

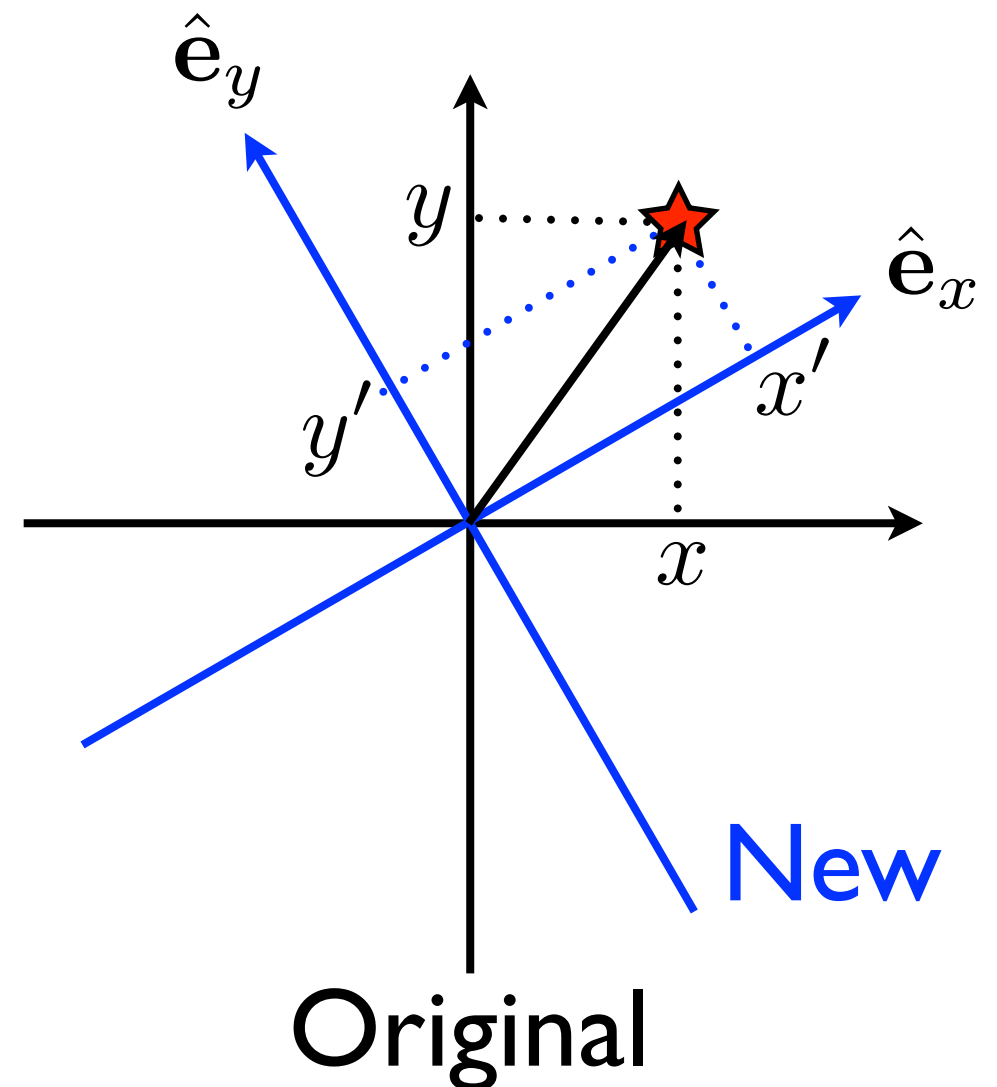


Computing Rotation

- To compute coordinates in the rotated system, just *project to each of the new axis directions*
- Use dot products!

$$p'_x = \mathbf{p} \cdot \hat{\mathbf{e}}_x$$

$$p'_y = \mathbf{p} \cdot \hat{\mathbf{e}}_y$$



**More on transformations later,
but first...**

Matrices

$$\mathbf{M} = \begin{bmatrix} 3 & 1 & 8 & 5 \\ -1 & 4 & -3 & 3 \\ 2 & 0 & -1 & 4 \end{bmatrix}$$

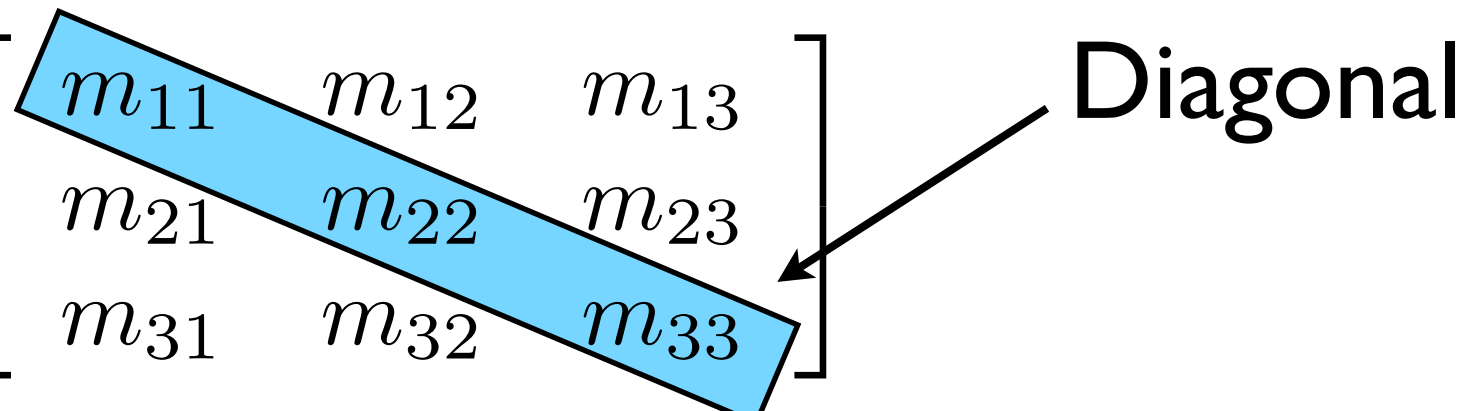
A matrix is an n by m array of numbers

Notation

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

Index an array by row, then column

Square Matrices

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$


The diagram shows a 3x3 matrix with elements m_{ij} . A light blue shaded parallelogram highlights the diagonal band, containing the elements m_{11} , m_{22} , and m_{33} . An arrow points from the word "Diagonal" to the m_{22} element.

Square matrices have the same number of rows as columns ($n = m$)

If everything off the diagonal is 0,
it is a *diagonal matrix*

Vectors as Matrices

$$\mathbf{v} = \begin{bmatrix} 2 \\ 3 \\ -1 \end{bmatrix}$$

A vector is simply an $n \times 1$ matrix

(technically, this is a *column vector*—some use *row vectors*)

Transposing

$$\mathbf{M} = \begin{bmatrix} 3 & 1 & 8 & 5 \\ -1 & 4 & -3 & 3 \\ 2 & 0 & -1 & 4 \end{bmatrix} \quad \mathbf{M}^T = \begin{bmatrix} 3 & -1 & 2 \\ 1 & 4 & -3 \\ 8 & -3 & -1 \\ 5 & 3 & 4 \end{bmatrix}$$

“M transpose”

The transposition of a matrix simply swaps
the rows for the columns

$$\mathbf{M}_{ij}^T = \mathbf{M}_{ji}$$

Stacks of Transposed Vectors

$$\mathbf{M} = \begin{bmatrix} \boxed{3 \quad 1 \quad 8 \quad 5} \\ \boxed{-1 \quad 4 \quad -3 \quad 3} \\ \boxed{2 \quad 0 \quad -1 \quad 4} \end{bmatrix}$$

A matrix is an n by m array of numbers

OR a matrix is a stack of n transposed vectors,
each with m elements

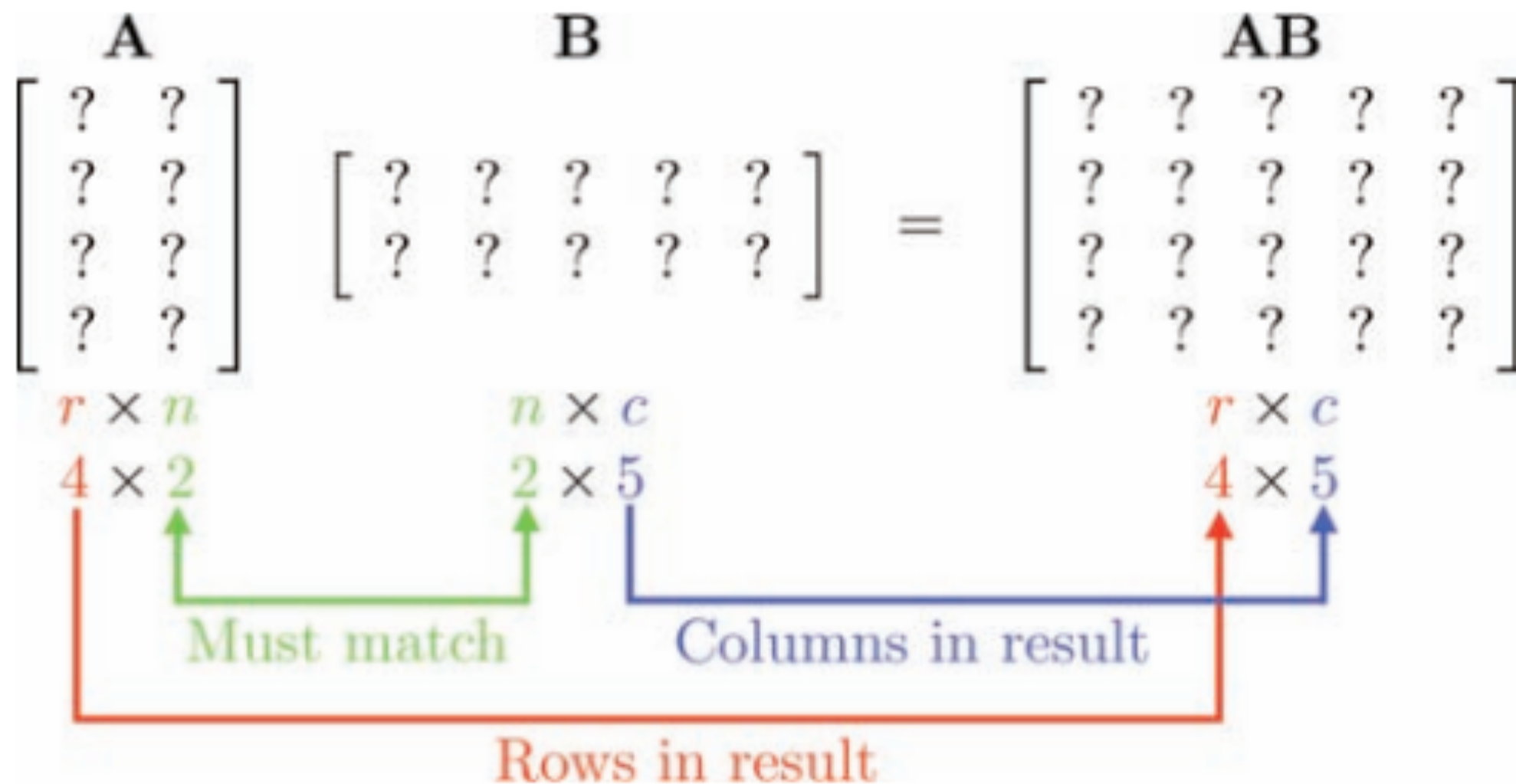
Multiplying by Scalar

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

$$k\mathbf{M} = k \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} k m_{11} & k m_{12} & k m_{13} \\ k m_{21} & k m_{22} & k m_{23} \\ k m_{31} & k m_{32} & k m_{33} \end{bmatrix}$$

Multiply a matrix by a scalar
multiplies each element accordingly

Matrix Multiplication



Width of first must match height of second

Matrix Multiplication

$$\mathbf{C} = \mathbf{AB}$$

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \end{bmatrix}$$

$c_{24} = a_{21}b_{14} + a_{22}b_{24}$

↑
Look, a dot product!

Alternate View

$$\mathbf{C} = \mathbf{A}\mathbf{B}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \end{bmatrix}$$
$$c_{43} = a_{41}b_{13} + a_{42}b_{23}$$

$$c_{ij} = \mathbf{A}.\text{row}[i] \cdot \mathbf{B}.\text{col}[j]$$


Identity Matrix

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{MI} = \mathbf{IM} = \mathbf{M}$$

Matrix Inversion

The inverse of a matrix is the matrix such that

$$\mathbf{M}\mathbf{M}^{-1} = \mathbf{M}^{-1}\mathbf{M} = \mathbf{I}$$


inverse

There are multiple ways to compute the inverse of a matrix, but we won't cover that here

Matrix Multiplication

- Matrix multiplication is associative
- And distributes over addition
- Matrix multiplication is NOT commutative, but...
- And...

$$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$$

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$$

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$$

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}$$

Multiplying by Vector



$$\mathbf{b} = \mathbf{M} \mathbf{a}$$

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Multiplying a vector by a matrix is just a compact way of writing a bunch of dot products

Row vs. Column



“row vectors” “column vector”

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

$$\begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

“row vector” “column vectors”

Row vs. Column

- Column vectors:
 - Most often used in math and science
 - Used in most scientific computing code
 - Used in many graphics libraries (e.g., DirectX)
 - Writes a little cleaner
 - Read backwards:
- Row vectors:
 - Used by your book and many programmers
 - Used in many graphics libraries (OpenGL)
 - Read forwards:

$$\mathbf{CBA}\mathbf{v} = \mathbf{C}(\mathbf{B}(\mathbf{A}\mathbf{v}))$$

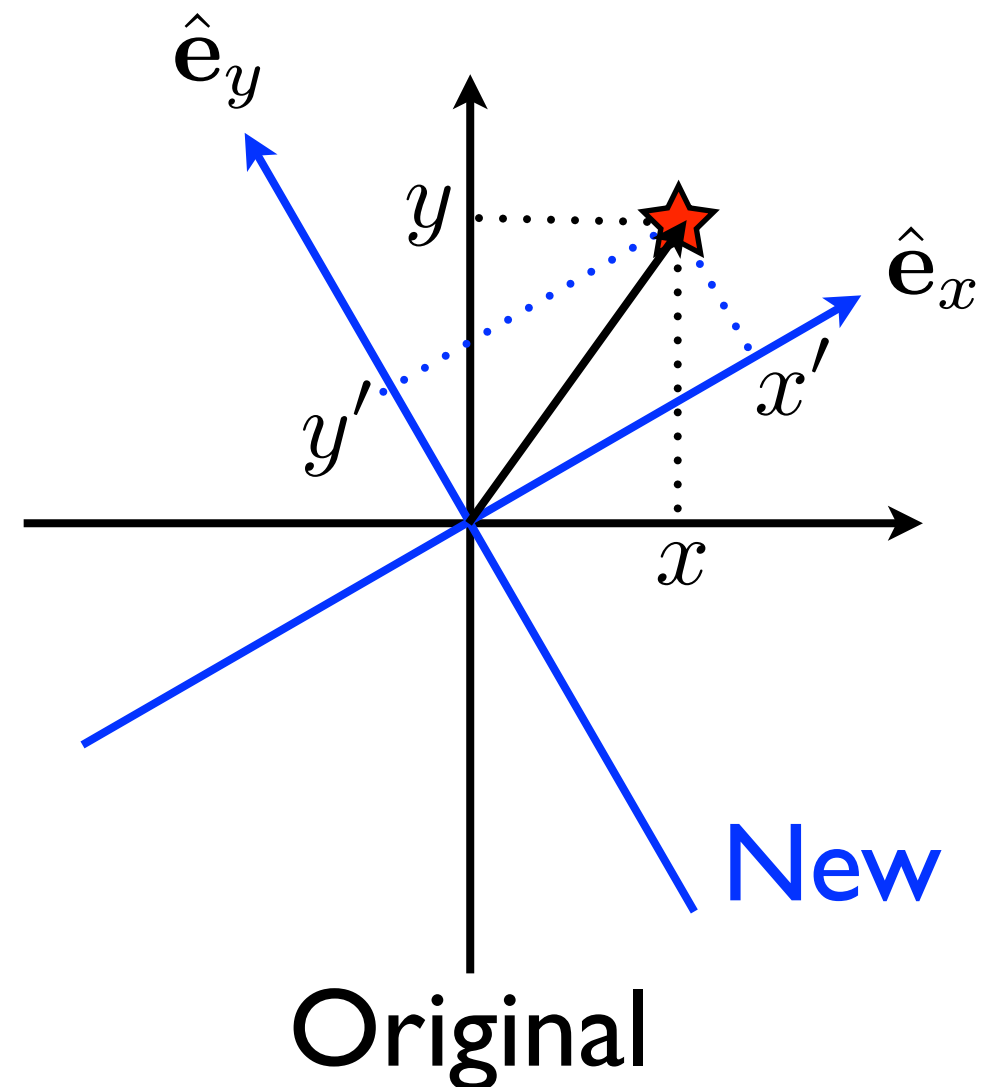
$$\mathbf{vABC} = (((\mathbf{vA})\mathbf{B})\mathbf{C})$$

Computing Rotation

- To compute coordinates in the rotated system, just *project to each of the new axis directions*
- Use dot products!

$$p'_x = \mathbf{p} \cdot \hat{\mathbf{e}}_x$$

$$p'_y = \mathbf{p} \cdot \hat{\mathbf{e}}_y$$

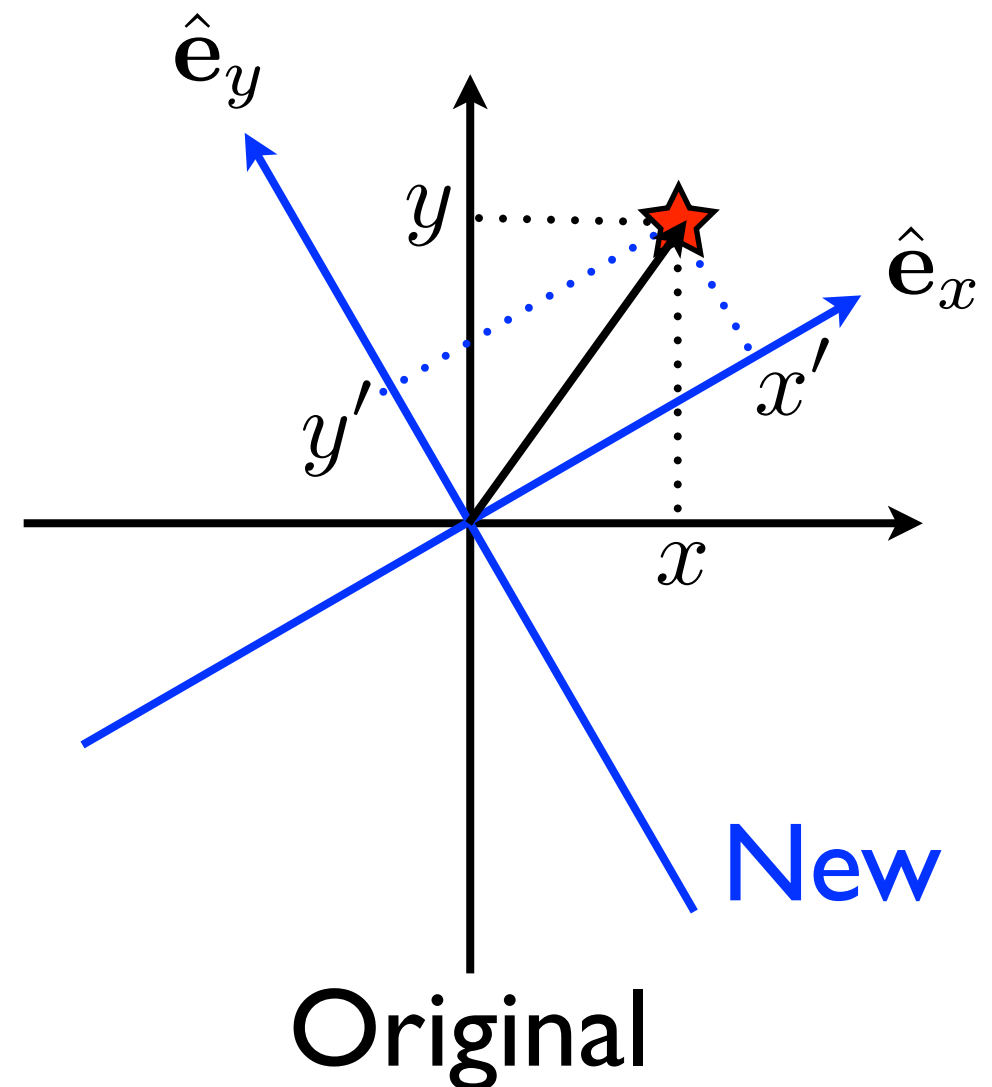


Computing Rotation

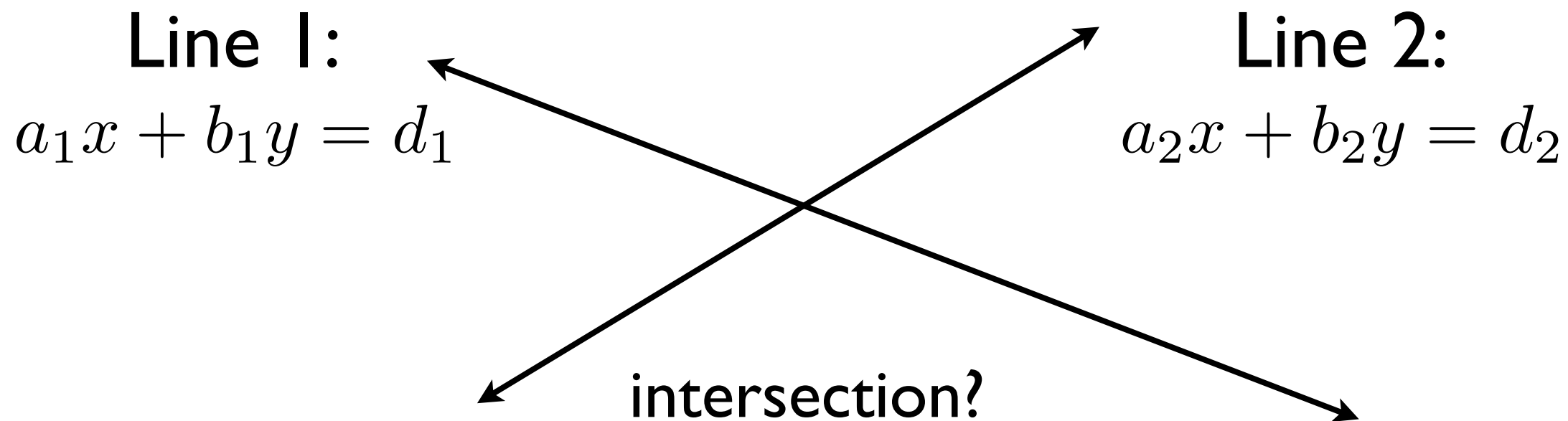
- To compute coordinates in the rotated system, just *project to each of the new axis directions*
- Use dot products!

$$\mathbf{p}' = \begin{bmatrix} e_{x1} & e_{x2} \\ e_{y1} & e_{y2} \end{bmatrix} \mathbf{p}$$

But do it with a matrix!!



More Applications



$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

This is a *system of linear equations*

Ways to solve these are covered in Math 313

Coming up...

- Linear (matrix) transformations
 - Forward
 - Inverse