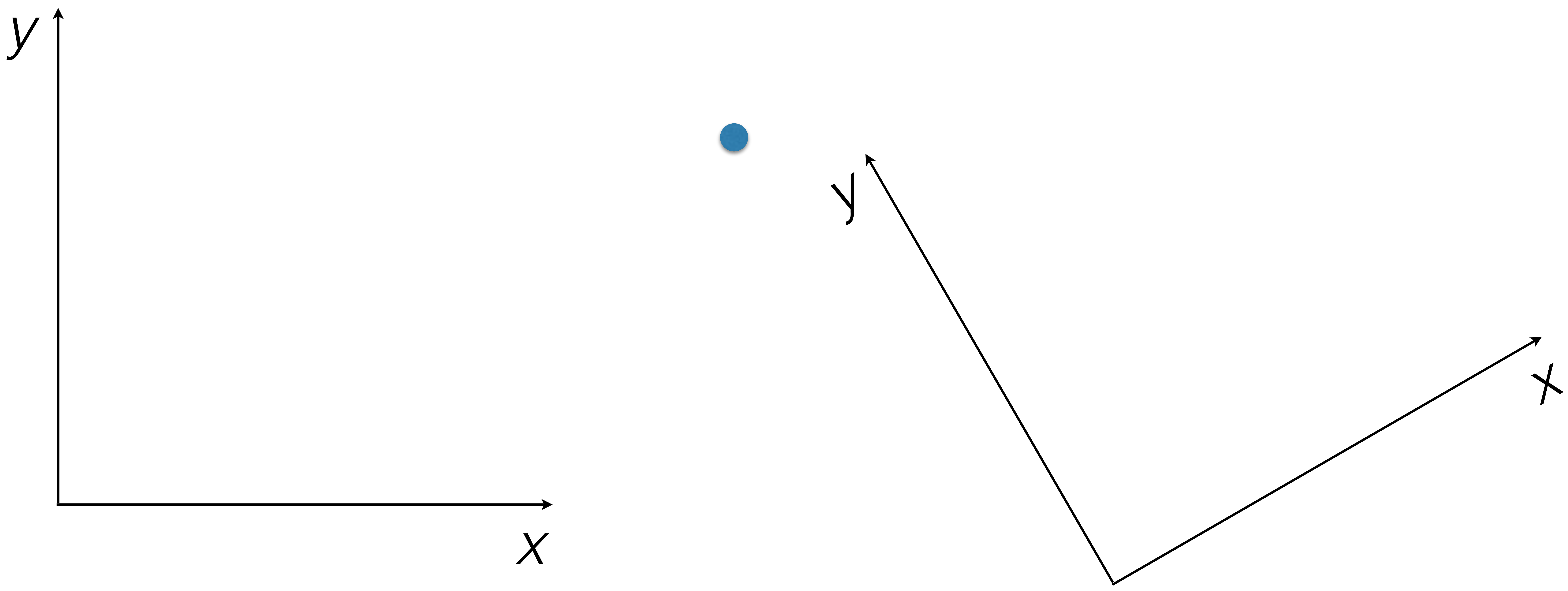
Introduction to Transformations & Review of Matrices

CS 355: Introduction to Graphics and Image Processing

Changes of Coordinates

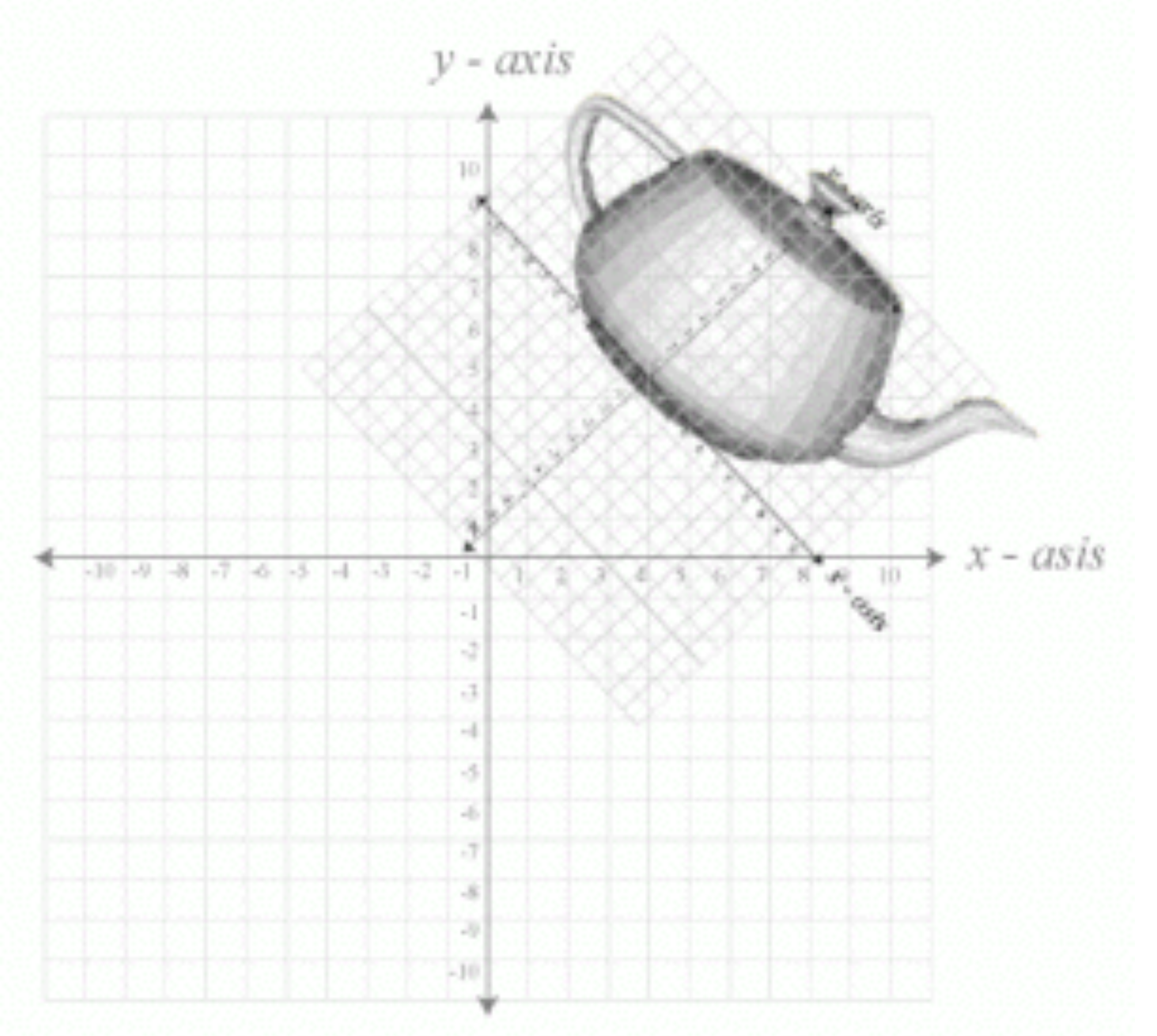


What if we have a point described on one coordinate system…

But we want to describe it in another?

Why Change Coordinates?

- Moving stuff around (Lab 4)
- Model in one space, place in another (Lab 5)
- Modeling hierarchically (Lab 6)
- Where are 3D points *relative to the camera*? (Lab 7, Lab 9)
- Geometric tests (Lab 9)
- Data transformations (Lab 10)



Translation

- *Translating* a coordinate system simply moves the origin
- Or can be thought of as moving the point
- Keeps the x and y directions the same
- Just add desired x, y offsets

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

OR

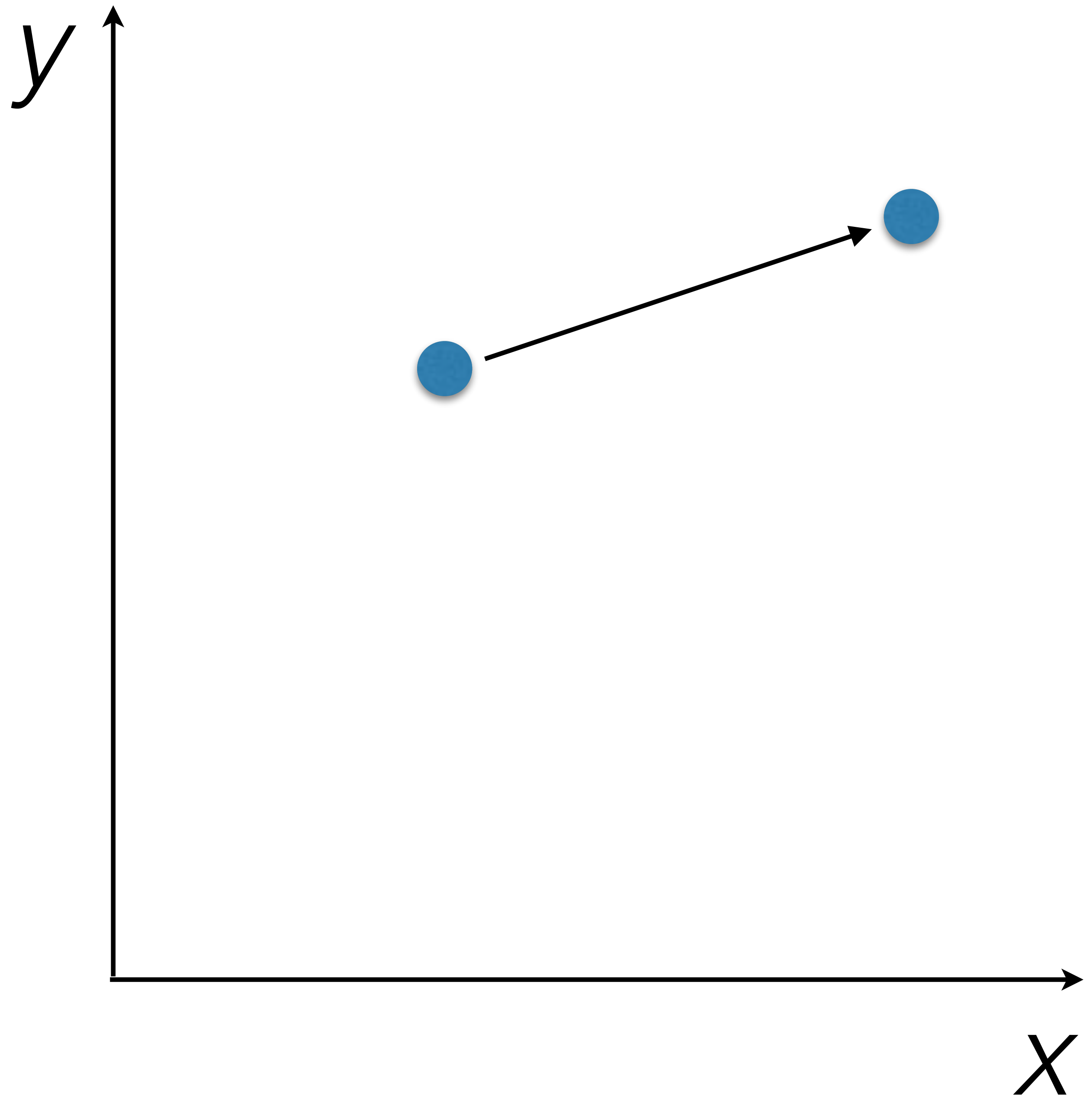
$$\mathbf{p}' = \mathbf{p} + \mathbf{t}$$

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

Example: Translation

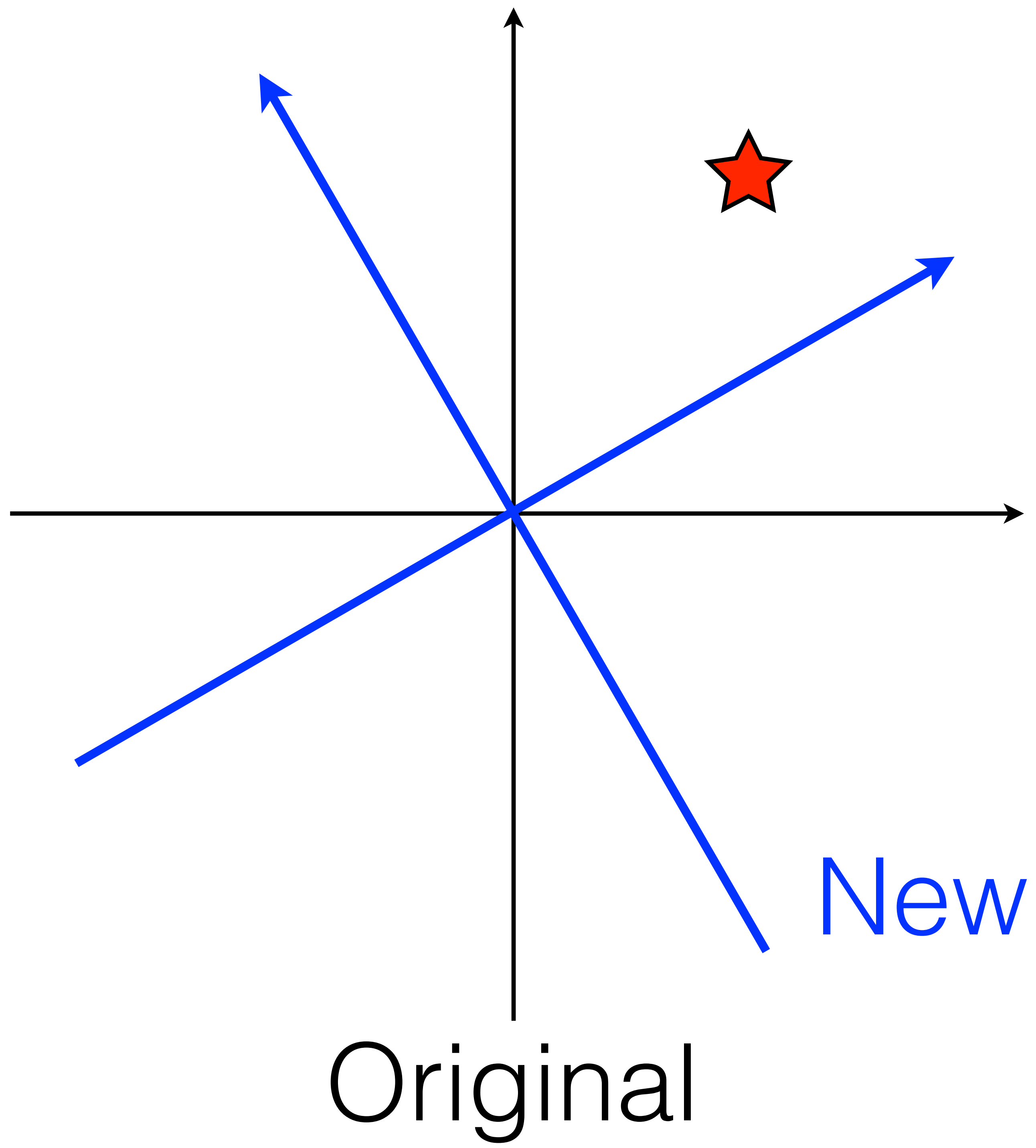
- Suppose that you have a point ***p*** at (100,200) and you want to translate it by [160,50]?

$$
\begin{aligned}
\mathbf{p}' &= \mathbf{p} + \mathbf{t} \\
&= (100, 200) + [160, 50] \\
&= (260, 250)
\end{aligned}
$$



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

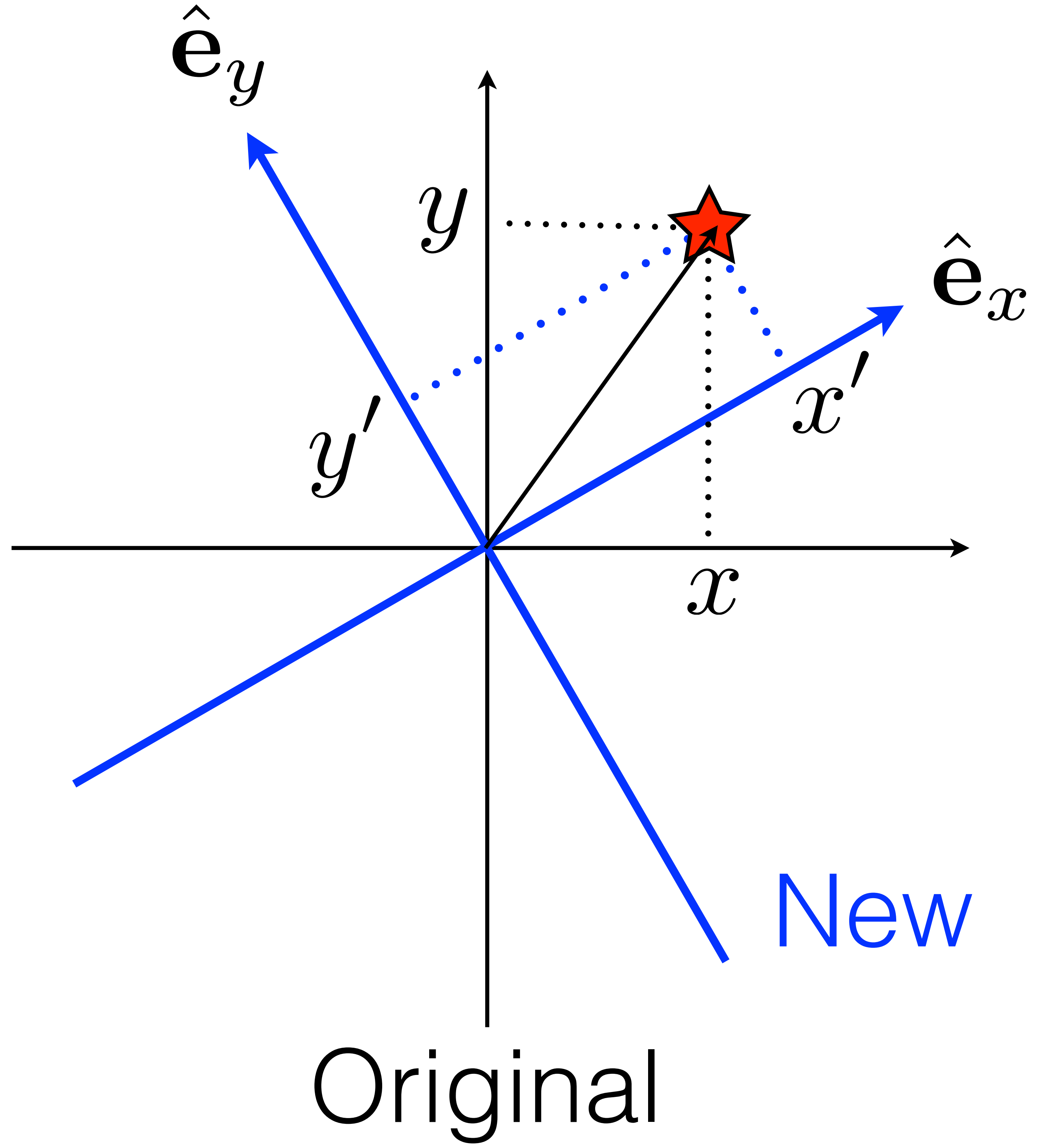


Computing Rotation

- 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 let's review…

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

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

Diagonal

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 x 1 matrix

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

Transposing

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

"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

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

A matrix is an n by m array of numbers

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

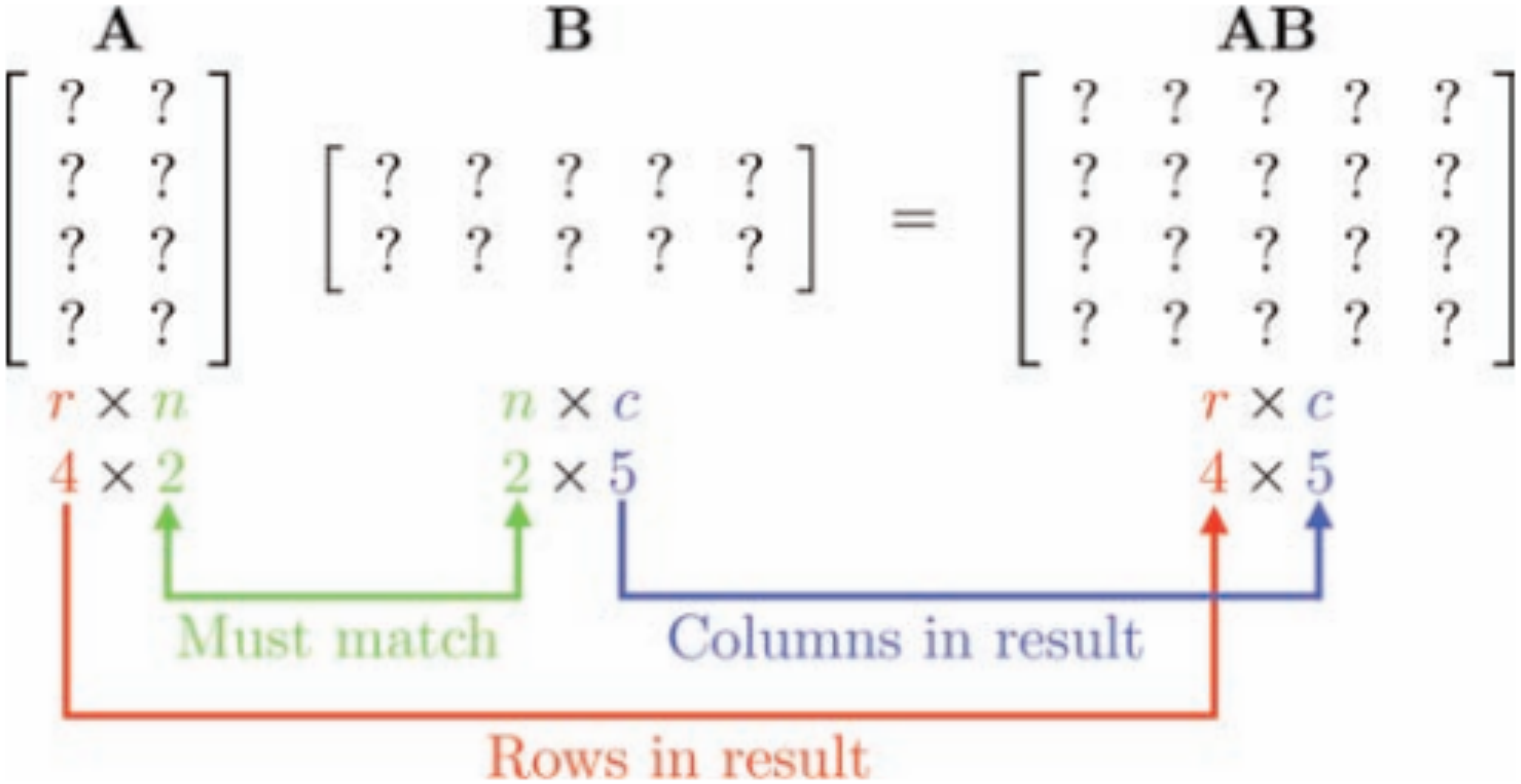
Multiplying by Scalar

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

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

Multiply a matrix by a scalar
multiplies each element accordingly

Matrix Multiplication



Width of first must match height of second

Matrix Multiplication

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

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

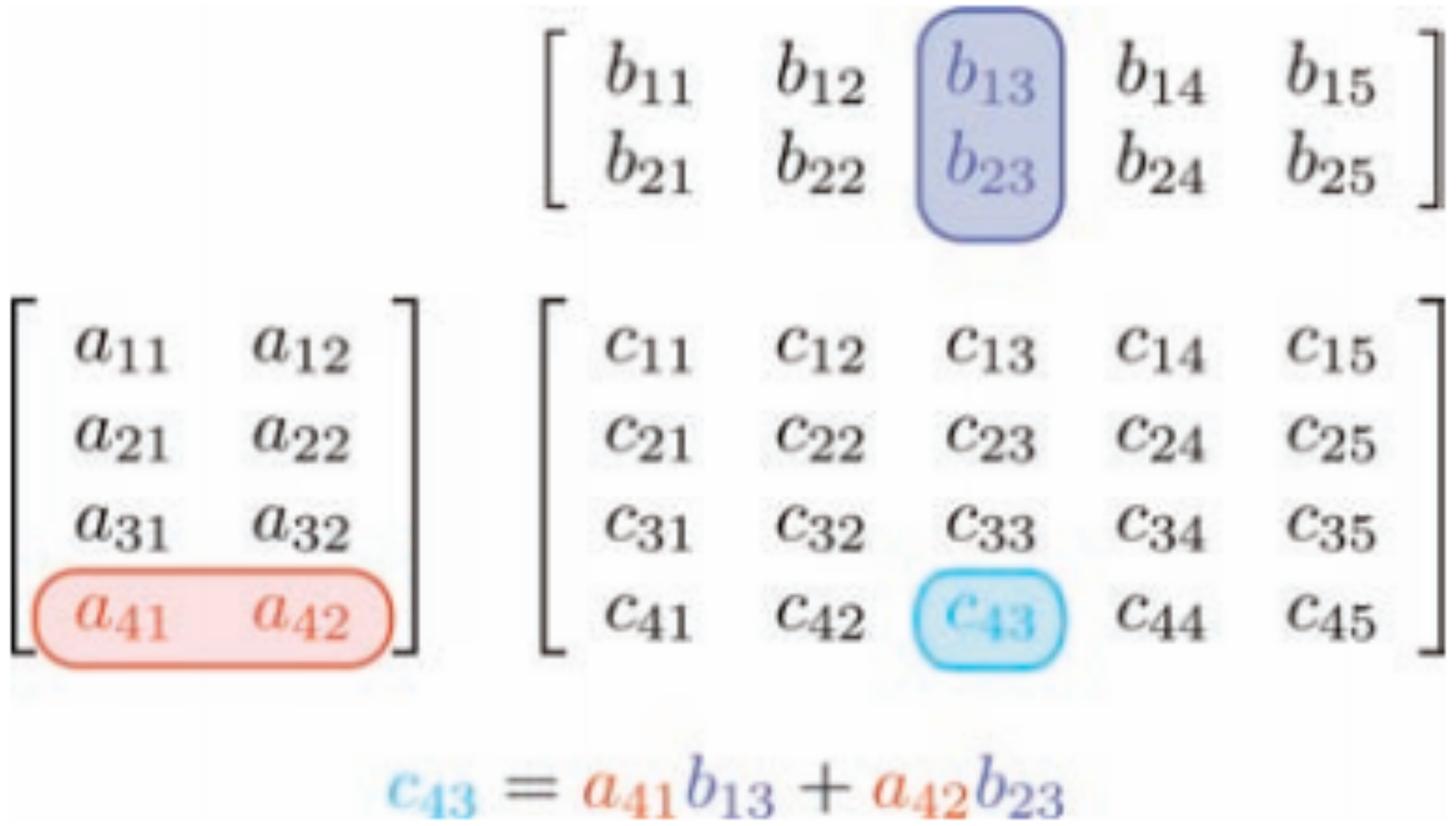
$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & \boxed{c_{24}} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41}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

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

Look, a dot product!

Alternate View

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



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

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

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

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 $$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$$
- And distributes over addition $$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$$
- Is NOT commutative, but... $$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$$
- And... $$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$$

Multiplying by Vector

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

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

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 common in math and science
  - Used in most scientific computing code
  - Used in many graphics libraries
  - Writes a little more compactly
  - Read right-to-left:

$$\mathbf{CBAv} = \mathbf{C}(\mathbf{B}(\mathbf{Av}))$$

- Row vectors:
  - Used many programmers and graphics books
  - Used in many graphics libraries
  - Much less compact to write
  - Read left-to-right:

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

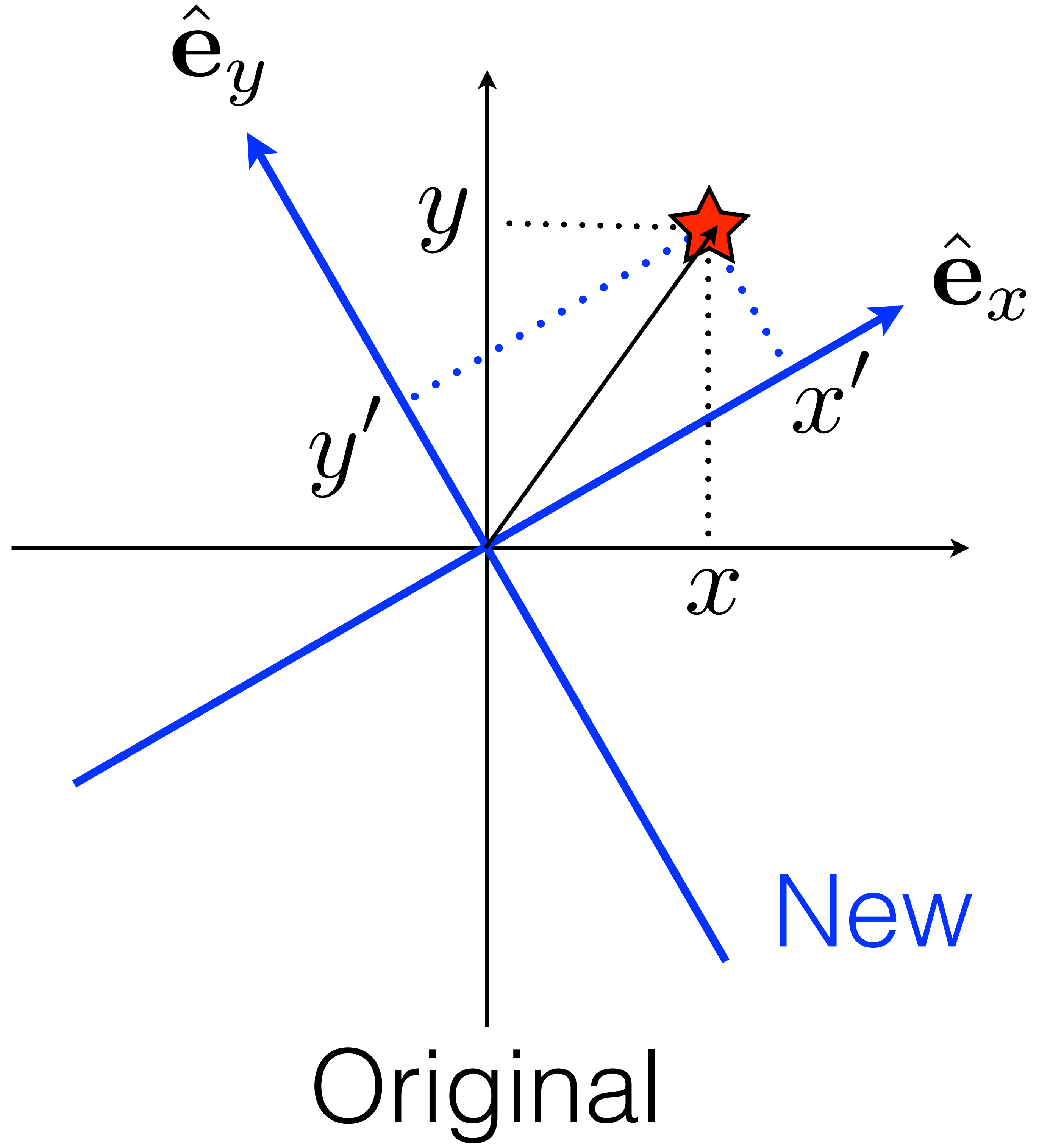
Computing Rotation

- To compute coordinates in the rotated system, just project to each of the new axis directions

- Use dot products!

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

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



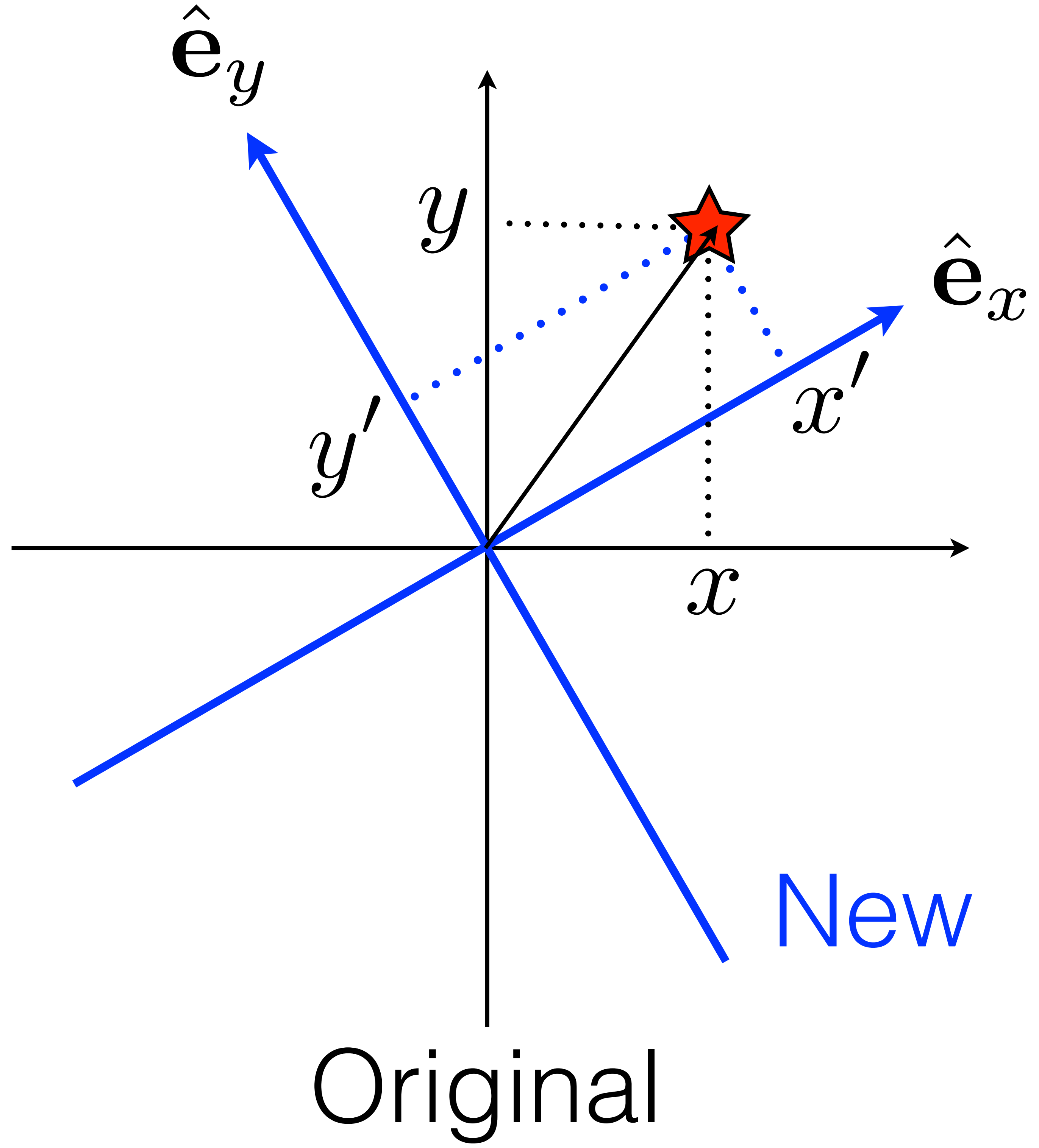
Computing Rotation

- 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!!



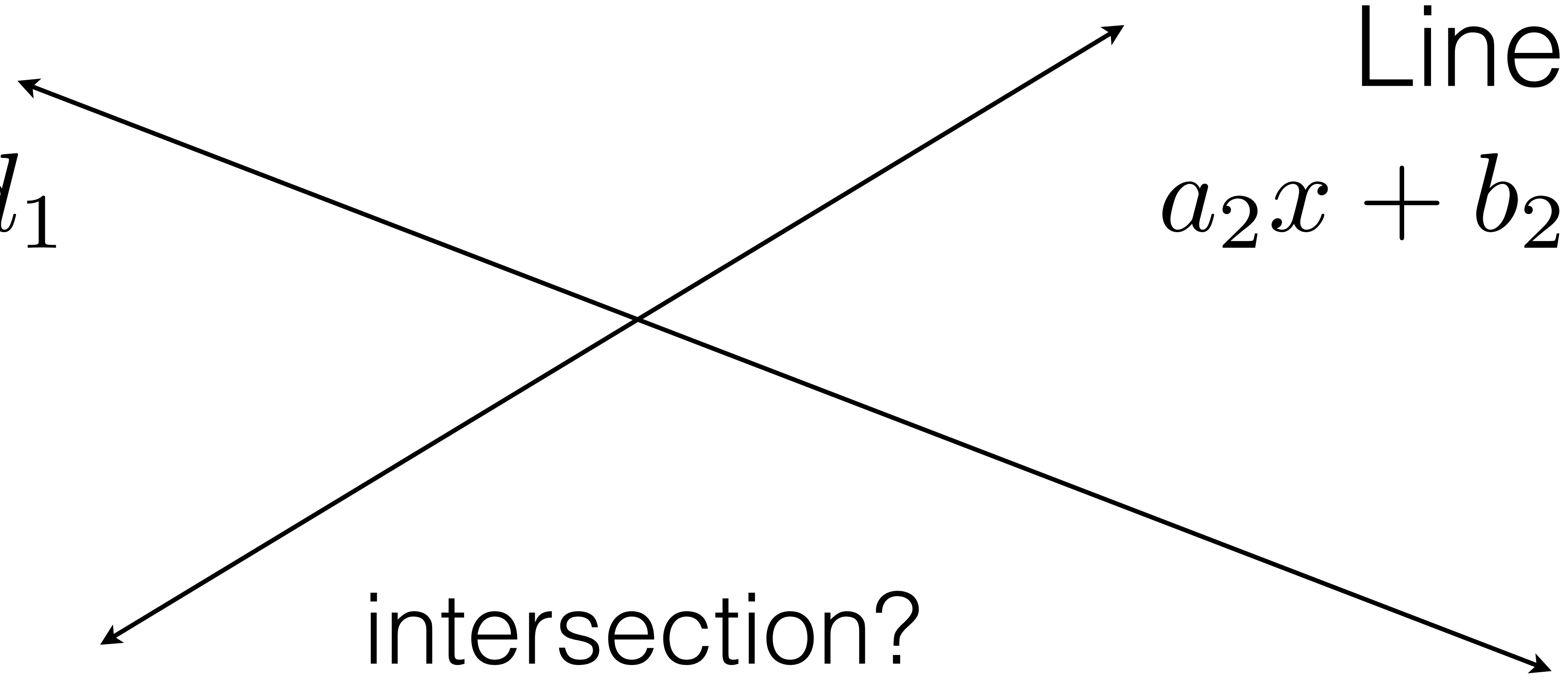
More Applications

Line 1:

$$a_1 x + b_1 y = d_1$$

Line 2:

$$a_2 x + b_2 y = d_2$$



intersection?

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

This is a *system of linear equations*

Ways to solve these are covered in Math 313, but we'll use code in Python when we have to

Coming up...

- Linear (matrix) transformations
- Homogeneous coordinates