UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

COSC 4370 Fall 2023

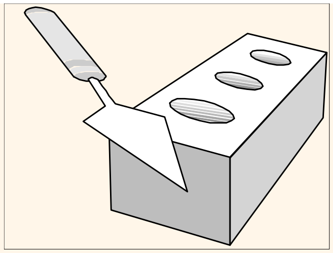
Interactive Computer Graphics

M & W 5:30 to 7:00 PM

Prof. Victoria Hilford

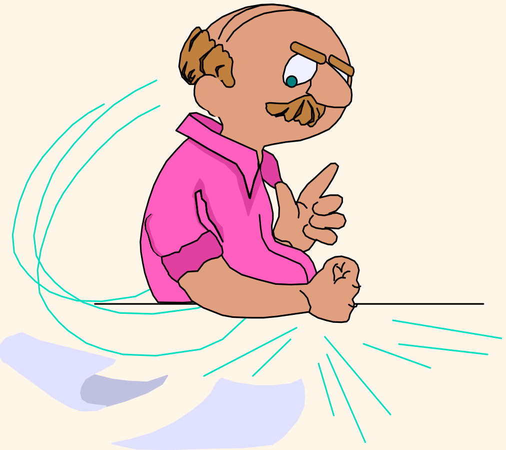
PLEASE TURN your webcam ON

NO CHATTING during LECTURE



COSC 4370

5:30 to 7



PLEASE

LOG IN

CANVAS

Please close all other windows.

From 5:30 to 6:25 PM – 55 minutes.

08.30.2023 (W 5:30 to 7) (4)		Math Review 1
09.06.2023 (W 5:30 to 7) (5)	Homework 2	Lecture 3
09.11.2023 (M 5:30 to 7) (6)		Math Review 2
09.13.2023 (W 5:30 to 7) (7)	Homework 3	Lecture 4
09.18.2023 (M 5:30 to 7) (8)		PROJECT 1
09.20.2023 (W 5:30 to 7) (9)		EXAM 1 REVIEW
09.25.2023 (M 5:30 to 7) (10)		EXAM 1

COSC 4370 – Computer Graphics

Math Review 1

4

Geometric objects and Transformations

Chapter 4

Notation: Scalars, Vectors, Matrices

- **scalar**
  - (lower case, italic)
- **vector**
  - (lower case, bold)
- **matrix**
  - (upper case, bold)

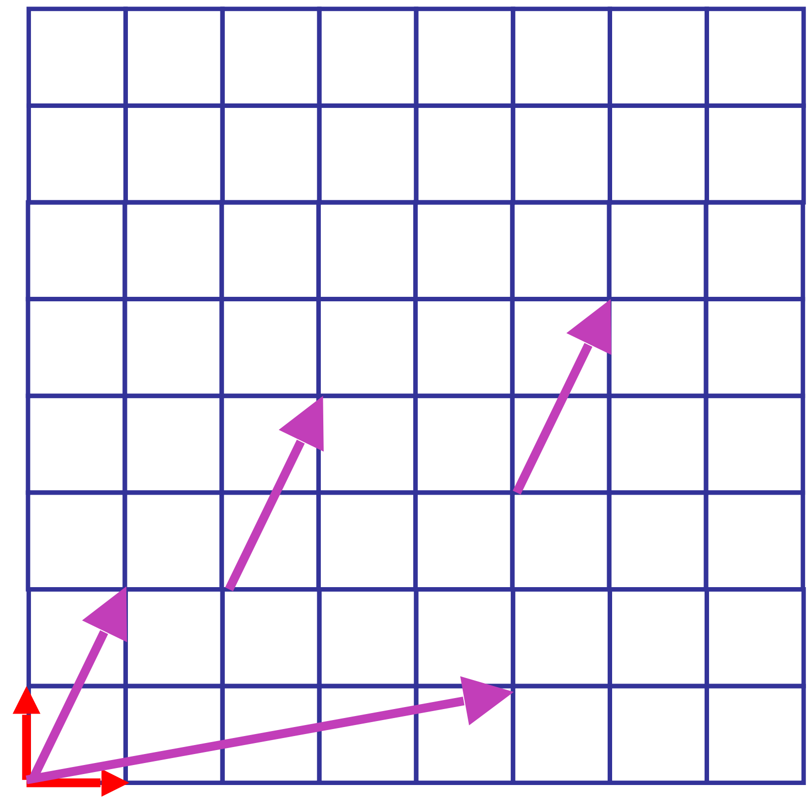
$$a$$

$$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$$

$$\mathbf{A} = \begin{array}{cc} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} & \begin{array}{l} \text{row 1} \\ \text{row 2} \\ \text{row 3} \end{array} \\ \begin{array}{ccc} \text{column 1} & \text{column 2} & \text{column 3} \end{array} & \end{array}$$

Vectors

- arrow: length and direction
  - oriented segment in nD space
- offset / displacement
  - **location** if given **origin**



Column vs. Row Vectors

- row vectors

$$\mathbf{a}_{row} = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$$

- column vectors

$$\mathbf{a}_{col} = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix}$$

- switch back and forth with Transpose

$$\mathbf{a}_{col}^{T} = \mathbf{a}_{row}$$

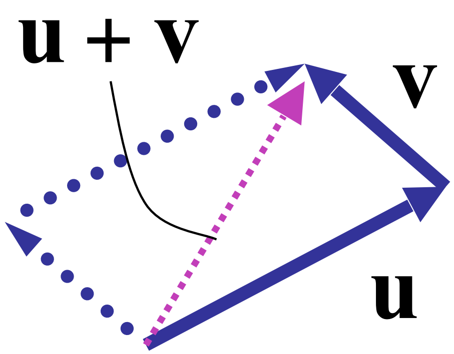
Vector Vector Addition

vector + vector = vector

- parallelogram rule
  - tail to head, complete the triangle

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

geometric



algebraic

$$\mathbf{u} + \mathbf{v} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \\ u_3 + v_3 \end{bmatrix}$$

examples:

$$(3,2) + (6,4) = \quad ????$$

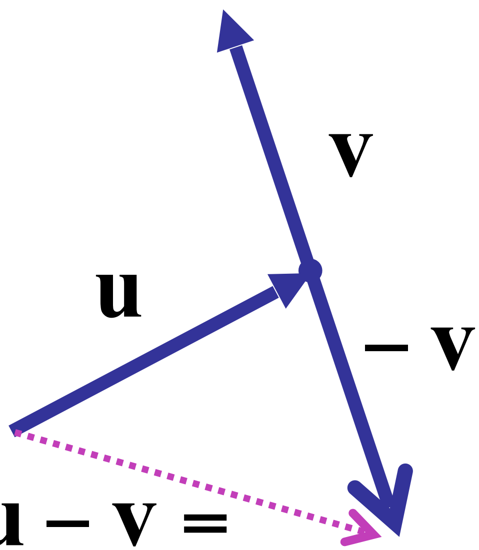
$$(2,5,1) + (3,1,-1) = \quad ????$$

Vector Vector Subtraction

vector - vector = vector

algebraic

geometric



$\mathbf{u} - \mathbf{v} =$
$\mathbf{u} + (-\mathbf{v})$

$$\mathbf{u} - \mathbf{v} = \begin{bmatrix} u_1 - v_1 \\ u_2 - v_2 \\ u_3 - v_3 \end{bmatrix}$$

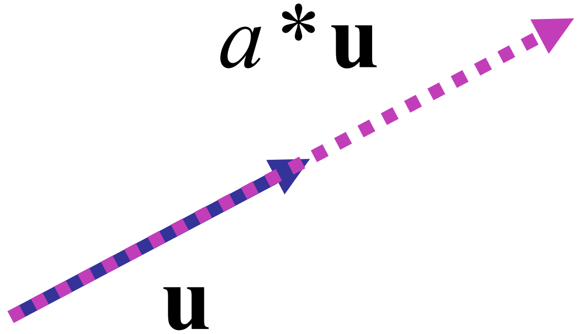
$(3,2) - (6,4) =$????

$(2,5,1) - (3,1,-1) =$????

Scalar Vector Multiplication

scalar * vector = vector

- vector is scaled



$$a * \mathbf{u} = (a * u_1, a * u_2, a * u_3)$$

$$2 * (3,2) = \qquad ????$$

$$.5 * (2,5,1) = \qquad ????$$

Vector Vector Multiplication

vector $\bullet$ vector = **scalar**

- **dot product**, aka **inner product**

$$\mathbf{u} \bullet \mathbf{v}$$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \bullet \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} =$$

????

Vector-Vector Multiplication

vector $\bullet$ vector = **scalar**

- dot product, aka inner product

$$\mathbf{u} \bullet \mathbf{v}$$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \bullet \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = (u_1 * v_1) + (u_2 * v_2) + (u_3 * v_3)$$

Vector-Vector Multiplication

vector $\bullet$ vector = scalar

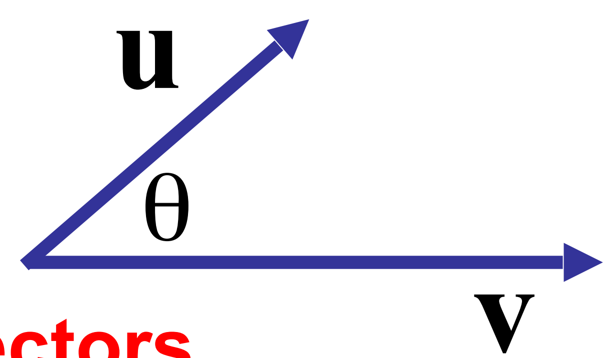
- dot product, aka inner product

$$\mathbf{u} \bullet \mathbf{v}$$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \bullet \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = (u_1 * v_1) + (u_2 * v_2) + (u_3 * v_3)$$

$$\mathbf{u} \bullet \mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\| \cos\theta$$

- geometric interpretation
  - lengths, angles
  - **can find angle between two vectors**

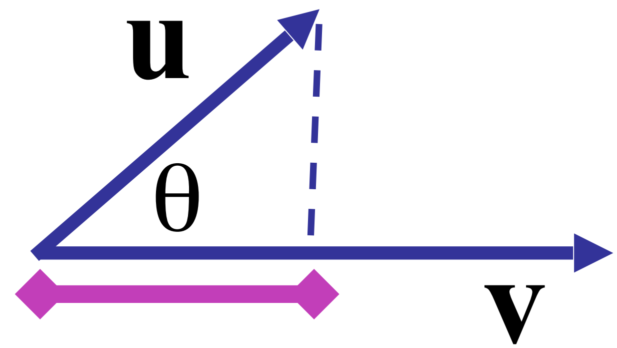


Dot Product Geometry

- can find **length of projection of u onto v**

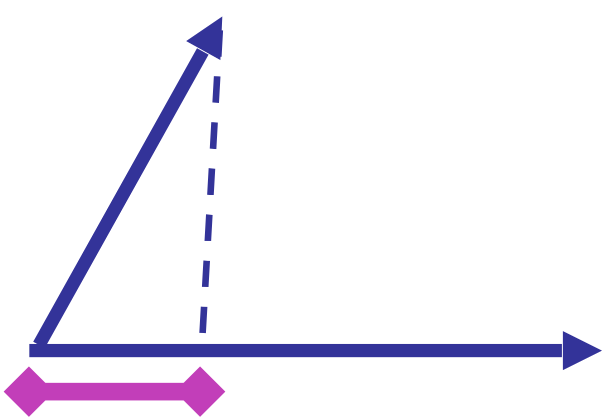
$$\mathbf{u} \bullet \mathbf{v} = \|\mathbf{u}\|\|\mathbf{v}\|\cos\theta$$

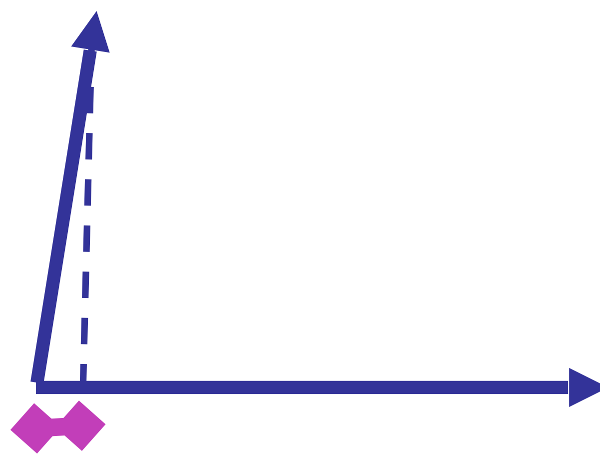
$$\|\mathbf{u}\|\cos\theta = \frac{\mathbf{u} \bullet \mathbf{v}}{\|\mathbf{v}\|}$$



$\|\mathbf{u}\|\cos\theta$

$$\mathbf{u} \bullet \mathbf{v} \rightarrow 0$$

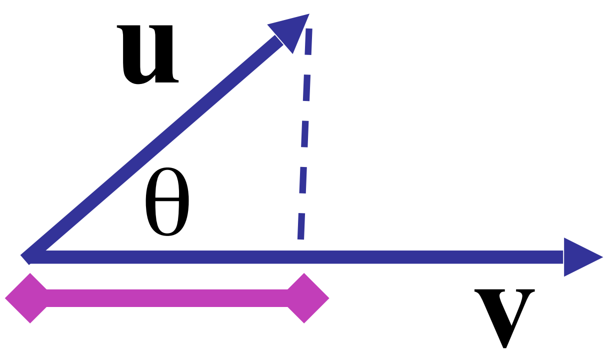




Dot Product Geometry

- can find **length of projection of u onto v**

$$\mathbf{u} \bullet \mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\| \cos\theta$$



$\|\mathbf{u}\| \cos\theta$

Dot Product Example

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \bullet \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = (u_1 * v_1) + (u_2 * v_2) + (u_3 * v_3)$$

$$\begin{bmatrix} 6 \\ 1 \\ 2 \end{bmatrix} \bullet \begin{bmatrix} 1 \\ 7 \\ 3 \end{bmatrix} =$$

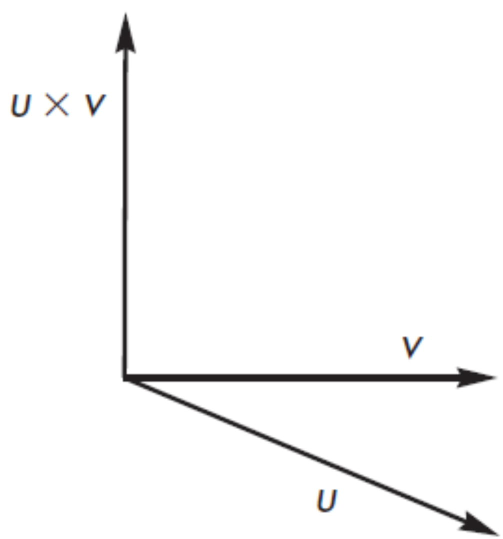
????

Vector Vector Multiplication, The Sequel

- vector **X** vector = vector
- **cross product**
  - Algebraic

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

  - Geometric



Cross product.

Vector-Vector Multiplication, The Sequel

vector **X** vector = vector

- cross product
  - algebraic

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

Vector-Vector Multiplication, The Sequel

- vector **X** vector = vector
- cross product
  - algebraic

$$\begin{array}{c} 3 \\[1.2em] 1 \\[1.2em] 2 \end{array} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

Vector-Vector Multiplication, The Sequel

vector X vector = vector

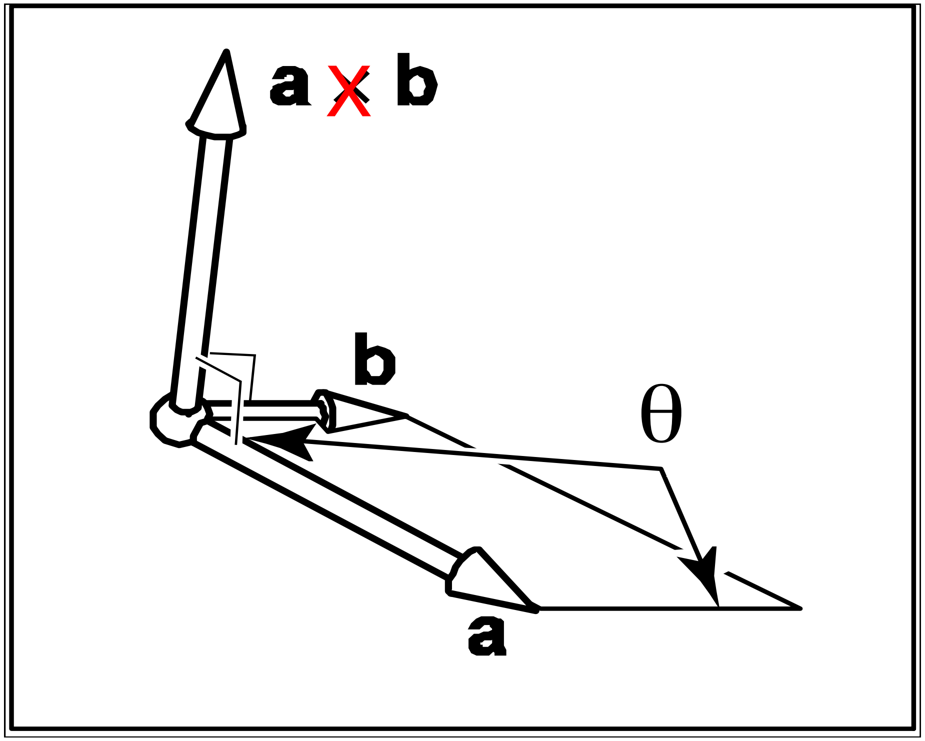
- cross product
  - **algebraic**

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

  - **geometric**

$$\|\mathbf{a} \times \mathbf{b}\| = \|a\| \|b\| \sin\theta$$

- $\|\mathbf{a} \times \mathbf{b}\|$ parallelogram area
- $\mathbf{a} \times \mathbf{b}$ perpendicular to parallelogram

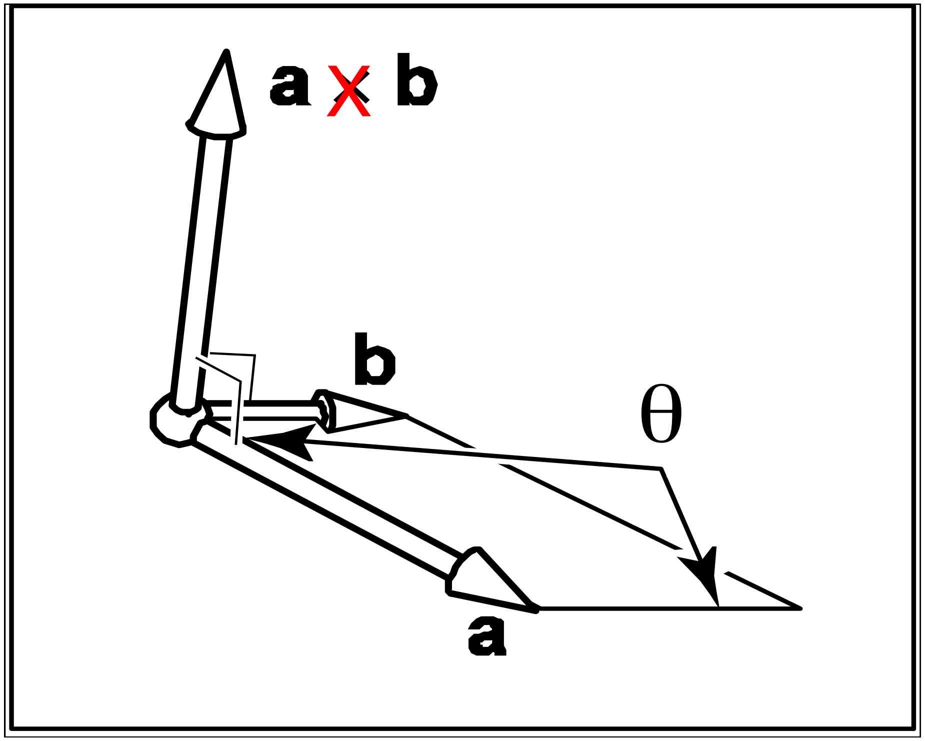


Vector-Vector Multiplication, The Sequel

vector **X** vector = vector

- **geometric**

$$\sin\theta = \frac{\|\mathbf{a} \times \mathbf{b}\|}{\|a\|\|b\|}$$



Normals

The University of New Mexico

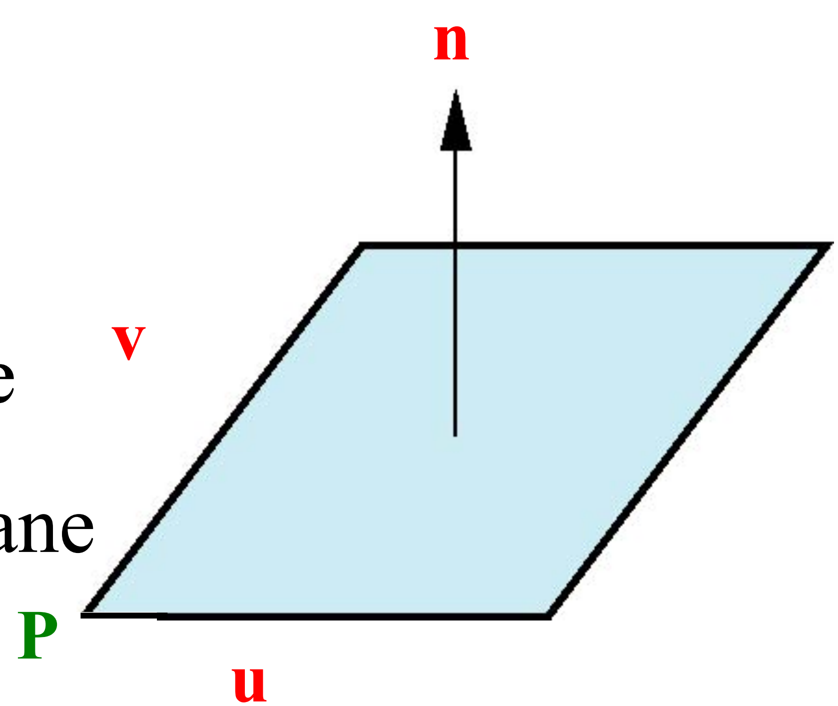
Every Plane has a vector **n** normal (**perpendicular, orthogonal**) to it

From **Point**-two **Vector** form $P(\alpha,\beta)=P+\alpha\mathbf{u}+\beta\mathbf{v}$, we know we can use the **cross product** × to find $\mathbf{n} = \mathbf{u} \times \mathbf{v}$ and the equivalent form using the **dot product** ·

$$(P(\alpha) - P) \cdot \mathbf{n} = 0$$

$P(\alpha)$ – parametric form for the line

$P(\alpha, \beta)$ – parametric form for the plane



Coordinate Systems and Frames

Chapter 4.3

Objectives

- Introduce concepts such as dimension and basis
- Introduce **coordinate systems** for representing **vectors Spaces** and **frames** for representing affine **Spaces**
- Discuss change of **frames** and bases
- Introduce **homogeneous coordinates**

Linear Independence

A set of **vectors** v_1, v_2, …, v_n is *linearly independent* if

$$\alpha_1 v_1+\alpha_2 v_2+ .. \alpha_n v_n=0 \quad \textbf{iff} \quad \alpha_1=\alpha_2=\ldots\alpha_n=0$$

If a set of **vectors** is **linearly independent**, we cannot represent one in terms of the others

If a set of **vectors** is **linearly dependent**, as least **one** can be written in terms of the **others**

Dimension

In a **Vector Space**, the maximum number of **linearly independent vectors** is fixed and is called the ***dimension* of the space**

In an *n*-dimensional space, any set of **n linearly independent vectors** form a ***basis* for the space**

Given a basis $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$, any **vector** $\boldsymbol{v}$ can be written as

$$\boldsymbol{v} = \alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2 + \ldots + \alpha_n \boldsymbol{v}_n$$

where the $\{\alpha_i\}$ are unique

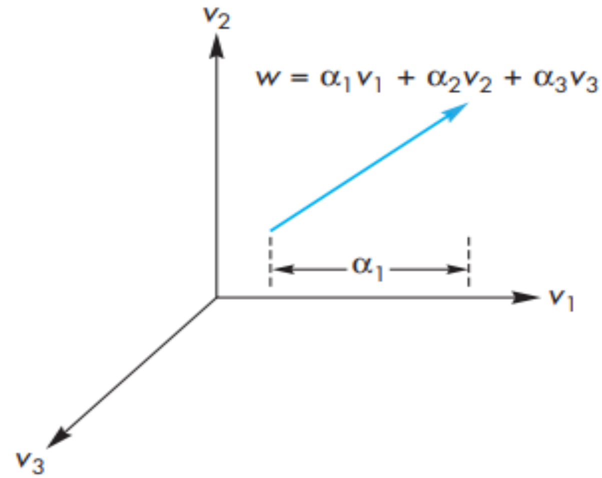
Representation

Until now we have been able to work with geometric entities **without using any frame of reference, such as a coordinate system**

Need a **frame of reference** to relate **Points** and objects to our physical world.

- For example, **where is a Point?** Can't answer without a **reference system**
- **World coordinates**
- **Camera coordinates**

Coordinate Systems



Vector derived from three basis vectors.

- Consider a basis $\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_n}$
- A **vector** is written $\boldsymbol{v} = \alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} + \ldots + \alpha_n \boldsymbol{v_n}$
- The list of scalars $\{\alpha_1, \alpha_2, \ldots \alpha_n\}$ is the *representation* of $\boldsymbol{v}$ with respect to the given **basis**
- We can write the representation as a row or column array of scalars

$$\mathbf{a} = [\alpha_1 \;\; \alpha_2 \;\; \ldots \;\; \alpha_n]^T = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \alpha_n \end{bmatrix}$$

Example

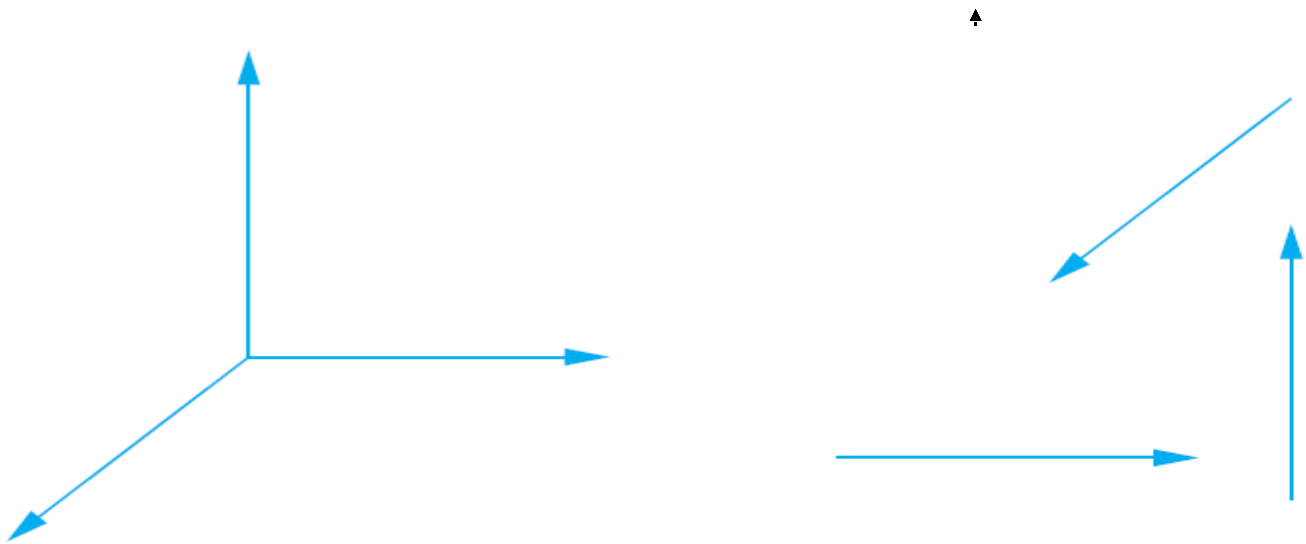
$$\mathbf{v} = 2\mathbf{v}_1 + 3\mathbf{v}_2 - 4\mathbf{v}_3$$

$$\mathbf{a} = [2 \quad 3 \quad -4]^T$$

- Note that this representation is with respect to a **particular basis**
- For example, in **OpenGL** we start by representing **vectors** using the **object basis** but later the system needs a representation in terms of the **camera or eye basis**

Coordinate Systems

- Which is correct?



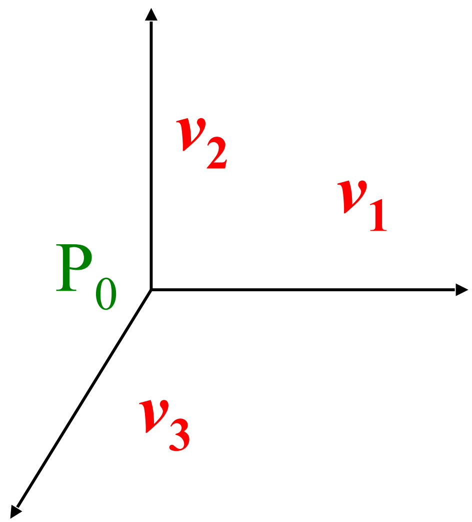
- Both are because **vectors** have no fixed location

equivalent

Frames

A **coordinate system** is insufficient to represent **Points**

If we work in an **affine space** we can add a single Point, the *origin* P_0, to the **basis vectors** v_1, v_2, v_3 to form a ***frame***



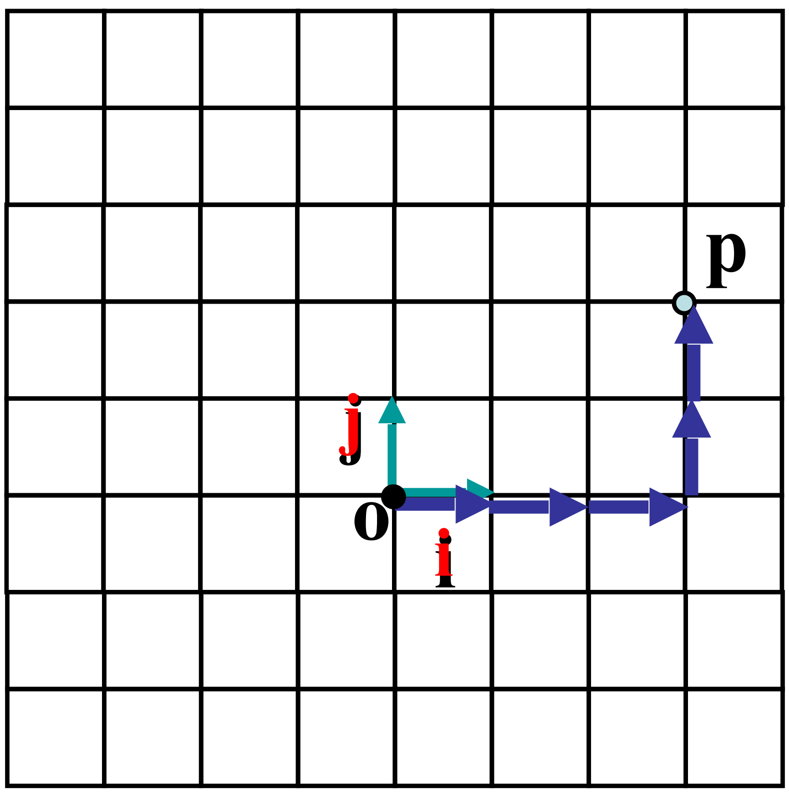
Basis vectors and Origins

coordinate system: just **basis vectors**

– can only specify offset: **vectors**

coordinate frame: **basis vectors i**, **j** and origin **o**

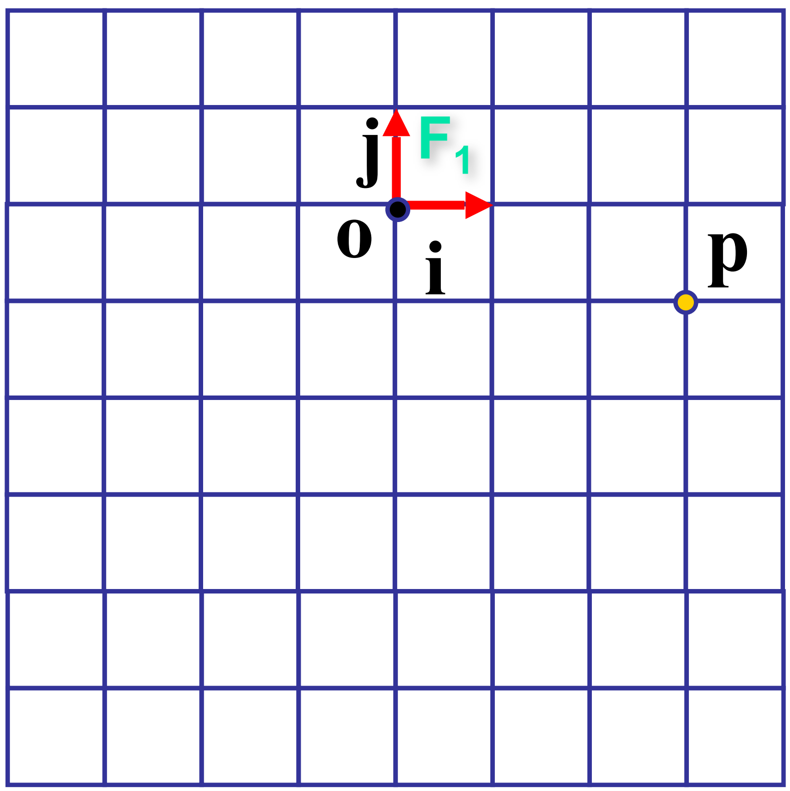
– can specify location as well as offset: Points



$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

$$\mathbf{p} = \mathbf{o} + 3\mathbf{i} + 2\mathbf{j}$$

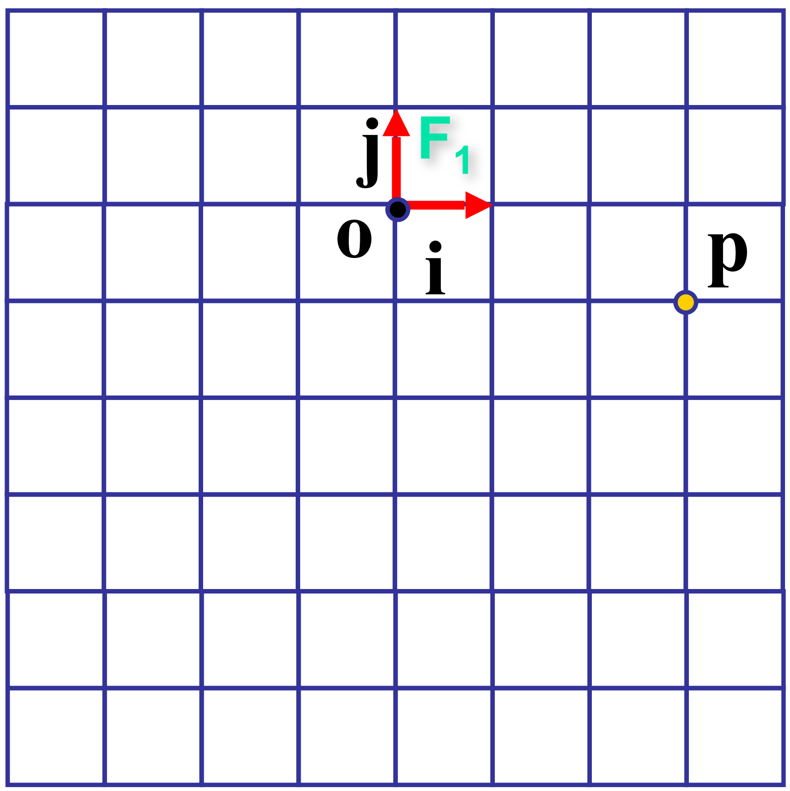
Working with **Frames**



$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

F_1 $\quad$ **p = (?,?)**

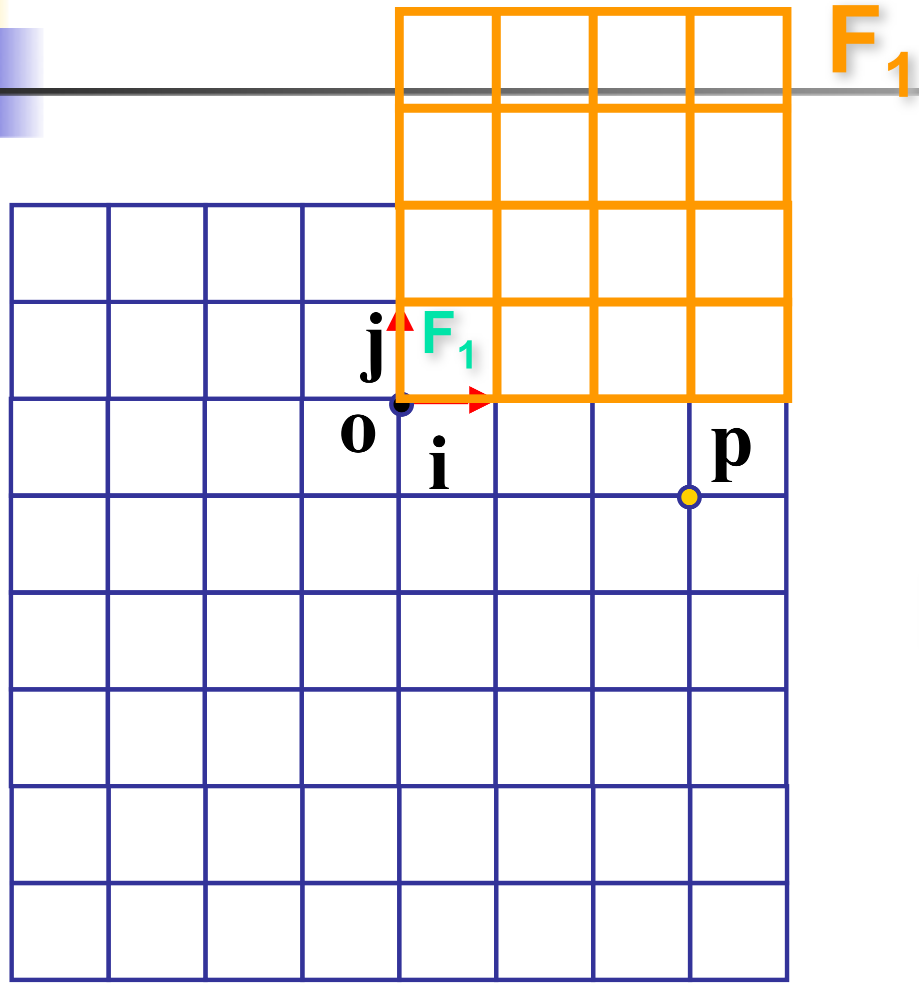
Working with **Frames**



$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

F_1 $\quad$ $\mathbf{p} = (3,-1)$

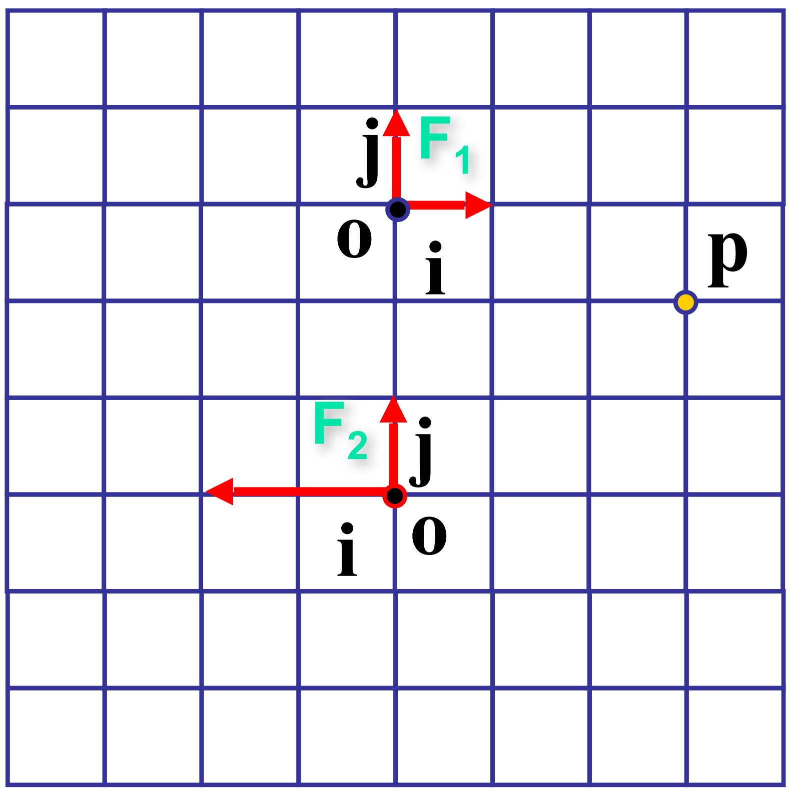
Working with **Frames**



$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

F_1 **p = (3,-1)**

Working with **Frames**

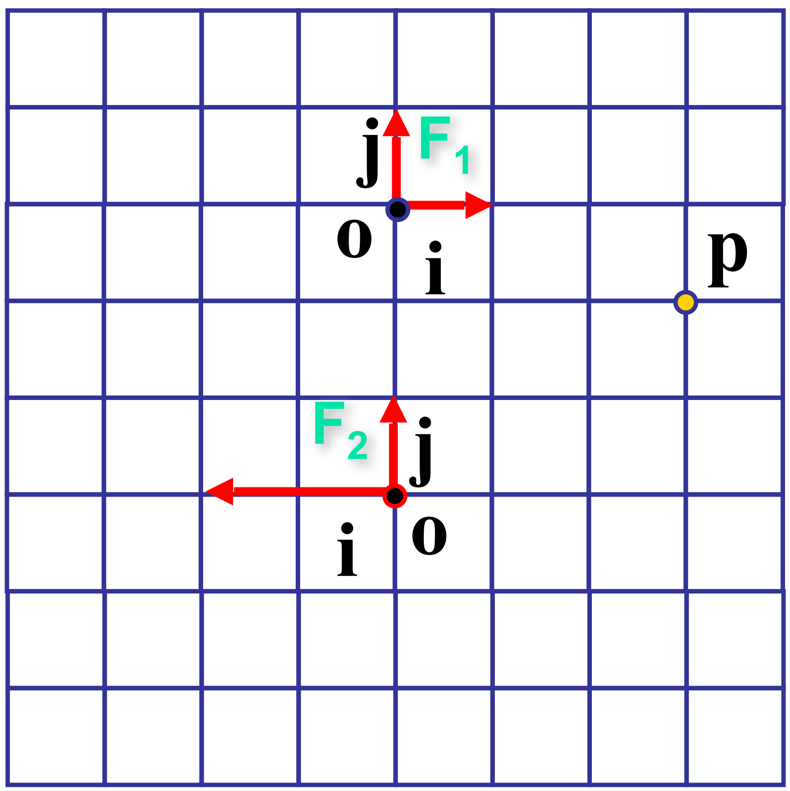


$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

F_1 $\quad$ $\mathbf{p} = (3,-1)$

F_2 $\quad$ $\mathbf{p} = (?,?)$

Working with **Frames**

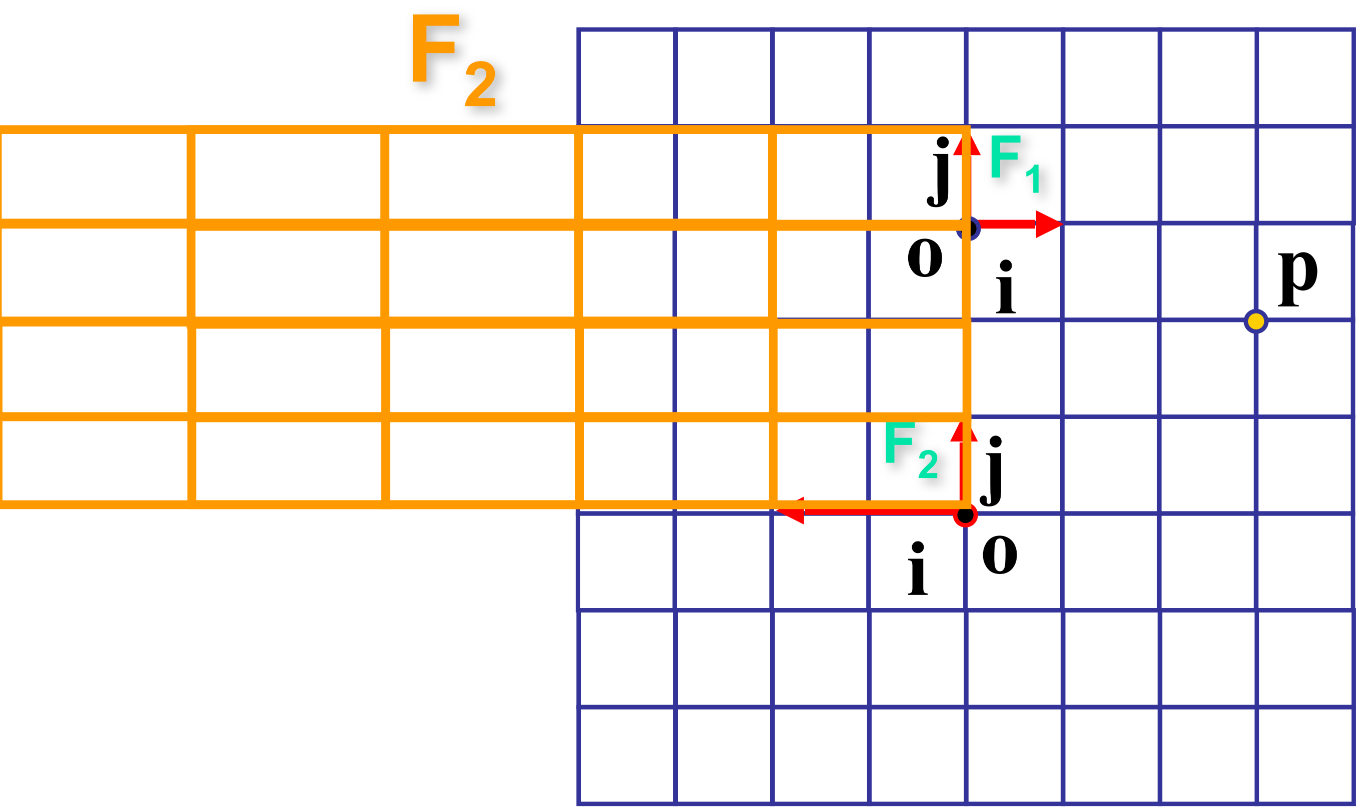


$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

F_1 p = (3,-1)

F_2 p = (-1.5,2)

Working with **Frames**

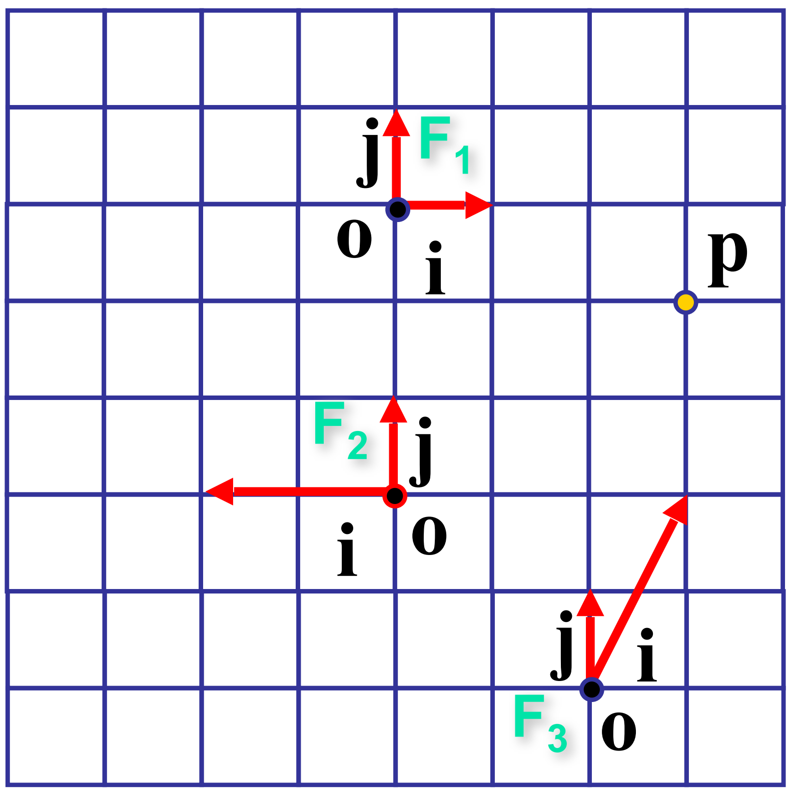


$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

F_1 p = (3,-1)

F_2 p = (-1.5,2)

Working with **Frames**



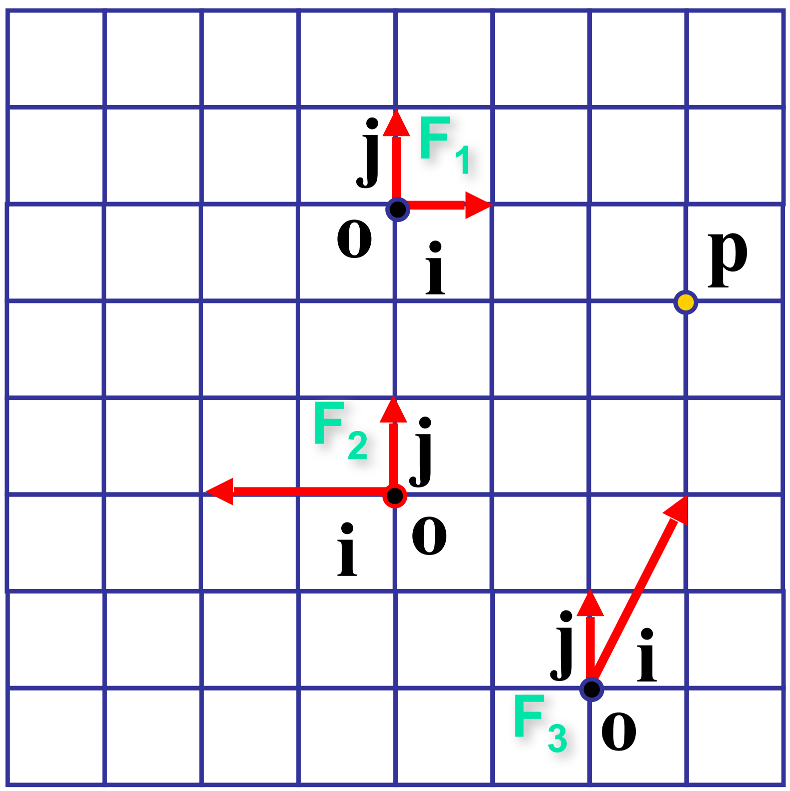
$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

F_1 p = (3,-1)

F_2 p = (-1.5,2)

F_3 p = (?,?)

Working with Frames



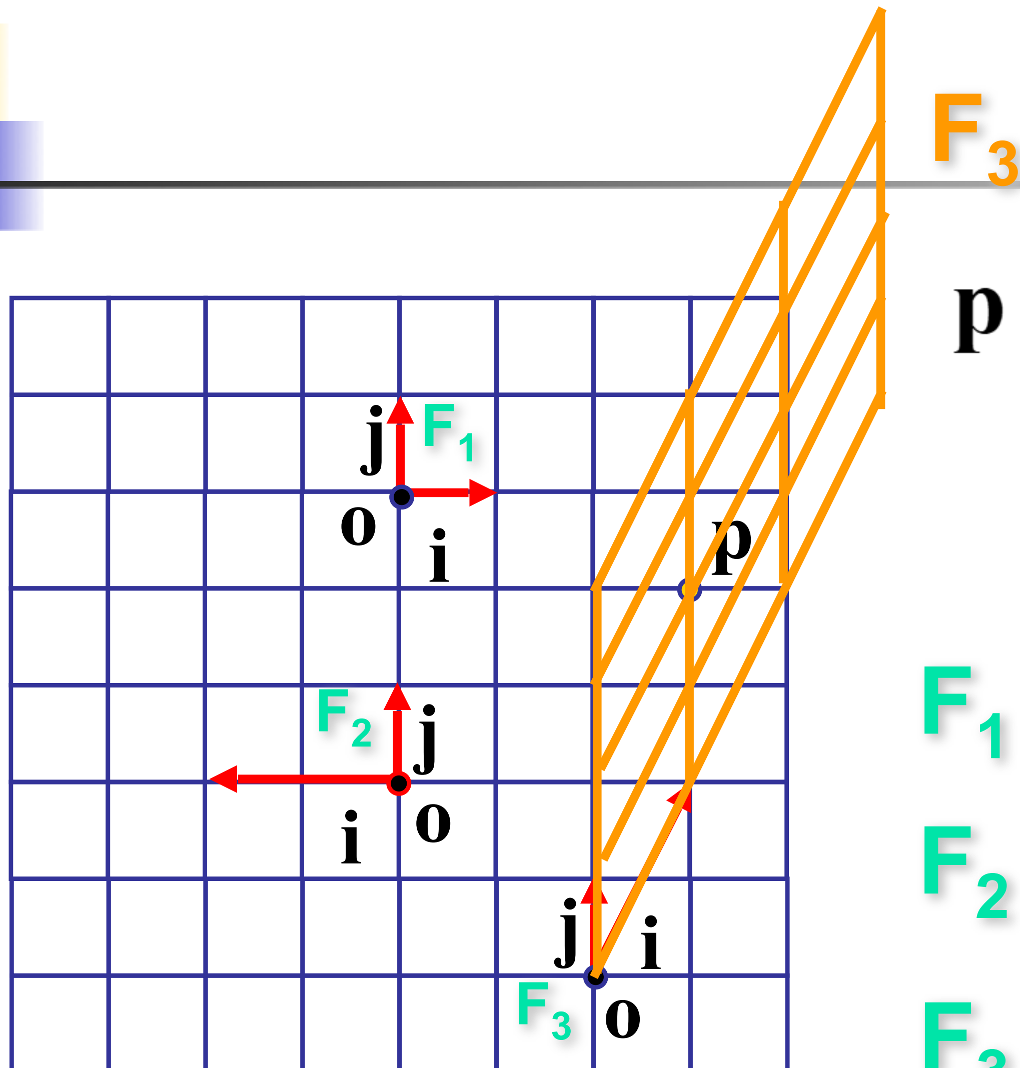
$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

F_1 p = (3,-1)

F_2 p = (-1.5,2)

F_3 p = (1,2)

Working with **Frames**



$$\mathbf{p} = \mathbf{o} + x\,\mathbf{i} + y\,\mathbf{j}$$

F1 **p = (3,-1)**

F2 **p = (-1.5,2)**

F3 **p = (1,2)**

Named Coordinate Frames

$$\mathbf{p} = \mathbf{o} + a\mathbf{x} + b\mathbf{y} + c\mathbf{z}$$

- Origin and **basis vectors**
- pick **canonical** frame of reference
  - then don't have to store Origin, **basis vectors**
  - just $\mathbf{p} = (a, b, c)$
  - convention: **Cartesian orthonormal**

  - handy to specify others as needed
  - really common ones given names in **CG**
    - **object**, **world**, **camera**, **screen**, ...

Representation in a Frame

- **Frame** determined by (P_0, $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$)
- Within this **frame**, every **Vector** can be written as

$$\boldsymbol{v} = \alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} + \ldots + \alpha_n \boldsymbol{v_n}$$

- Every **Point** can be written as

$$P = P_0 + \beta_1 \boldsymbol{v_1} + \beta_2 \boldsymbol{v_2} + \ldots + \beta_n \boldsymbol{v_n}$$

Confusing Points and Vectors

Consider the Point and the **Vector**

$$P = P_0 + \beta_1 \boldsymbol{v_1} + \beta_2 \boldsymbol{v_2} + \ldots + \beta_n \boldsymbol{v_n}$$

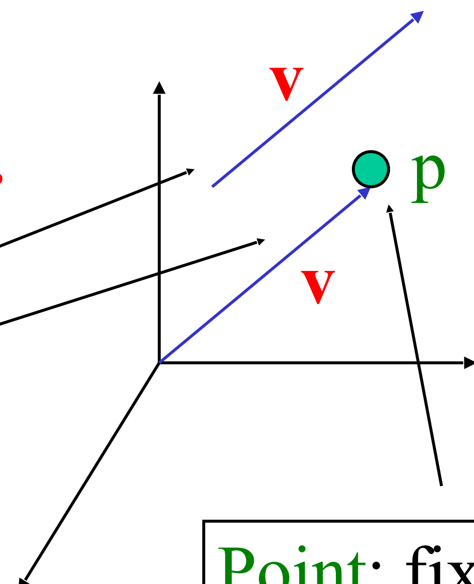
$$\boldsymbol{v} = \alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} + \ldots + \alpha_n \boldsymbol{v_n}$$

They appear to have the similar representations

$p=[\beta_1\ \beta_2\ \beta_3]$ $\quad\quad$ $\mathbf{v}=[\alpha_1\ \alpha_2\ \alpha_3]$

which confuses the Point with the **Vector**

A **Vector has no position**



Vector can be placed anywhere

Point: fixed

$\boldsymbol{v_1}$, $\boldsymbol{v_2}$, $\boldsymbol{v_3}$ are **basis vectors**

$v_1 = (1,0,0)^T\ v_2 = (0,1,0)^T\ v_3 = (0,0,1)^T$

A Single Representation

If we define $0 \cdot \mathbf{P} = \mathbf{0}$ and $1 \cdot \mathbf{P} = \mathbf{P}$ then we can write

$$\mathbf{v} = \alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2 + \alpha_3 \boldsymbol{v}_3 = [\alpha_1\ \alpha_2\ \alpha_3\ 0]\,[\boldsymbol{v}_1\ \boldsymbol{v}_2\ \boldsymbol{v}_3\ \mathrm{P}_0]^T$$

$$\mathrm{P} = \mathrm{P}_0 + \beta_1 \boldsymbol{v}_1 + \beta_2 \boldsymbol{v}_2 + \beta_3 \boldsymbol{v}_3 = [\beta_1\ \beta_2\ \beta_3\ 1]\,[\boldsymbol{v}_1\ \boldsymbol{v}_2\ \boldsymbol{v}_3\ \mathrm{P}_0]^T$$

Thus, we obtain the four-dimensional ***homogeneous coordinate*** **representation**

$$\mathbf{v} = [\alpha_1\ \alpha_2\ \alpha_3\ 0]^T$$

$$\mathrm{P} = [\beta_1\ \beta_2\ \beta_3\ 1]^T$$

Homogeneous coordinates avoid this difficulty by using a four dimensional representation for both points and vectors in three dimensions. In the frame specified by $\boxed{(v_1, v_2, v_3, P_0)}$, any point P can be written uniquely as

Frame:$(\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}, P_0)$

$$P = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + P_0.$$

$$P = \alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} + \alpha_3 \boldsymbol{v_3} + P_0$$

. Equivalently, we can say that P is represented by the column matrix

$$\mathbf{p} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{bmatrix}.$$

If we agree to define the "multiplication" of

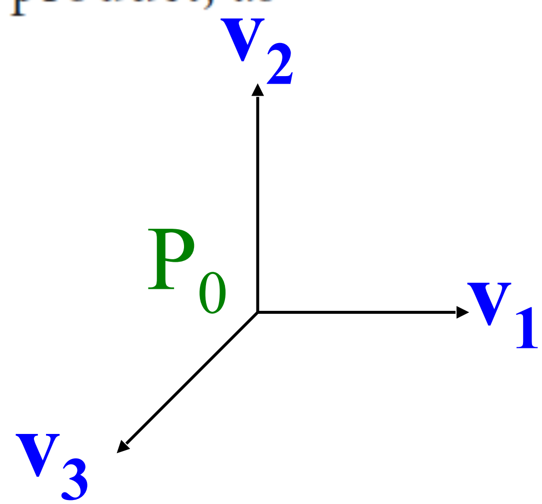
$$0 \cdot P = \mathbf{0},$$

$$1 \cdot P = P,$$

then we can express this relation formally, using a matrix product, as

$$P = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad 1]\,[\boldsymbol{v_1} \quad \boldsymbol{v_2} \quad \boldsymbol{v_3} \quad P_0]^T$$

$$P = [\,\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad 1\,] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}.$$



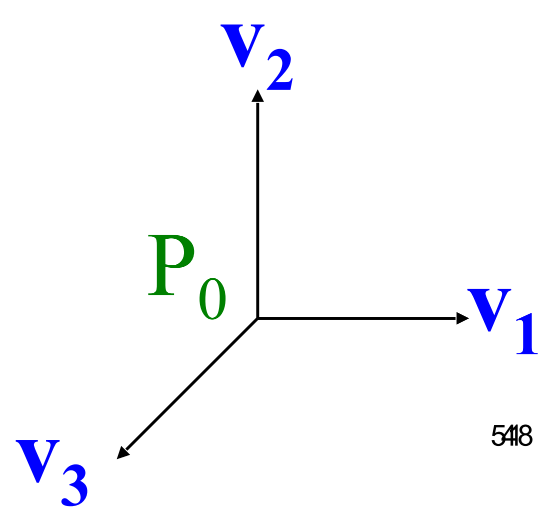
In the same frame, any vector w can be written as

$$\mathbf{w} = \delta_1 v_1 + \delta_2 v_2 + \delta_3 v_3$$

$$= [\ \delta_1 \quad \delta_2 \quad \delta_3 \quad 0\ ]^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}.$$

Thus, $\mathbf{w}$ can be represented by the column matrix

$$\mathbf{w} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ 0 \end{bmatrix}.$$



Homogeneous Coordinates

The University of New Mexico

The **homogeneous coordinates** form for a three-dimensional **Point** [x y z] is given as

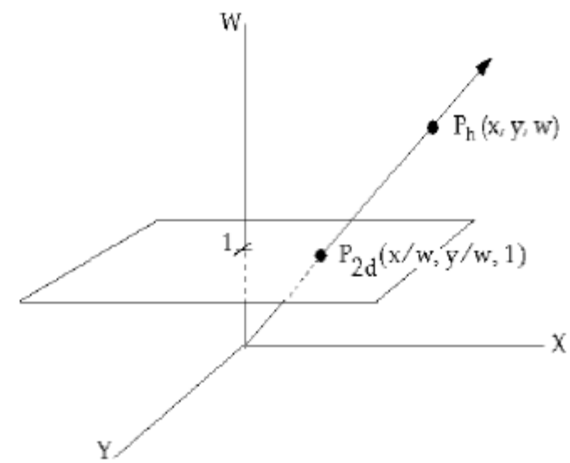
$\mathbf{P} = [x' \; y' \; z' \; w]^T = [wx \; wy \; wz \; w]^T$

We return to a three-dimensional **Point** (for $w \neq 0$) by

$x \leftarrow x'/w$

$y \leftarrow y'/w$

$z \leftarrow z'/w$



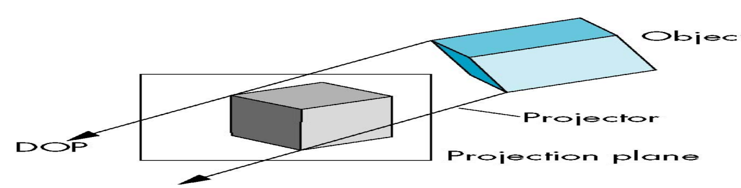
If $w=0$, the representation is that of a **Vector**

Note that **homogeneous coordinates** replaces **Points** in three dimensions by **lines through the origin in four dimensions**

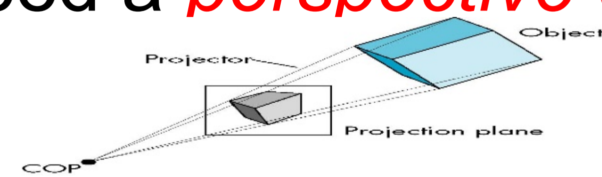
For $w=1$, the representation of a **Point** is [x y z 1]

Homogeneous Coordinates and Computer Graphics

- **Homogeneous coordinates** are key to all computer graphics systems
  - All standard transformations (rotation, translation, scaling) can be implemented with matrix multiplications using 4 x 4 matrices
  - Hardware pipeline works with 4 dimensional representations
  - For **orthographic viewing**, we can maintain $w=0$ for **Vectors** and $w=1$ for **Points**



  - For **perspective viewing** we need a *perspective division*



Change of Coordinate Systems

- Consider two representations of a the same **Vector** with respect to **two different bases**. The representations are

$$\mathbf{a} = [\alpha_1 \ \alpha_2 \ \alpha_3]$$
$$\mathbf{b} = [\beta_1 \ \beta_2 \ \beta_3]$$

where

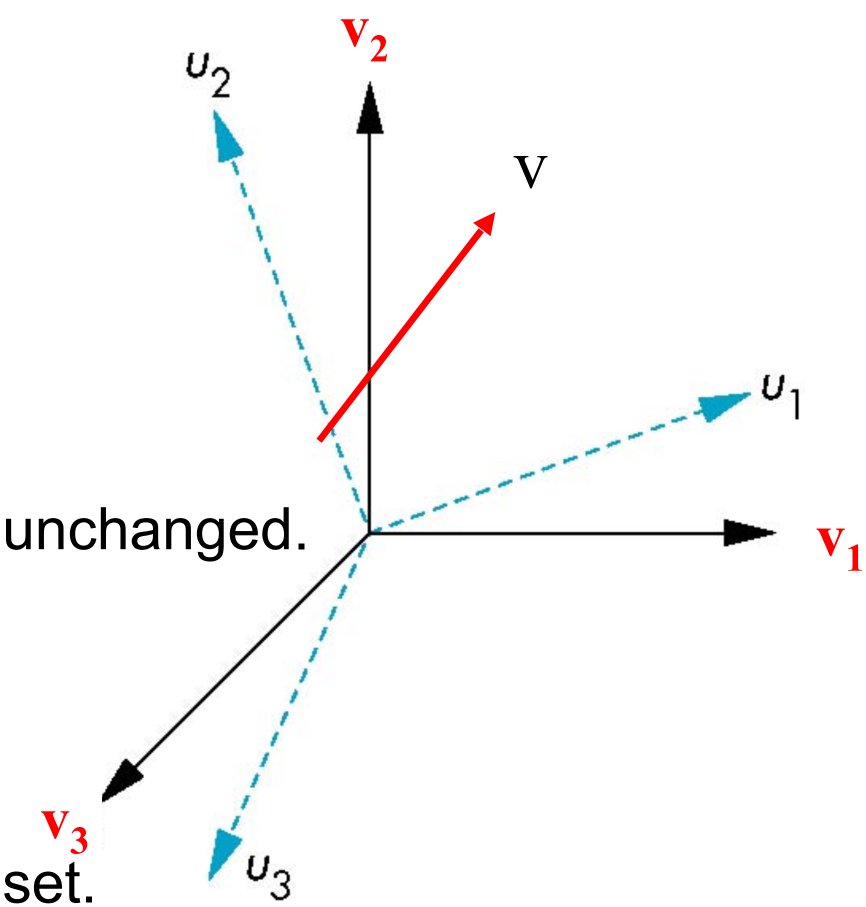
$$\mathbf{v} = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 = [\alpha_1 \ \alpha_2 \ \alpha_3] \ [v_1 \ v_2 \ v_3]^T$$

$$= \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 = [\beta_1 \ \beta_2 \ \beta_3] \ [u_1 \ u_2 \ u_3]^T$$

Representing second basis in terms of first

Each of the **basis vectors**, $\mathbf{u_1}$,$\mathbf{u_2}$, $\mathbf{u_3}$, are **Vectors** that can be represented in terms of the first basis

$$\mathbf{u_1} = \gamma_{11}\mathbf{v_1}+\gamma_{12}\mathbf{v_2}+\gamma_{13}\mathbf{v_3}$$
$$\mathbf{u_2} = \gamma_{21}\mathbf{v_1}+\gamma_{22}\mathbf{v_2}+\gamma_{23}\mathbf{v_3}$$
$$\mathbf{u_3} = \gamma_{31}\mathbf{v_1}+\gamma_{32}\mathbf{v_2}+\gamma_{33}\mathbf{v_3}$$



These changes in **basis** leave the **origin** unchanged.

We can use them to represent

rotation and **scaling** of a set

of **basis vectors** to derive another basis set.

γ (Eepsilon)

Matrix Form

The coefficients define a 3 x 3 matrix

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

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \mathbf{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix},$$

$$\mathbf{u}=\mathbf{M}\mathbf{v}$$

and the **bases** can be related by

$$\mathbf{v}=\mathbf{M}^T\mathbf{u}$$

Example Change of Representation

Suppose that we have a vector w whose representation in some basis is

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

We can denote the three basis vectors as v_1, v_2, and v_3. Hence,

$$w = v_1 + 2v_2 + 3v_3.$$

Now suppose that we want to make a new basis from the three vectors v_1, v_2, and v_3 where

$$u_1 = v_1,$$
$$u_2 = v_1 + v_2,$$
$$u_3 = v_1 + v_2 + v_3.$$

The matrix $\mathbf{M}$ is

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

$$\mathbf{T} = (\mathbf{M}^T)^{-1}$$
$$= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}^{-1}$$
$$= \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}.$$

In the new system, the representation of w is

$$\mathbf{b} = \mathbf{Ta} = \begin{bmatrix} -1 \\ -1 \\ 3 \end{bmatrix}.$$

That is,

$$w = -u_1 - u_2 + 3u_3.$$

$$\begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix}$$

The matrix that converts a representation in v_1, v_2, and v_3 to one in which the basis vectors are u_1, u_2, and u_3 is

Change of Frames

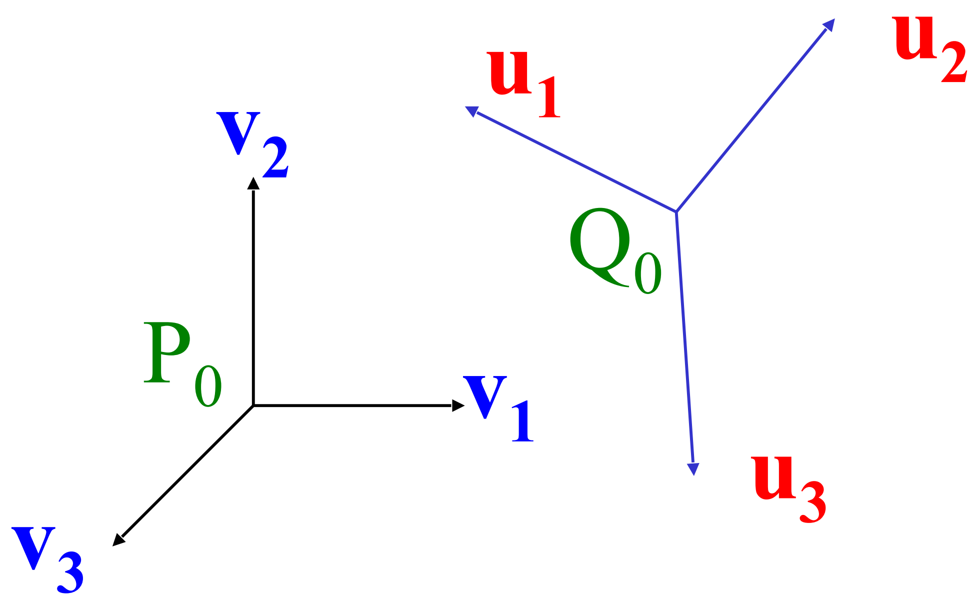
The University of New Mexico

We can apply a similar process in **homogeneous coordinates** to the representations of both **Points** and **Vectors**

Consider two frames:

$(P_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$

$(Q_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$



Any **Point** or **Vector** can be represented in **either frame**

We can represent $Q_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ in terms of $P_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$

The University of New Mexico

Representing One Frame in Terms of the Other

Extending what we did with **change of bases**

$$
\begin{aligned}
\mathbf{u}_1 &= \gamma_{11}\mathbf{v}_1+\gamma_{12}\mathbf{v}_2+\gamma_{13}\mathbf{v}_3\\
\mathbf{u}_2 &= \gamma_{21}\mathbf{v}_1+\gamma_{22}\mathbf{v}_2+\gamma_{23}\mathbf{v}_3\\
\mathbf{u}_3 &= \gamma_{31}\mathbf{v}_1+\gamma_{32}\mathbf{v}_2+\gamma_{33}\mathbf{v}_3\\
Q_0 &= \gamma_{41}\mathbf{v}_1+\gamma_{42}\mathbf{v}_2+\gamma_{43}\mathbf{v}_3 + P_0
\end{aligned}
$$

defining a 4 x 4 matrix

$$
\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}
$$

Working with Representations

The University of New Mexico

Within the **two frames** any **Point** or **Vector** has a representation of the same form

$\mathbf{a}=[\alpha_1\ \alpha_2\ \ \alpha_3\ \ \alpha_4\,]$ in the **first frame**
$\mathbf{b}=[\beta_1\ \beta_2\ \ \beta_3\ \ \beta_4\,]$ in the **second frame**

where $\alpha_4=\beta_4=1$ for **Points** and $\alpha_4=\beta_4=0$ for **Vectors** and

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

The matrix $\mathbf{M}$ is 4 x 4 and specifies an **affine transformation** in **homogeneous coordinates**

The University of New Mexico

Affine Transformations

- Every linear transformation is equivalent to a change in frames

- Every affine transformation preserves lines

- However, an affine transformation has only 12 *degrees of freedom* because 4 of the elements in the matrix are fixed and are a subset of all possible 4 x 4 linear transformations

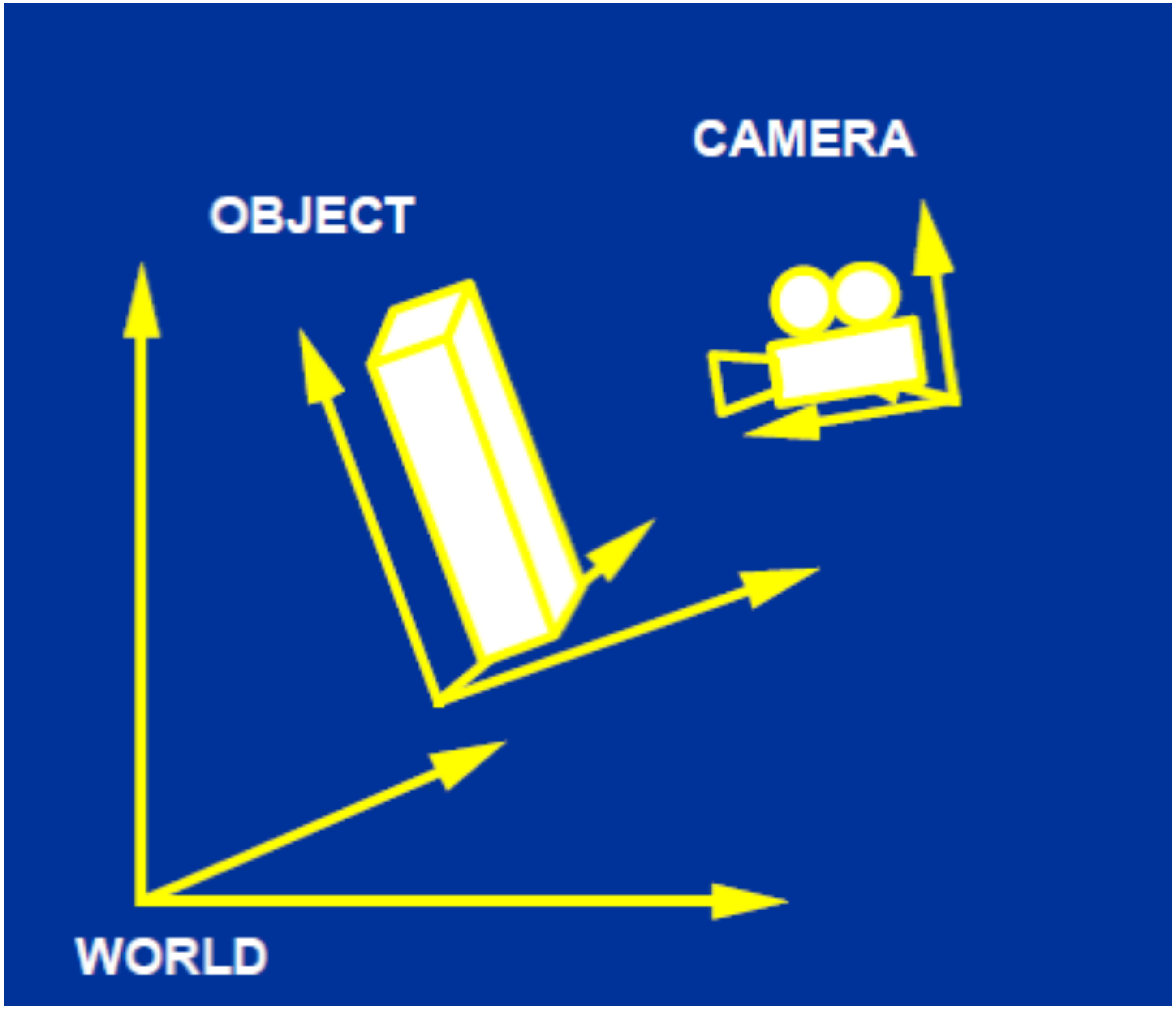
$$\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

The World and Camera Frames

The University of New Mexico

- When we work with representations, we work with **n-tuples** or arrays of scalars
- **Changes in frame** are then defined by 4 x 4 matrices
- In OpenGL, the **base frame** that we start with is the **world frame**
- Eventually we represent **entities** in the **camera frame** by changing the **world representation** using the ***model-view matrix***
- Initially these **frames** are the same ($\mathbf{M}=\mathbf{I}$)

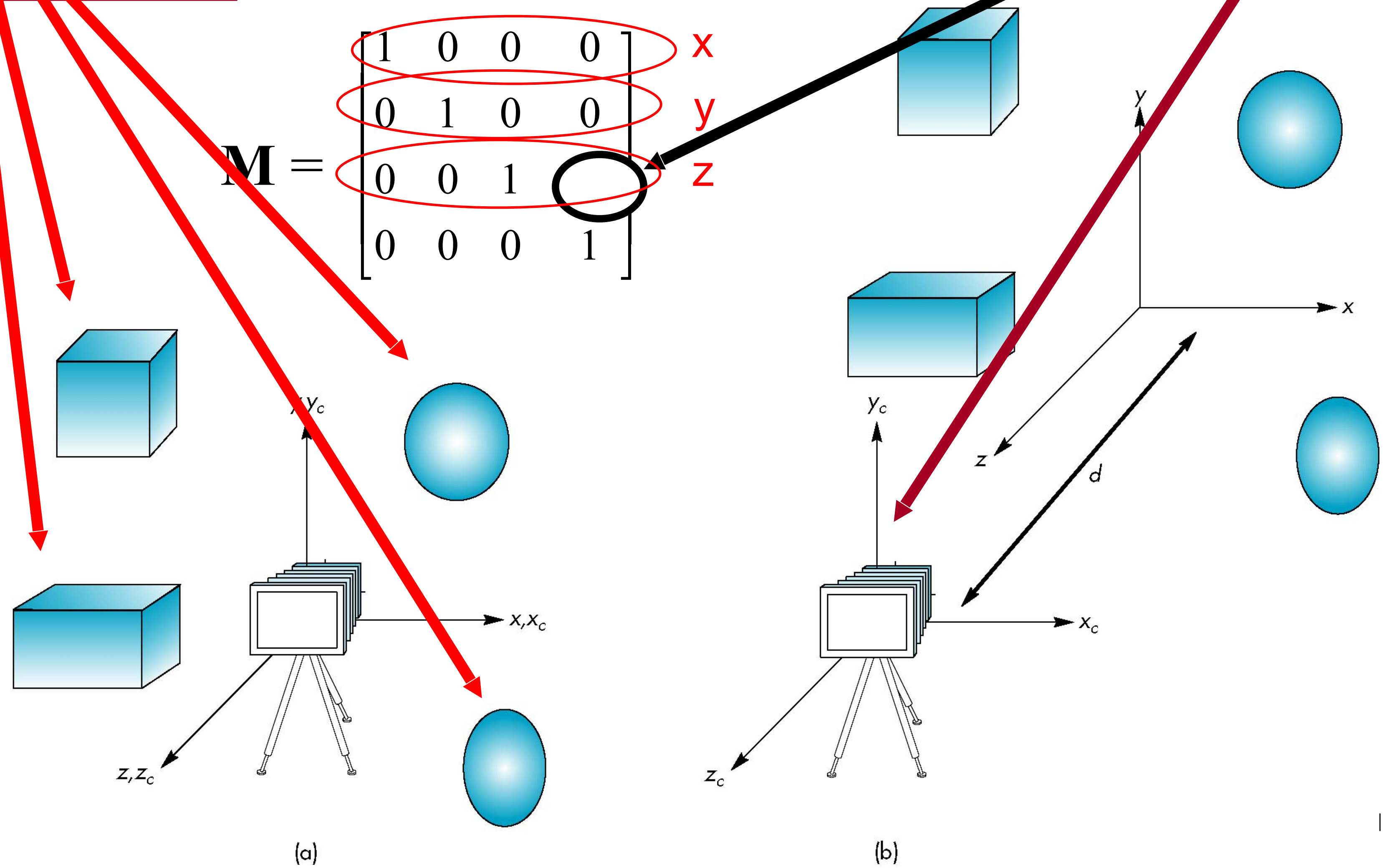
The World, the Object and Camera Frames



Moving the Camera

If objects are on both sides of z=0, we must move camera frame

$$\mathbf{M} = \begin{array}{c} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{array}{c} x \\ y \\ z \\ \\ \end{array}$$



Matrix Matrix Addition

matrix + matrix = matrix

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} + \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} = \begin{bmatrix} n_{11} + m_{11} & n_{12} + m_{12} \\ n_{21} + m_{21} & n_{22} + m_{22} \end{bmatrix}$$

- example

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} + \begin{bmatrix} -2 & 5 \\ 7 & 1 \end{bmatrix} = \begin{bmatrix} & \\ & \end{bmatrix}$$

????

Scalar Matrix Multiplication

scalar * matrix = matrix

$$a\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = \begin{bmatrix} a * m_{11} & a * m_{12} \\ a * m_{21} & a * m_{22} \end{bmatrix}$$

- example

$$3\begin{bmatrix} 2 & 4 \\ 1 & 5 \end{bmatrix} = \begin{bmatrix} & \\ & \end{bmatrix}$$

????

Matrix Matrix Multiplication

- can only multiply (n,**k**) by (**k**,m):
  number of left cols = number of right rows
  - legal

$$\begin{bmatrix} a & b & c \\ e & f & g \end{bmatrix} \begin{bmatrix} h & i \\ j & k \\ l & m \end{bmatrix}$$

  - **undefined**

$$\begin{bmatrix} a & b & c \\ e & f & g \\ o & p & q \end{bmatrix} \begin{bmatrix} h & i \\ j & k \end{bmatrix}$$

Matrix Matrix Multiplication

- row by column

$$\begin{bmatrix} \boxed{m_{11} \quad m_{12}} \\ m_{21} \quad m_{22} \end{bmatrix} \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} = \begin{bmatrix} \boxed{p_{11}} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

$$p_{11} = m_{11}n_{11} + m_{12}n_{21}$$

Matrix-Matrix Multiplication

- row by column

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

$$p_{11} = m_{11}n_{11} + m_{12}n_{21}$$

$$p_{21} = m_{21}n_{11} + m_{22}n_{21}$$

Matrix-Matrix Multiplication

- row by column

$$\begin{bmatrix} \boxed{m_{11} \quad m_{12}} \\ m_{21} \quad m_{22} \end{bmatrix} \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} = \begin{bmatrix} p_{11} & \boxed{p_{12}} \\ p_{21} & p_{22} \end{bmatrix}$$

$$p_{11} = m_{11}n_{11} + m_{12}n_{21}$$

$$p_{21} = m_{21}n_{11} + m_{22}n_{21}$$

$$p_{12} = m_{11}n_{12} + m_{12}n_{22}$$

Matrix-Matrix Multiplication

- row by column

$$\begin{bmatrix} m_{11} & m_{12} \\ \boxed{m_{21} \quad m_{22}} \end{bmatrix} \begin{bmatrix} n_{11} & \boxed{n_{12}} \\ n_{21} & \boxed{n_{22}} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & \boxed{p_{22}} \end{bmatrix}$$

$$p_{11} = m_{11}n_{11} + m_{12}n_{21}$$

$$p_{21} = m_{21}n_{11} + m_{22}n_{21}$$

$$p_{12} = m_{11}n_{12} + m_{12}n_{22}$$

$$p_{22} = m_{21}n_{12} + m_{22}n_{22}$$

Matrix-Matrix Multiplication

- row by column

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

$$p_{11} = m_{11}n_{11} + m_{12}n_{21}$$

$$p_{21} = m_{21}n_{11} + m_{22}n_{21}$$

$$p_{12} = m_{11}n_{12} + m_{12}n_{22}$$

$$p_{22} = m_{21}n_{12} + m_{22}n_{22}$$

- noncommutative: **AB != BA**

Matrix Vector Multiplication

- **points** as column **vectors**: premultiply

$$\begin{bmatrix} x' \\ y' \\ z' \\ h' \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ h \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{M}\mathbf{p}$$

$$\begin{bmatrix} x' & y' & z' & h' \end{bmatrix} = \begin{bmatrix} x & y & z & h \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}^T$$

$$\mathbf{p}'^T = \mathbf{p}^T \mathbf{M}^T$$

Matrices

- **transpose**

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}^T = \begin{bmatrix} m_{11} & m_{21} & m_{31} & m_{41} \\ m_{12} & m_{22} & m_{32} & m_{42} \\ m_{13} & m_{23} & m_{33} & m_{43} \\ m_{14} & m_{24} & m_{34} & m_{44} \end{bmatrix}$$

- **identity**

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

- **inverse**

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

  - **not all matrices are invertible**

Matrices and Linear Systems

- linear system of n equations, n unknowns

$$3x + 7y + 2z = 4$$
$$2x - 4y - 3z = -1$$
$$5x + 2y + z = 1$$

- matrix form **Ax=b**

$$\begin{bmatrix} 3 & 7 & 2 \\ 2 & -4 & -3 \\ 5 & 2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \\ 1 \end{bmatrix}$$

Trigonometry

Trigonometric Functions

The following ratios are for right angle trigonometry. The angle must be *acute* (angle is less than 90°).

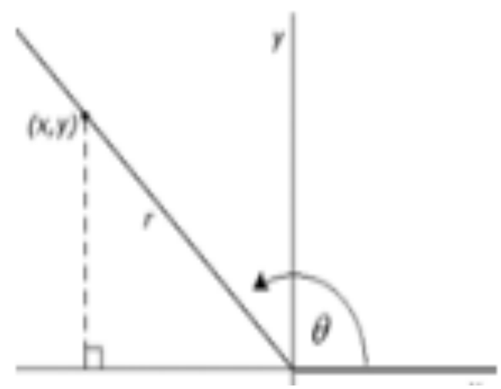
$$\sin\theta = \frac{\text{opp}}{\text{hyp}} \qquad \cos\theta = \frac{\text{adj}}{\text{hyp}} \qquad \tan\theta = \frac{\text{opp}}{\text{adj}}$$

$$\csc\theta = \frac{\text{hyp}}{\text{opp}} \qquad \sec\theta = \frac{\text{hyp}}{\text{adj}} \qquad \cot\theta = \frac{\text{adj}}{\text{opp}}$$

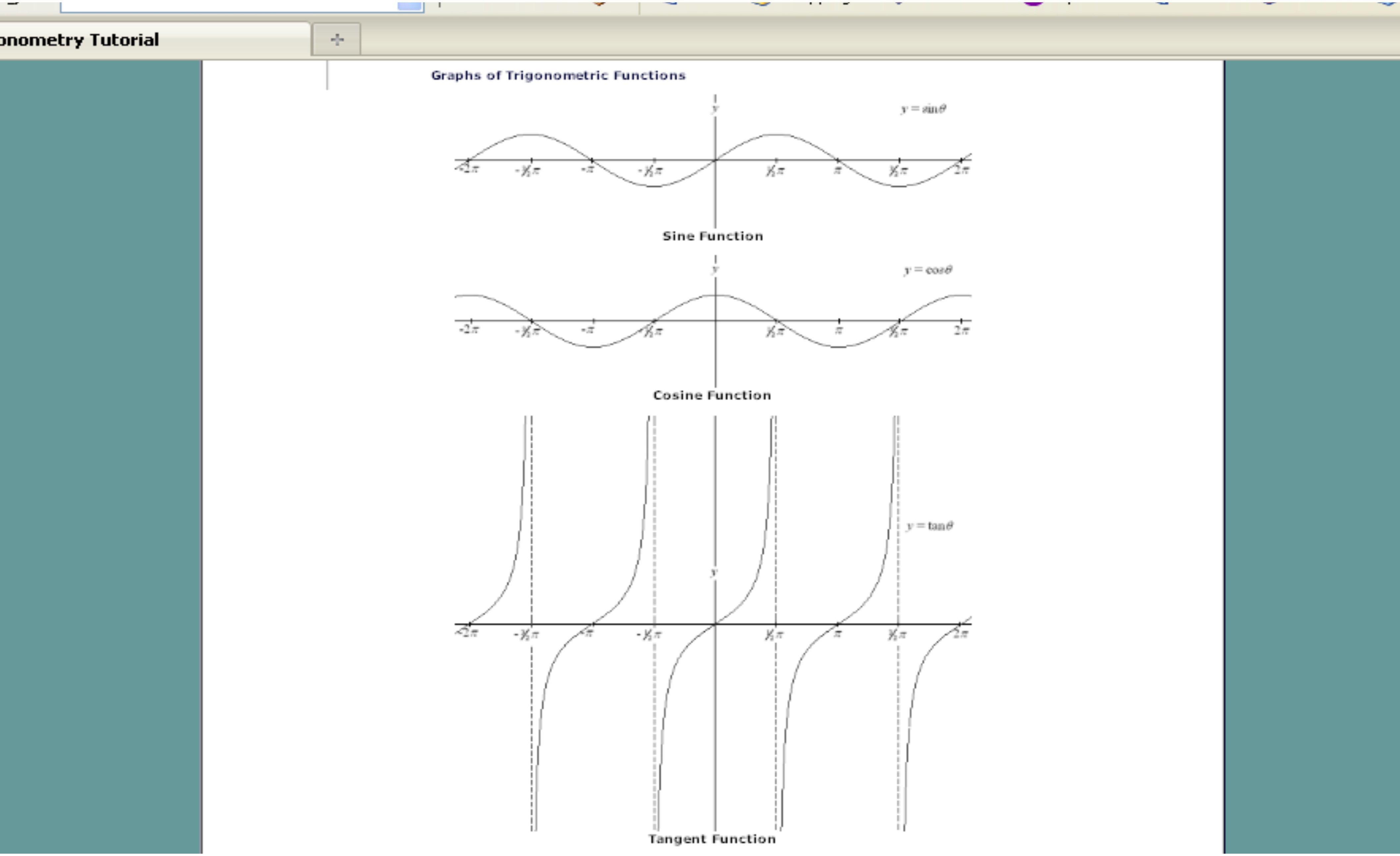
For angles that are *obtuse* (angle is greater than 90°) or negative, we use the following trigonometric ratios. The x and y variables are the values of the x and y coordinates, respectively. The r variable represents the distance from the origin, to the point (x,y). This value can be found using the Pythagorean theorem.

$$\sin\theta = \frac{y}{r} \qquad \cos\theta = \frac{x}{r} \qquad \tan\theta = \frac{y}{x}$$

$$\csc\theta = \frac{r}{y} \qquad \sec\theta = \frac{r}{x} \qquad \cot\theta = \frac{x}{y}$$

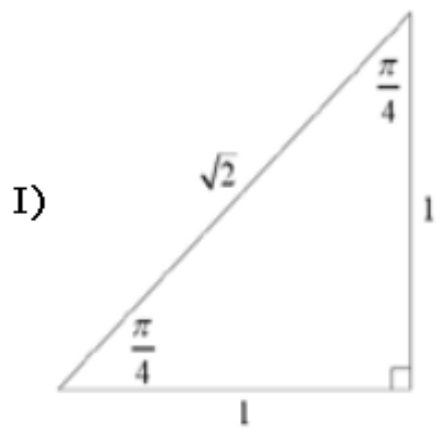


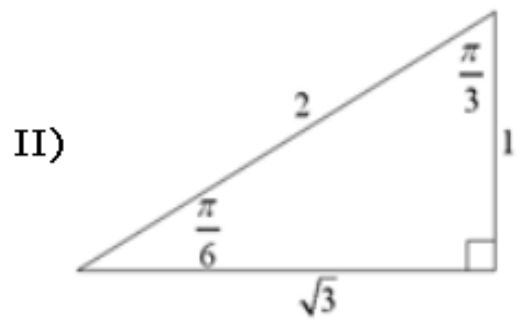
Trigonometry



Trigonometry

Special Triangles





Using the "special" triangles above, we can find the exact trigonometric ratios for angles of pi/3, pi/4 and pi/6. These triangles can be constructed quite easily and provide a simple way of remembering the trigonometric ratios. The table below lists some of the more common angles (in both radians and degrees) and their exact trigonometric ratios.

θ	rad	$\sin\theta$	$\cos\theta$	$\tan\theta$
$0°$	0	0	1	0
$30°$	$\pi/6$	$1/2$	$\sqrt{3}/2$	$\sqrt{3}/3$
$45°$	$\pi/4$	$\sqrt{2}/2$	$\sqrt{2}/2$	1
$60°$	$\pi/3$	$\sqrt{3}/2$	$1/2$	$\sqrt{3}$
$90°$	$\pi/2$	1	0	–

Trigonometry

metry Tutorial

Trigonometric Identities

The identities listed below are the basic trigonometric identities. They can be combined with one another to create many more identities.

Reciprocal Identities

$$\sin\theta = \frac{1}{\csc\theta} \qquad \csc\theta = \frac{1}{\sin\theta}$$

$$\cos\theta = \frac{1}{\sec\theta} \qquad \sec\theta = \frac{1}{\cos\theta}$$

$$\tan\theta = \frac{1}{\cot\theta} \qquad \cot\theta = \frac{1}{\tan\theta}$$

Quotient Identities

$$\tan\theta = \frac{\sin\theta}{\cos\theta}$$

$$\cot\theta = \frac{\cos\theta}{\sin\theta}$$

Pythagorean Identities

$$\underline{\sin^2\theta + \cos^2\theta = 1}$$

$$\sec^2\theta = 1 + \tan^2\theta$$

$$\csc^2\theta = 1 + \cot^2\theta$$

Trigonometric Formulas

The trigonometric formulas below can be combined with the identities above to create very complex trigonometric identities. These formulas are often necessary when proving trigonometric identities.

Addition/Subtraction Formulas

$$\underline{\sin(x+y) = \sin x\cos y + \cos x\sin y}$$

$$\sin(x-y) = \sin x\cos y - \cos x\sin y$$

$$\underline{\cos(x+y) = \cos x\cos y - \sin x\sin y}$$

$$\cos(x-y) = \cos x\cos y + \sin x\sin y$$

$$\tan(x+y) = \frac{\tan x + \tan y}{1 - \tan x\tan y}$$

$$\tan(x-y) = \frac{\tan x - \tan y}{1 + \tan x\tan y}$$

Double Angle Formulas

$$\sin 2x = 2\sin x\cos x$$

$$\begin{aligned}\cos 2x &= \cos^2 x - \sin^2 x \\ &= 2\cos^2 x - 1 \\ &= 1 - 2\sin^2 x\end{aligned}$$

$$\tan 2x = \frac{2\tan x}{1 - \tan^2 x}$$

Point Transformations – 2D
(around origin!)

- Representation of **Points** 2 X 1 matrix: $\begin{bmatrix} x \\ y \end{bmatrix}$

General Problem: **|B| = |T| |A|**

- **|T|** represents a generic **operator** to be applied to **point** **|A|**
- **|T|** is a **geometric transformation matrix**
- **|B|** **is the transformed point**.

Algebraic Form ????

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{aligned} x' &= ax + cy \\ y' &= bx + dy \end{aligned}$$

Point Transformations – 2D
Special Cases

|T| - I - identity matrix

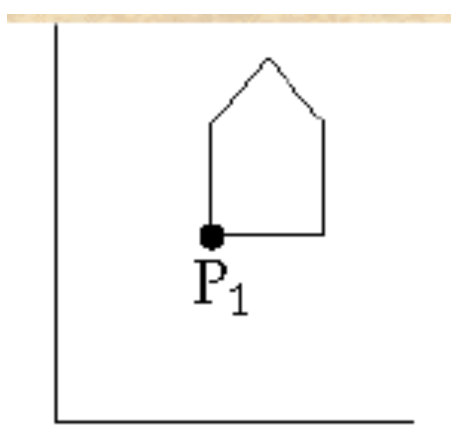
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$a = d = 1,\ b = c = 0$

$$x' = x$$
$$y' = y$$

$$x' = ax \quad 0$$
$$y' = 0 \quad dy$$

$$P_1 = \begin{matrix} x \\ y \end{matrix}$$



House at P_1

Point Transformations – 2D
Special Cases

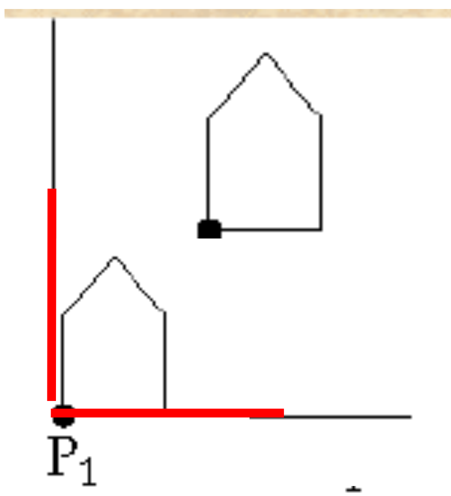
|T| - 0 - zero matrix

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

$a = d = b = c = 0$

$$x' = 0$$
$$y' = 0$$

$$x' = 0$$
$$y' = 0$$



Point Transformations – 2D
Scaling & Reflection
(around origin!)

$$|T| = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \qquad \begin{array}{c} a \\ \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \\ d \end{array} \qquad b = c = 0$$

$$x' = ax \qquad\qquad x' = ax \; 0$$

$$y' = dy \qquad\qquad y' = 0 \; dy$$

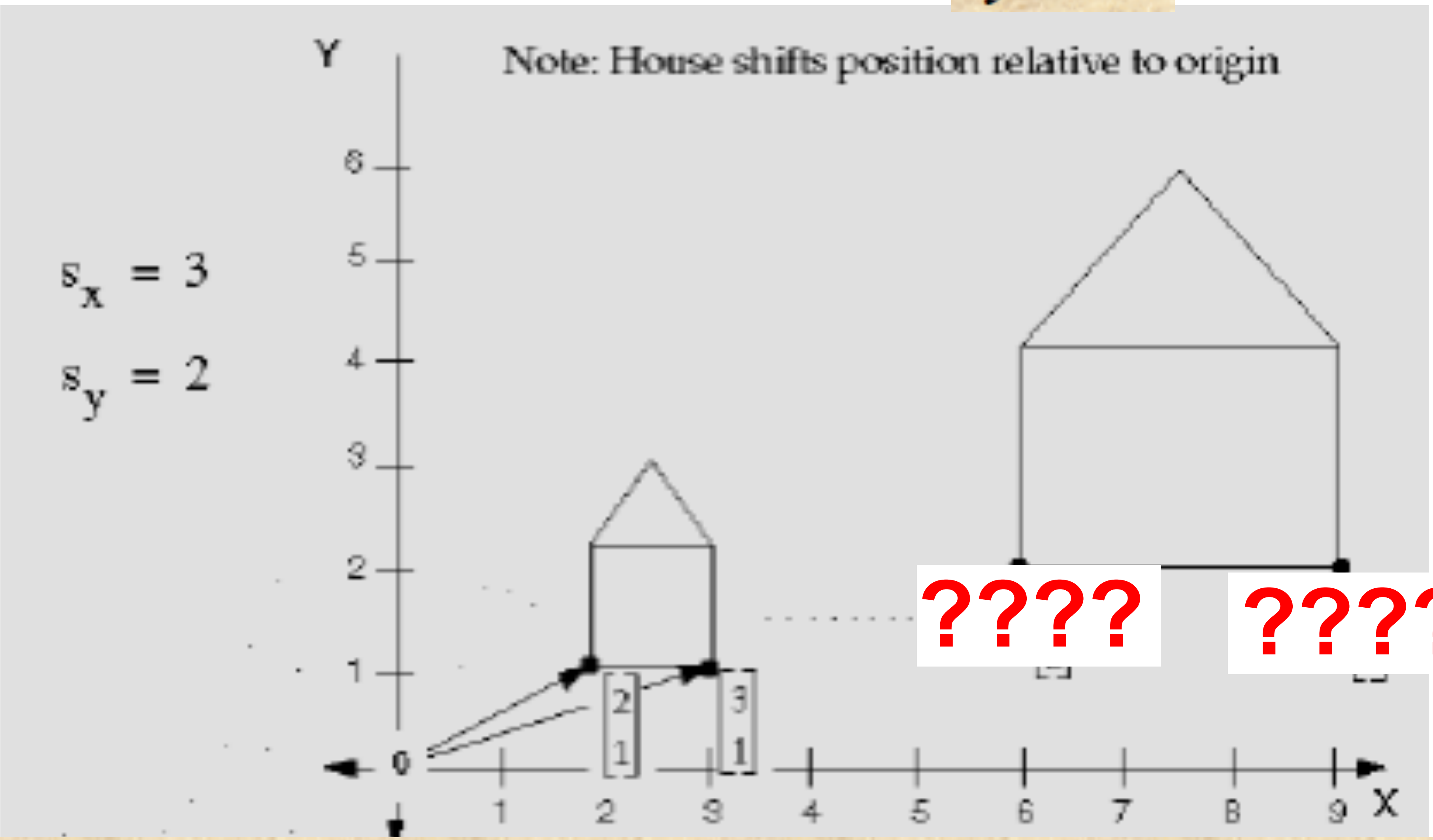
IF $a = d > 1$ **enlargement**

IF $0 < a = d < 1$ **compression**

IF $a = d$ **uniform scaling**

Point Transformations – 2D
Scaling
(around origin!)

$$x' = s_x \, x$$
$$y' = s_y \, y$$



Only diagonal terms involved in scaling and reflections

Point Transformations – 2D
Scaling & Reflection
(around origin!)

$$|T| = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \quad \begin{array}{c} a \\ \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \\ d \end{array} \qquad b = c = 0$$

$$x' = ax$$
$$y' = dy$$

$$x' = ax \; 0$$
$$y' = 0 \; dy$$

IF $a = d < 0$ **reflection through an axis or plane**

Only diagonal terms involved in scaling and reflections

Point Transformations – 2D
Scaling & Reflection
(around origin!)

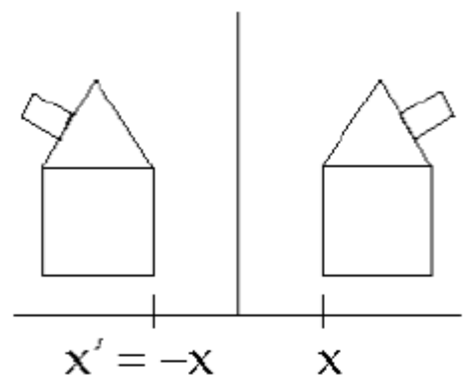
$$|T| = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \quad \begin{array}{c} a \\ \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \\ d \end{array}$$

$b = c = 0$

$$x' = -x$$
$$y' = y$$

$$x' = ax \quad 0$$
$$y' = 0 \quad dy$$

IF a = -1 d =1 **reflection about Y axis**



Point Transformations – 2D
Scaling & Reflection
(around origin!)

$$|T| = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \quad \begin{array}{c} a \\ \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \\ d \end{array}$$

$b = c = 0$

$$x' = x$$
$$y' = -y$$

$$x' = ax \; 0$$
$$y' = 0 \; dy$$

IF a = 1 d = -1 **reflection about x axis**

Point Transformations – 2D
Scaling & Reflection
(around origin!)

$$|T| = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \qquad \begin{array}{c} a \\ \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \\ d \end{array}$$

$b = c = 0$

$$x' = x$$
$$y' = y$$

$$x' = ax \;\; 0$$
$$y' = 0 \;\; dy$$

IF $a = 1$ $d = 1$

reflection about xy = 45 degree axis

Point Transformations – 2D
Scaling & Reflection
(around origin!)

$$|T| = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \qquad \begin{array}{c} a \\ \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \\ d \end{array} \qquad b = c = 0$$

$$x' = -x \qquad x' = ax + 0$$

$$y' = -y \qquad y' = 0 + dy$$

IF $a = -1$ $d = -1$

reflection about xy = - 45 degree axis

Point Transformations – 2D
Shearing
(around origin!)

$$|T| = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

$c = 0$

$$x' = x$$
$$y' = bx + y$$

$$x' = ax \quad 0$$
$$y' = bx + dy$$

IF $a = 1$ $d = 1$ $b = 2$

Off diagonal terms

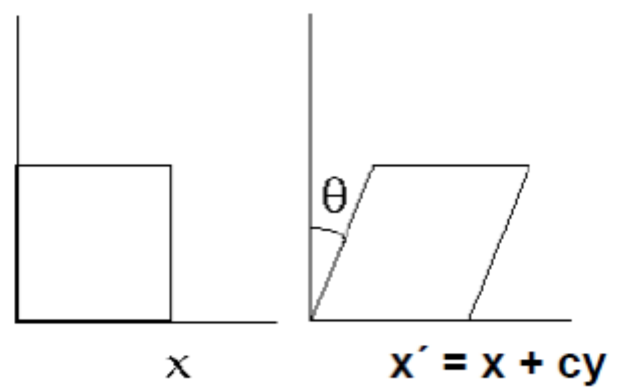
Point Transformations – 2D
Shearing
(around origin!)

$$|T| = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

$$x' = ax + cy$$
$$y' = 0 \quad dy$$

$$x' = x + cy$$
$$y' = y$$

IF a = 1 d = 1 b = 0 c =2



Off diagonal terms

Point Transformations – 2D
Rotation
(around origin!)

A.3.4 Compound Angle

For two angles α and β, the sines of the sum and difference of the angles are, respectively,

$$\boxed{\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta} \quad (A.19)$$

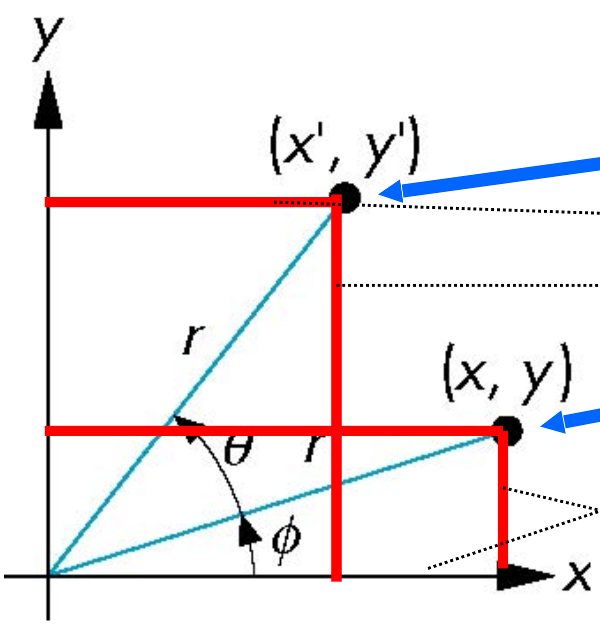
$$\sin(\alpha - \beta) = \sin\alpha\cos\beta - \cos\alpha\sin\beta \quad (A.20)$$

Similarly, the cosines of the sum and difference of the angles are

$$\boxed{\cos(\alpha + \beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta} \quad (A.21)$$

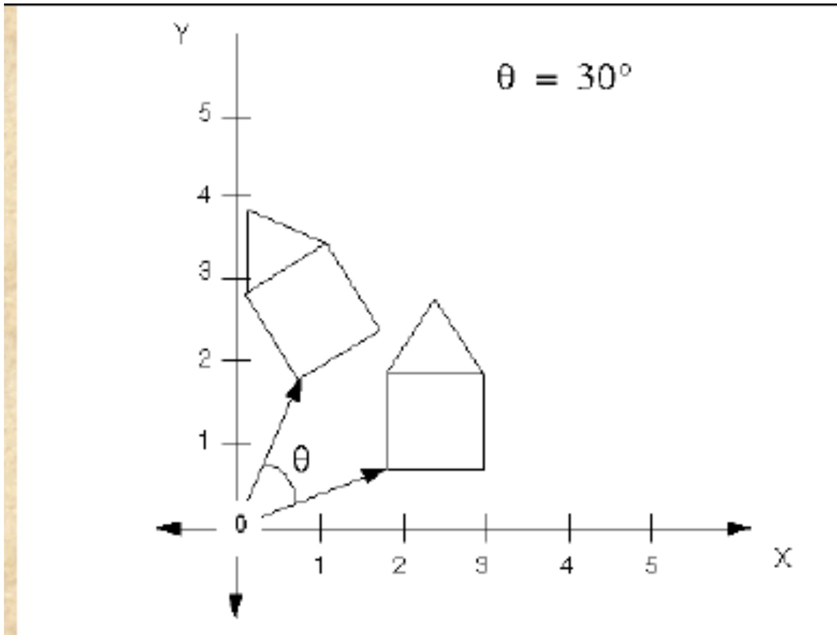
$$\cos(\alpha - \beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta \quad (A.22)$$

Positive Rotations: counter clockwise about the origin



$$x' = r\,\boxed{\cos(\phi + \theta)}$$
$$y' = r\,\boxed{\sin(\phi + \theta)}$$

$$\boxed{x = r\cos\phi}$$
$$\boxed{y = r\sin\phi}$$



Point Transformations – 2D
Rotation
(around origin!)

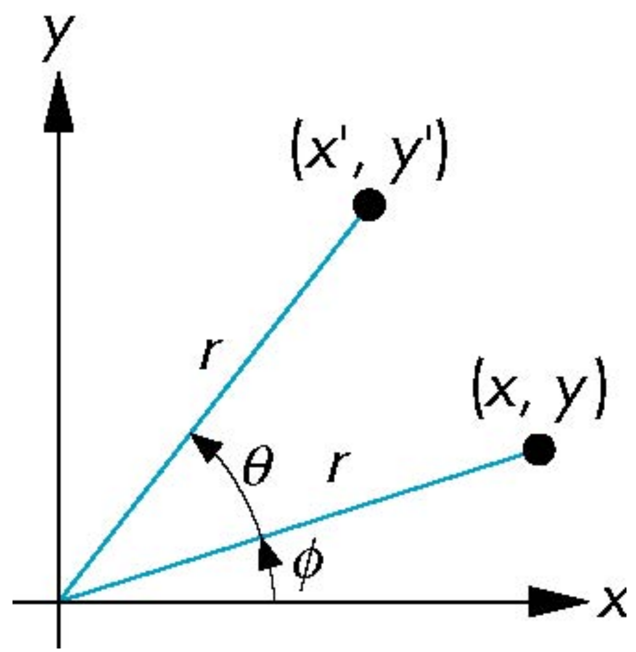
r = 1

$$x = r\cos\phi$$
$$y = r\sin\phi$$

$$x' = \cos(\phi + \theta)$$
$$y' = \sin(\phi + \theta)$$

$$\boxed{\begin{aligned}\cos(\phi+\theta) &= \overset{X}{\cos\phi}\cos\theta - \overset{Y}{\sin\phi}\sin\theta \\ \sin(\phi+\theta) &= \underset{Y}{\sin\phi}\cos\theta + \underset{X}{\cos\phi}\sin\theta\end{aligned}}$$

$$\boxed{\begin{aligned}x' &= X\cos\theta - Y\sin\theta \\ y' &= X\sin\theta + Y\cos\theta\end{aligned}}$$



Point Transformations – 2D
Rotation
(around origin!)

$$\boxed{\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}}$$

Rotation matrix is orthogonal!

$$|T| = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

For rotations, det|T| = 1 and $|T|^T = |T|^{-1}$

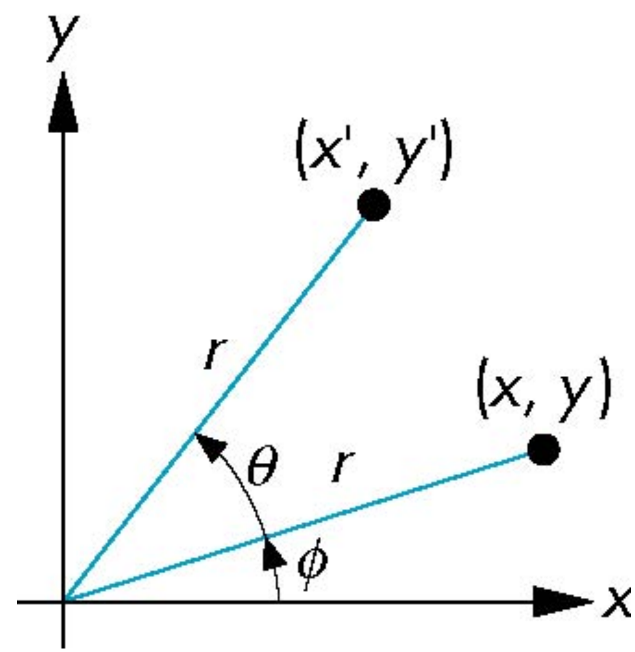
Again, from the Pythagorean theorem we know that

$$a^2 + b^2 = c^2$$

where c is the length of the hypotenuse and a and b are the lengths of the other two sides. In the case where the length of the hypotenuse is 1, the length of the other two sides are $\cos\theta$ and $\sin\theta$, so

$$\boxed{\sin^2\theta + \cos^2\theta = 1} \tag{A.5}$$

where $\sin^2\theta = (\sin\theta)(\sin\theta)$, and similarly for $\cos^2\theta$.



Point Transformations – 2D
Rotation
(around origin!)

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

Negative Rotation -θ!

For rotations, det|T| = 1 and $|T|^T = |T|^{-1}$

$$|T| = \begin{bmatrix} \boxed{\cos(\theta) \quad -\sin(\theta)} \\ \boxed{\sin(\theta) \quad \cos(\theta)} \end{bmatrix}$$

Two of the trigonometric functions, cosine and secant, are symmetric across $\theta = 0$ and are called *even* functions:

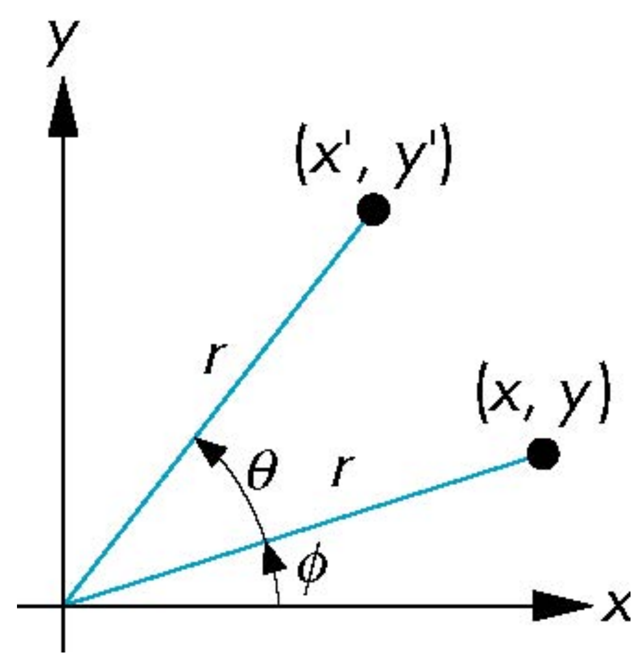
$$\boxed{\cos(-\theta) = \cos\theta} \tag{A.13}$$

$$\sec(-\theta) = \sec\theta \tag{A.14}$$

The remainder are antisymmetric across $\theta = 0$ and are called *odd* functions:

$$\boxed{\sin(-\theta) = -\sin\theta} \tag{A.15}$$

$$\csc(-\theta) = -\csc\theta \tag{A.16}$$



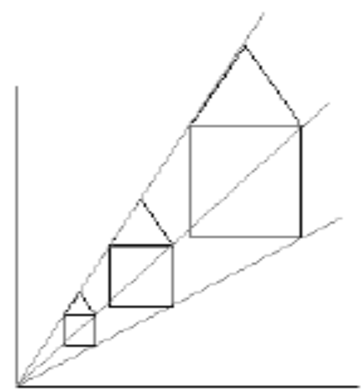
Point Transformations – 2D
Translations (around origin!)

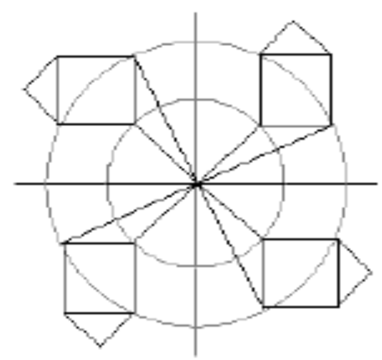
$B = A + T_r$, where $T_r = |tx\ ty|^T$

We cannot directly represent translations as matrix multiplication, as we can rotations and scaling!

Where else are translations introduced?

1) Rotations - when object not centered at the origin.
2) Scaling - when objects / lines not centered at the origin.
   - line from (2,1) to (4,1) scaled by 2 in x & y.
   - If line intersects the origin, no translation.
   - Scaling is about the origin.





Can we represent translations in our general transformation matrix? YES!

Point Transformations – 2D

Translations (around origin!)

$B = A + T_r$, where $T_r = |tx\ ty|^T$

Can we represent translations in our general transformation matrix?

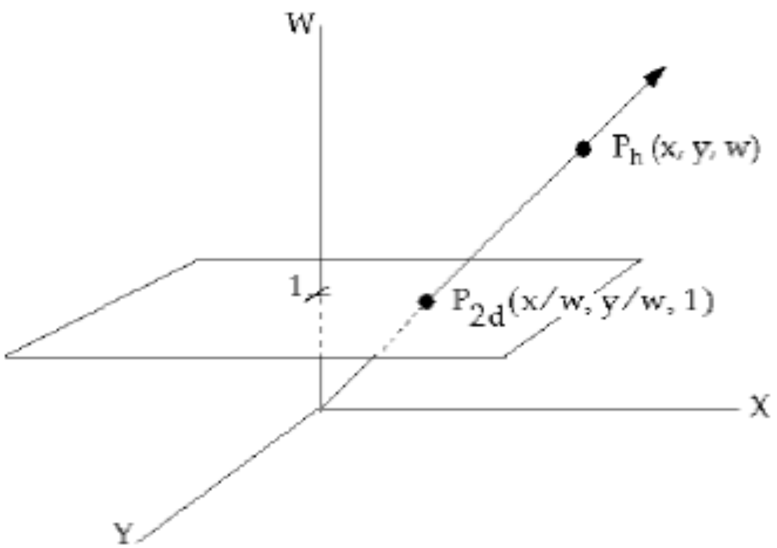
Yes, by using homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ Z' \end{bmatrix} = \begin{bmatrix} a & c & t_x \\ b & d & t_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$x' = ax + cy + tx$

$y' = bx + dy + ty$

Each point is now represented by a triple: (x, y, W)



x/W, y/W are called the Cartesian coordinates of the homogeneous points.

Cartesian coordinates are just the plane w=1 in this space.

Two homogeneous coordinates (x1, y1, w1) & (x2, y2, w2) may represent the same point, iff they are multiples of one another: (1,2,3) & (3,6,9). There is no unique homogeneous representation of a point.

Composite Transformations – 2D
Translations, Rotations, Scaling (around origin!)

If we want to apply a series of transformations T1, T2, T3 to a set of points, We can do it 2 ways:

1) We can calculate p'=T1*p, p'' = T2*p', p'''=T3*p''

2) Calculate T= T1*T2*T3, then p'''= T*p.

Method 2, saves large number of adds and multiplies. Approximately 1/3 as many operations. Therefore, we concatenate or compose the matrices into one final transformation matrix that we apply to the points.

Translations

Translate the points by tx1, ty1, then by tx2, ty2:

$$\begin{bmatrix} 1 & 0 & (tx1 + tx2) \\ 0 & 1 & (tx1 + tx2) \\ 0 & 0 & 1 \end{bmatrix}$$

Composite Transformations – 2D
Translations, Rotations, Scaling (around origin!)

Scaling Similar to translations

Rotations Rotate by q1, then by q2, stick the (q1+q2) in for q,
or calculate T1 for q1, then T2 for q2 & multiply them.

Composite Transformations – 2D
Rotation about an arbitrary point P in space
(NOT around origin!)

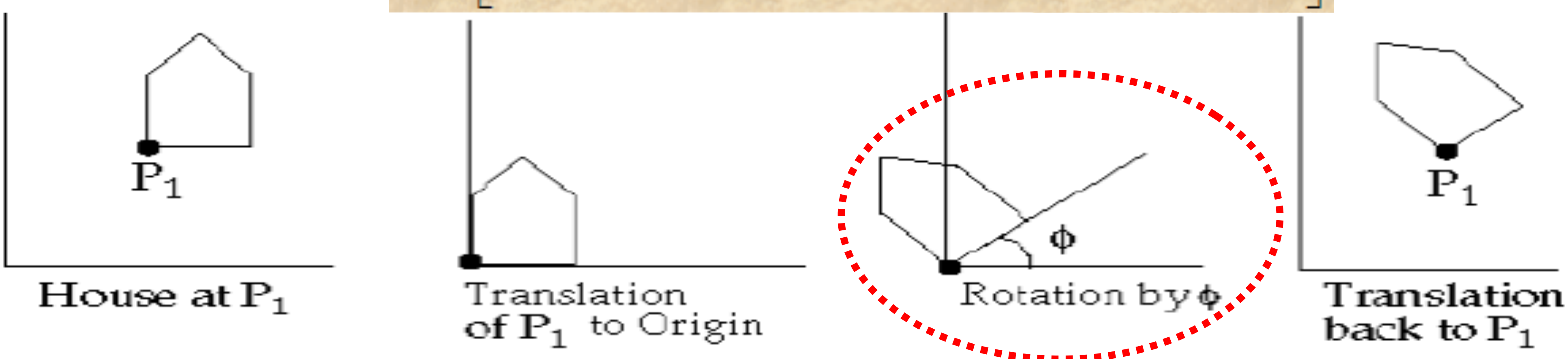
As we mentioned before, rotations are about the origin.
So to rotate about a point P in space, translate so that P coincides with the origin, then rotate, then translate back:

Translate by (-Px, -Py)
Rotate
Translate by (Px, Py)

T = T1(Px,Py) * T2(q) * T3(-Px, -Py)

$$= \begin{bmatrix} 1 & 0 & Px \\ 0 & 1 & Py \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -Px \\ 0 & 1 & -Py \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & Px*(1-\cos(\theta))+Py*(\sin(\theta) \\ \sin(\theta) & \cos(\theta) & Py*(1-\cos(\theta))-Px*\sin(\theta) \\ 0 & 0 & 1 \end{bmatrix}$$



Composite Transformations – 2D
Scaling about an arbitrary point P in space
(NOT around origin!)

Translate P to the origin

Scale

Translate P back

T = T1(Px,Py)* T2(sx, sy)*T3(-Px, -Py)

$$T = \begin{bmatrix} Sx & 0 & \{Px*(1-Sx)\} \\ 0 & Sy & \{Py*(1-Sy)\} \\ 0 & 0 & 1 \end{bmatrix}$$

Composite Transformations – 2D
Commutivity of Transformations

If we scale, then translate to the origin, then translate back, is that equivalent to translate to origin, scale, translate back?

When is the order of matrix multiplication unimportant?

When does T1*T2 = T2*T1?

Cases where T1*T2 = T2*T1:

T1	T2
translation	translation
scale	scale
rotation	rotation
scale(uniform)	rotation

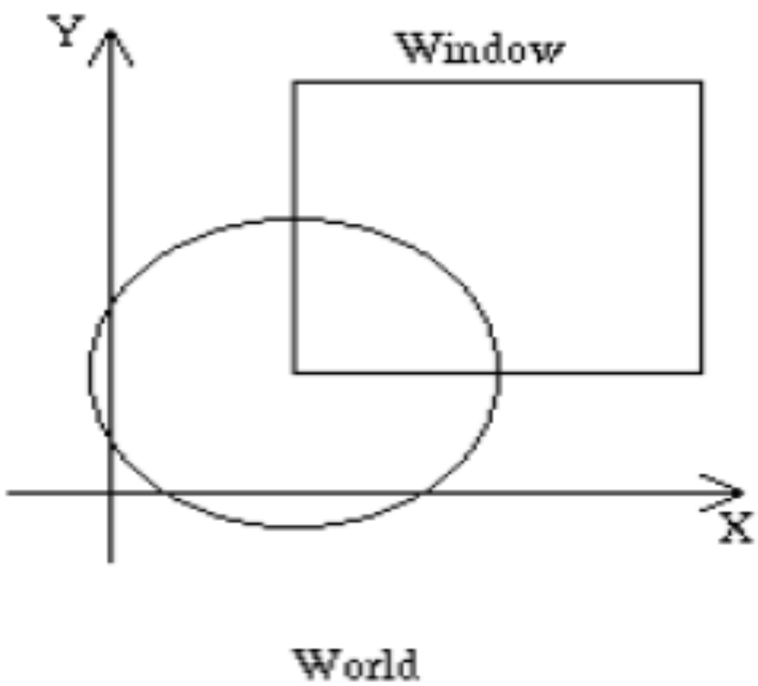
Coordinate Systems

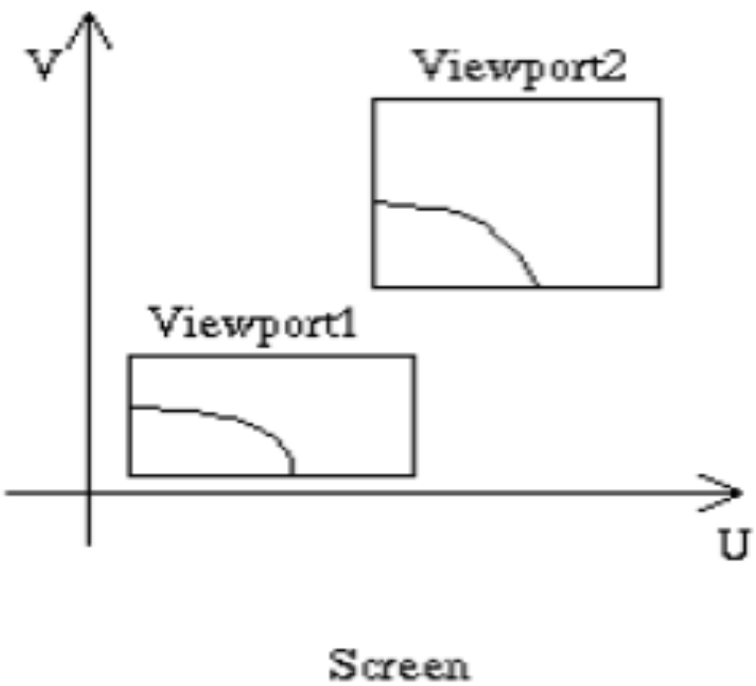
Screen Coordinates: The coordinate system used to address the screen (device coordinates)

World Coordinates: A user-defined application specific coordinate system having its own units of measure, axis, origin, etc.

Window: The rectangular region of the world that is visible.

Viewport: The rectangular region of the screen space that is used to display the window.





Window to Viewport Transformation

Want to find the transformation matrix that maps the window in world coordinates to the viewport in screen coordinates.

Viewport: (u, v space) denoted by: $u_{min}, v_{min}, u_{max}, v_{max}$

Window: (x, y space) denoted by: $x_{min}, y_{min}, x_{max}, y_{max}$

The transformation:

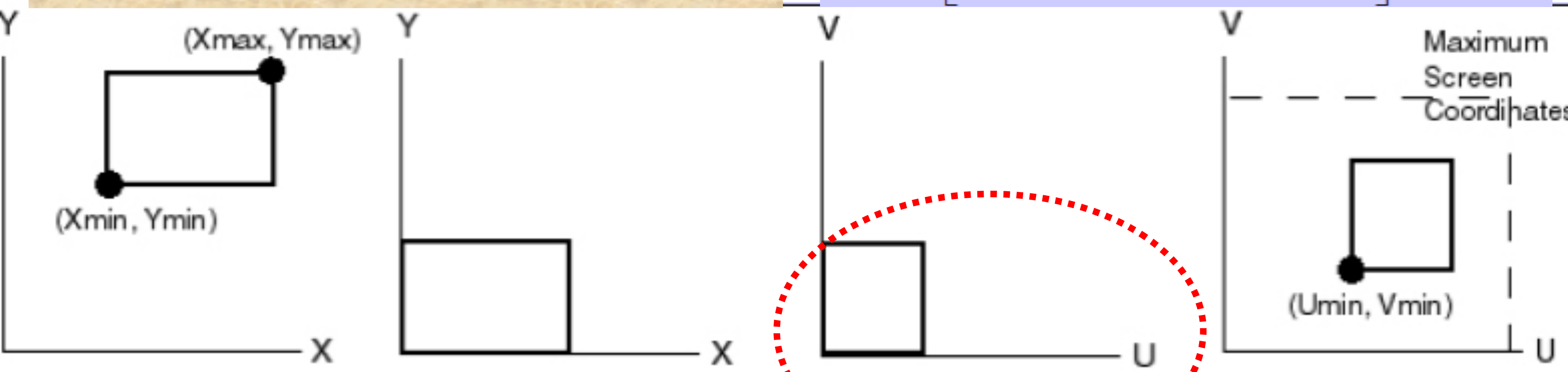
- Translate the window to the origin
- Scale it to the size of the viewport
- Translate it to the viewport location

$$M_{WV} = T(U_{\min}, V_{\min}) * S(S_x, S_y) * T(-x_{\min}, -y_{\min});$$

$$S_x = (U_{\max} - U_{\min}) / (x_{\max} - x_{\min});$$

$$S_y = (V_{\max} - V_{\min}) / (y_{\max} - y_{\min});$$

$$M_{WV} = \begin{bmatrix} S_x & 0 & (-x_{\min} * S_x + U_{\min}) \\ 0 & S_y & (-y_{\min} * S_y + V_{\min}) \\ 0 & 0 & 1 \end{bmatrix}$$



Window in World Coordinates

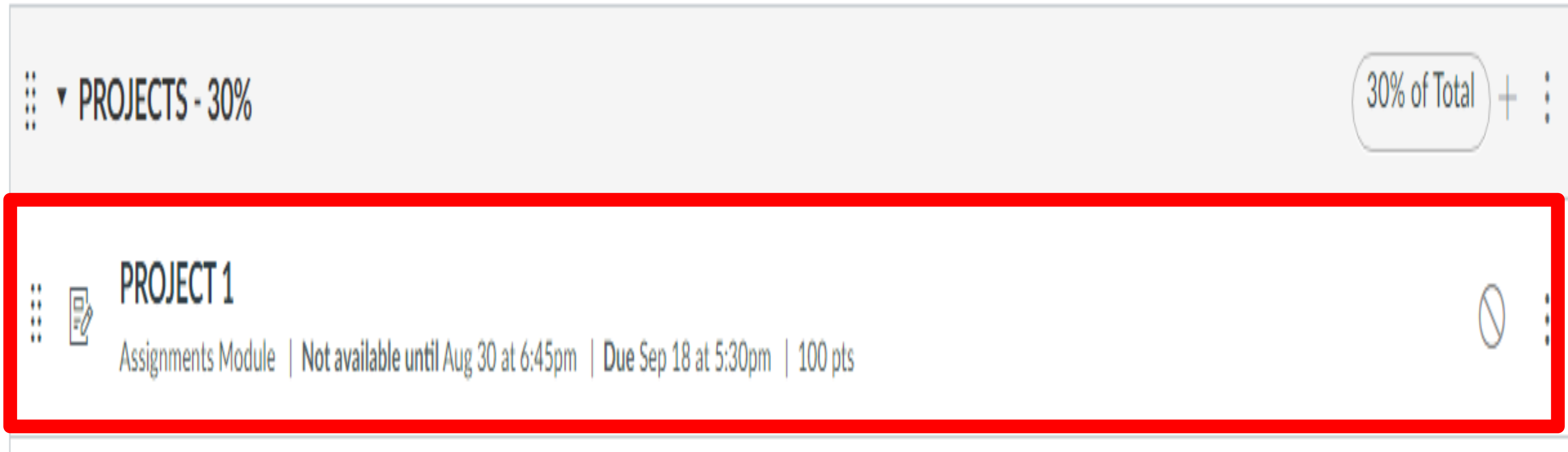
Window translated to origin

Window Scaled to size of Viewport

Viewport Translated to final position

THE END

At 6:25 PM



VH Publish.

PROJECT 1

Publish | Edit

The PROJECT is due by 5:30 PM of the due date class.

PROJECT 1.doc

Additional files for PROJECT 1:

S'cool Bus FALL 2023.pdf

PROJECT 1 Acceptance Testing.DOC

PROJECT 1 Acceptance Testing Check Sheet.DOC

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

PROJECT 1 Presentation.pptx

Points 100

Submitting a file upload

Due	For	Available from	Until
Sep 18 at 5:30pm	Everyone	Aug 30 at 6:45pm	Sep 18 at 5:30pm

PROJECT 1

Publish | Edit

The PROJECT is due by 5:30 PM of the due date class.

PROJECT 1.doc

Additional files for PROJECT 1:

S'cool Bus FALL 2023.pdf

PROJECT 1 Acceptance Testing.DOC

PROJECT 1 Acceptance Testing Check Sheet.DOC

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

PROJECT 1 Presentation.pptx

Points 100

Submitting a file upload

Due	For	Available from	Until
Sep 18 at 5:30pm	Everyone	Aug 30 at 6:45pm	Sep 18 at 5:30pm

Name: ______________________________

PROJECT 1
(100 points)

Hours spent:

The PROJECT is to be turned in before 5:30 PM of the due date class.

Submit:

1. ***The "PROJECT 1 Blue Print" labels all rectangles and specify their vertices locations (map them as comments in our source code).***
2. ***The zipped Visual Studio2019 or WEBGL "PROJECT 1".***
3. ***The "PROJECT 1 Acceptance Testing".***
4. ***The "PROJECT 1 Acceptance Testing Check Sheet" as "score.OPENGL.doc or score.WEBGL.doc" to CANVAS.***
5. ***The ppt "PROJECT 1 PRESENTATION (no more than 10 slides)".***

S'COOL BUS – FALL 2023 Modeling Instructions

1. The **glutCreateWindow** OpenGL call will take as parameter "**YourLastName** FirstName **BUS Version 1**".

2. The file "**S'COOL BUS FALL 2023**" contains the specifications for the BUS that you will start modeling in this first PROJECT.

3. In the "`void myReshape(int w, int h)`" function, you need to adjust the viewing box to allow "`glOrtho`" to contain the parts of your BUS.

Add a screenshot of the "**YourLastName** FirstName **BUS Version 1**" window.

4. Adjust the camera so you can get the **Front** image of **BUS** (**gluLookAt** added to display function).

Add a screenshot of the "**YourLastName** FirstName **BUS Version 1**" window.

PROJECT 1

Publish | Edit

The PROJECT is due by 5:30 PM of the due date class.

PROJECT 1.doc

Additional files for PROJECT 1:

S'cool Bus FALL 2023.pdf

PROJECT 1 Acceptance Testing.DOC

PROJECT 1 Acceptance Testing Check Sheet.DOC

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

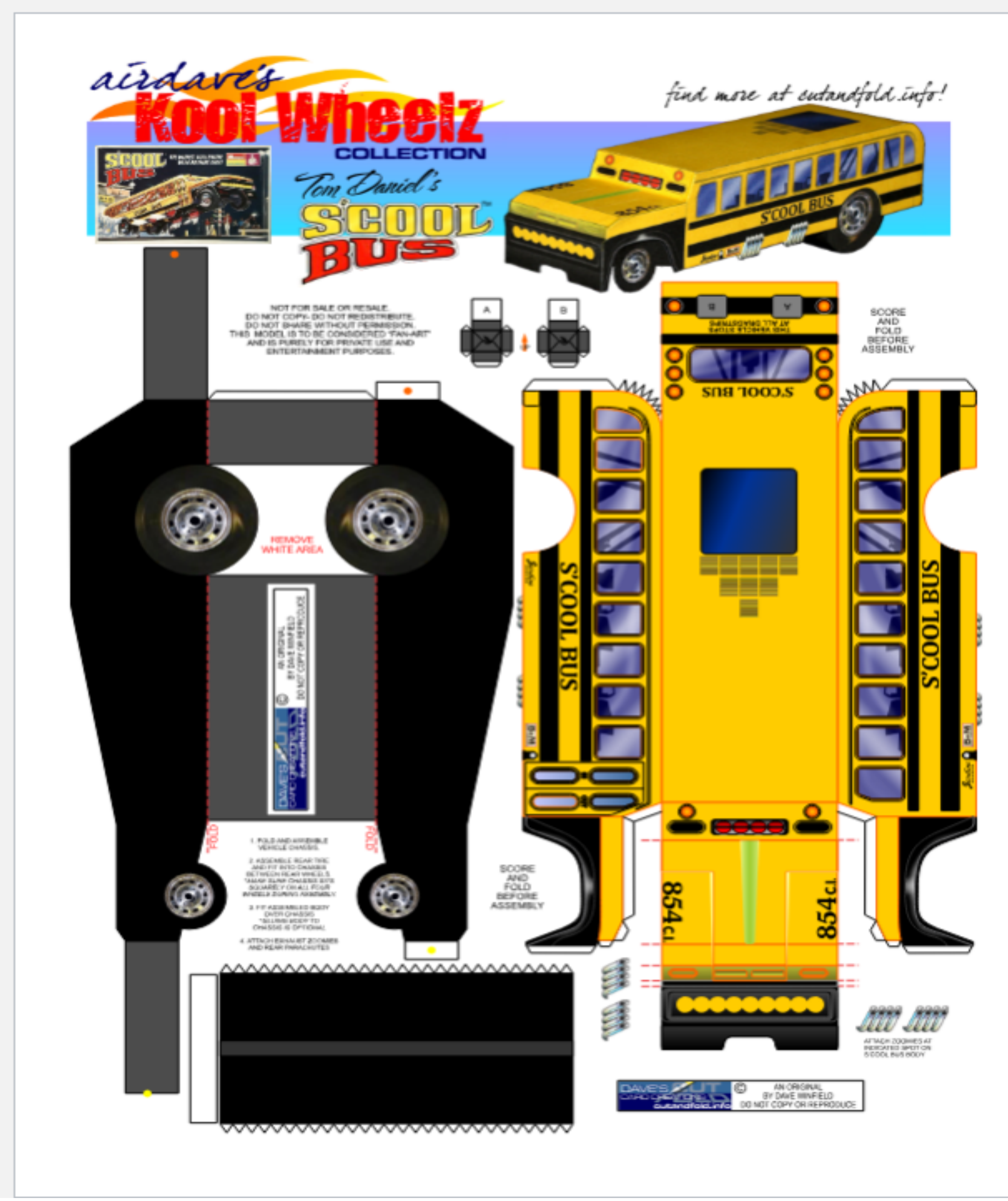
PROJECT 1 Presentation.pptx

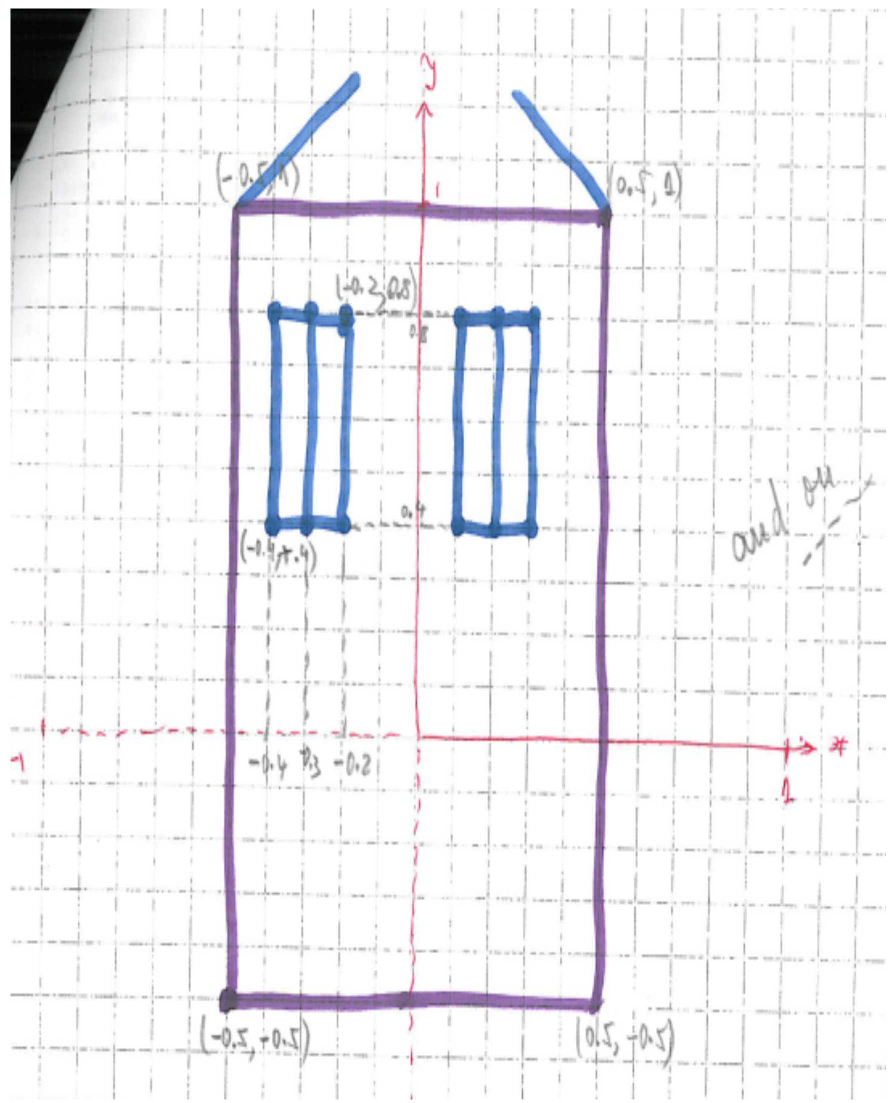
Points 100

Submitting a file upload

Due	For	Available from	Until
Sep 18 at 5:30pm	Everyone	Aug 30 at 6:45pm	Sep 18 at 5:30pm

1. *The "PROJECT 1 Blue Print" labels all rectangles and specify their vertices locations (map them as comments in our source code).*





Name: ______________________________

PROJECT 1
(100 points)

Hours spent:

The PROJECT is to be turned in before 5:30 PM of the due date class.

Submit:

1. ***The "PROJECT 1 Blue Print" labels all rectangles and specify their vertices locations (map them as comments in our source code).***
2. ***The zipped Visual Studio2019 or WEBGL "PROJECT 1".***
3. ***The "PROJECT 1 Acceptance Testing".***
4. ***The "PROJECT 1 Acceptance Testing Check Sheet" as "*****score.OPENGL.doc or score.WEBGL.doc*****" to CANVAS.***
5. ***The ppt "PROJECT 1 PRESENTATION (no more than 10 slides)".***

S'COOL BUS – FALL 2023 Modeling Instructions

1. The **glutCreateWindow** OpenGL call will take as parameter "**YourLastName** FirstName **BUS Version 1**".

2. The file "**S'COOL BUS FALL 2023**" contains the specifications for the BUS that you will start modeling in this first PROJECT.

3. In the "`void myReshape(int w, int h)`" function, you need to adjust the viewing box to allow "`glOrtho`" to contain the parts of your BUS.

Add a screenshot of the "**YourLastName** FirstName **BUS Version 1**" window.

4. Adjust the camera so you can get the **Front** image of **BUS** (**gluLookAt** added to display function).

Add a screenshot of the "**YourLastName** FirstName **BUS Version 1**" window.

PROJECT 1

Publish | Edit

The PROJECT is due by 5:30 PM of the due date class.

PROJECT 1.doc

Additional files for PROJECT 1:

S'cool Bus FALL 2023.pdf

PROJECT 1 Acceptance Testing.DOC

PROJECT 1 Acceptance Testing Check Sheet.DOC

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

PROJECT 1 Presentation.pptx

Points 100

Submitting a file upload

Due	For	Available from	Until
Sep 18 at 5:30pm	Everyone	Aug 30 at 6:45pm	Sep 18 at 5:30pm

The University of New Mexico

Name: ____________________

PROJECT 1 Acceptance Testing:

Insert screenshots of the output produced and insert it in this file ……..

1. The glutCreateWindow OpenGL call will take as parameter "YourLastName FirstName BUS Version 1".
SCREENSHOT
……..

2. The file "S'COOL BUS FALL 2023" contains the specifications for the BUS that you will start modeling in this first PROJECT.
SCREENSHOT
……..

3. In the "`void myReshape(int w, int h)`" function, you need to adjust the viewing box to allow "`glOrtho`" to contain the parts of your BUS.
SCREENSHOT
……..

4. Adjust the camera so you can get the **Front** image of **BUS** (**gluLookAt** added to display function).
SCREENSHOT Output of your program is:
……..
Copy and paste your gluLookAt here
……..

5. Adjust the camera so you can get the **Back** image of **BUS** (**gluLookAt** added to display function).
SCREENSHOT Output of your program is:
……..
Copy and paste your gluLookAt here
……..

6. Adjust the camera so you can get the Side image of **BUS** (**gluLookAt** added to display function).
SCREENSHOT Output of your program is:
……..
Copy and paste your gluLookAt here
……..

7. Adjust the camera so you can get the Isometric image of **BUS** (**gluLookAt** added to display function). This means that camera needs to be placed on the line from (0,0,0) to (1,1,1), aiming at the BUS origin (0,0,0) and the up vector is on the y axis (0,**1**,0).
SCREENSHOT Output of your program is:
……..
Copy and paste your gluLookAt here

PROJECT 1

Publish | Edit

The PROJECT is due by 5:30 PM of the due date class.

PROJECT 1.doc

Additional files for PROJECT 1:

S'cool Bus FALL 2023.pdf

PROJECT 1 Acceptance Testing.DOC

PROJECT 1 Acceptance Testing Check Sheet.DOC

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

PROJECT 1 Presentation.pptx

Points 100

Submitting a file upload

Due	For	Available from	Until
Sep 18 at 5:30pm	Everyone	Aug 30 at 6:45pm	Sep 18 at 5:30pm

The University

Name:

PROJECT 1

Hours:

Acceptance Testing Check Sheet:

CHECKLIST (YOU MUST SCORE YOURSELF!):

1. Submitted the Blue Print? 6 points
2. Submitted PROJECT1.zip? 6 points
3. Submitted Acceptance Testing.docx? 6 points
4. Submitted Acceptance Testing Check Sheet as score.OPENGL.docx or score.WEBGL.docx? 6 points
5. Submitted PROJECT 1 PRESENTATION.pptx? 6 points

6. ACCEPTANCE TESTING Step 1? 10 points
7. ACCEPTANCE TESTING Step 2? 10 points
8. ACCEPTANCE TESTING Step 3? 10 points
9. ACCEPTANCE TESTING Step 4? 10 points
10. ACCEPTANCE TESTING Step 5? 10 points
11. ACCEPTANCE TESTING Step 6? 10 points
12. ACCEPTANCE TESTING Step 7? 10 points

Score

I CERTIFY THAT THIS IS MY OWN WORK and THE CHECKBOXES reflect the completion of these DELIVERABLES.

PROJECT 1

Publish Edit

The PROJECT is due by 5:30 PM of the due date class.

PROJECT 1.doc

Additional files for PROJECT 1:

S'cool Bus FALL 2023.pdf

PROJECT 1 Acceptance Testing.DOC

PROJECT 1 Acceptance Testing Check Sheet.DOC

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

Rename it to score.WEBGL.docx. (MUST BE A .DOCX DOCUMENT)

PROJECT 1 Presentation.pptx

Points 100

Submitting a file upload

Due	For	Available from	Until
Sep 18 at 5:30pm	Everyone	Aug 30 at 6:45pm	Sep 18 at 5:30pm

PROJECT 1 Presentation

Name:

COSC 4370 - FALL 2023

OpenGL Visual Studio 2019 Project

WebGL Project

No more than 10 slides

09.18.2023 (M 5:30 to 7) (8)		PROJECT 1

From 6:30 to 6:45 PM

The Un

Class PARTICIPATION on Math Review 1

Not available until Aug 30 at 6:30pm | Due Aug 30 at 6:45pm | 100 pts

VH, publish

From 6:30 to 6:45 PM

The Un

Class PARTICIPATION on Math Review 1

Publish Edit

Related I

Download and complete this word document.

Class Participation on Math Review 1.docx

You will be prompted when to Upload completed document to CANVAS as **score.doc** (example 100.doc).

Warning:

TA, at random, will inspect the Uploaded document.

If you score is not honestly entered you will get a zero.

Points 100

Submitting a file upload

Due	For	Available from	Until
Aug 30 at 6:45pm	Everyone	Aug 30 at 6:30pm	Aug 30 at 6:45pm

The Un

Name: ____________________________

Total score:

Class PARTICIPATION on Math Review 1.doc ANSWER SHEET

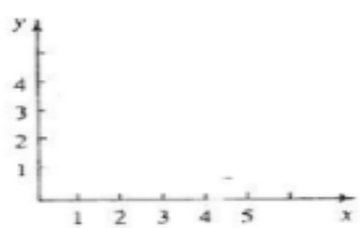
(Out of 100 points. Please record your own total score!)
(Attach as score.doc!)

1. (Coordinate System) In a right-handed 3D coordinate system the z-axis comes out of the screen. Circle one. (20 points)

T
F

2. (Vectors) In a right-handed 3D coordinate system the z-axis comes out of the screen. (20 points)

Draw point P = (1, 3) and Q = (4, 1).
The displacement from P to Q is a vector **v**. Draw **v**.
What are the components of **v** = ?



Self Graded - correctly

3. (Operations with vectors) (20 points)

Given two vectors **a** = (2, 5, 6) and **b** = (-2, 7, 1)

What is **a** + **b**?
What is 6**a**?

The Un

You will be prompted when to Upload completed document to CANVAS as **score.doc** (example 100.doc).

Warning:

TA, at random, will inspect the Uploaded document.

If you score is not honestly entered you will get a zero.

Please rename document to score.doc (example 100.doc)
Warning: if you score is not honestly honest you will get a zero.

Name: ______________________________

Total score: [ ]
Class PARTICIPATION on Math Review 1.doc **ANSWER SHEET**
(Out of 100 points. Please record your own total score!)
(Attach as score.doc!)

VH, it closes at 6:45 PM

NEXT.

The University of New Mexico

09.06.2023 (W 5:30 to 7) (5)	Homework 2	Lecture 3

HOMEWORK - 15%

15% of Total

Homework 2

HOMEWORKS 15% Module | Not available until Aug 28 at 6:45pm | Due Sep 6 at 5:30pm | 400 pts

At 6:45 PM.

End Class 4

VH, Download Attendance Report

Rename it:

8.30.2023 Attendance Report FINAL

VH, upload Math Review 1 to CANVAS.