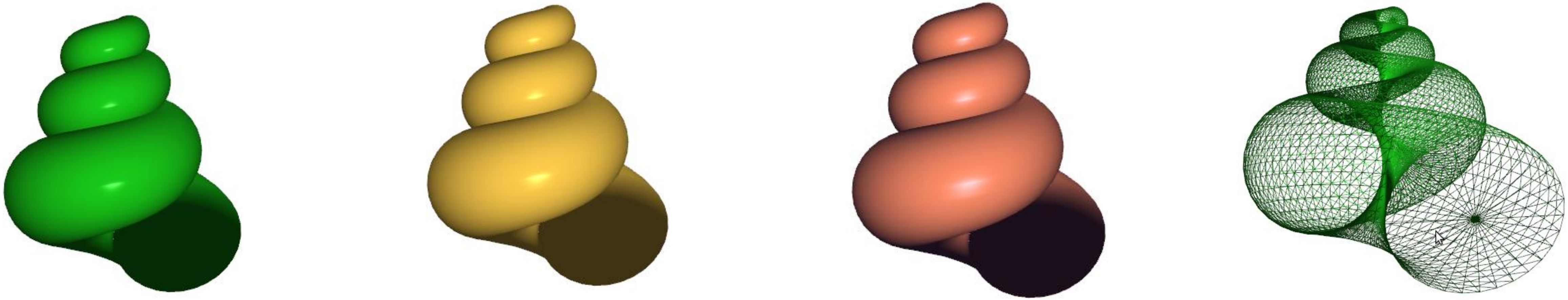
សាកលវិទ្យាល័យភូមិន្ទភ្នំពេញ

អប់រំ ស្រាវជ្រាវ និង សេវាសង្គម

មហាវិទ្យាល័យវិស្វកម្ម

2D Transformation

Department of IT Engineering



Administrivia

- Class
  - Theory: T002
    - Tuesday: 1:00pm – 2:30pm
  - Lab
    - Monday: 1:00pm – 2:30pm (G1)
    - Monday: 2:30pm – 4:00pm (G2)
- Exams
  - Final Exams: 60%
  - Assignment: 20%
  - Homework: 10%
  - Attendance: 10%

Contents

- Geometrical Transformations in 2 Dimension (Modeling Transformation)
  - Translation
  - Scaling
  - Rotation
- Homogenous Coordinates

Transformations in 2 Dimensions

- Important tasks in computer graphics is to transform the coordinates of either
  - Objects within the graphical scene
  - The camera that is viewing the scene
- Transforming the coordinates including
  - Position
  - Orientation
  - Size

2D Transformations

- It is also frequently necessary to transform coordinates from one coordinate system to another
  - Ex: World coordinates to viewport coordinates to screen coordinates
- All of these transformations can be efficiently and succinctly handled using some simple matrix representation
- It is useful for combining multiple transformations into a single composite transform matrix

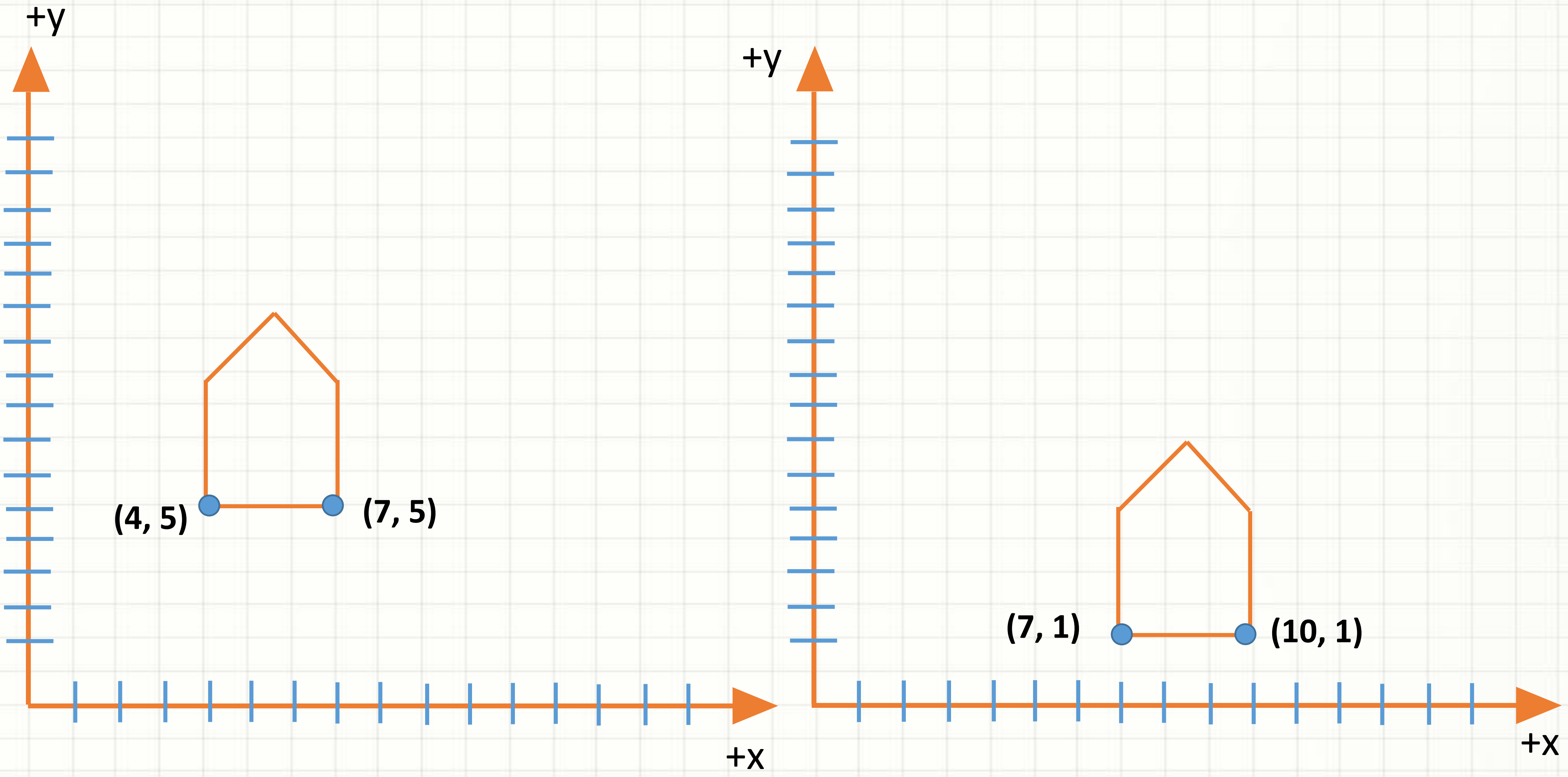
The Pipeline of Transformations

❖ Coordinate transformations

$$\underset{\text{Vertex}}{\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}} \rightarrow \underset{\text{Eye-Coordinates}}{\begin{pmatrix} \texttt{Model-} \\ \texttt{view-} \\ \texttt{Matrix} \end{pmatrix}} \rightarrow \underset{\text{Clip-Coordinates}}{\begin{pmatrix} \texttt{Pro-} \\ \texttt{jection-} \\ \texttt{Matrix} \end{pmatrix}} \rightarrow \underset{\text{Normal Device Coordinates}}{\begin{pmatrix} \texttt{Pers-} \\ \texttt{pective} \\ \texttt{Devision} \end{pmatrix}} \rightarrow$$

$$\begin{pmatrix} \texttt{Viewport-} \\ \texttt{Trans-} \\ \texttt{formation} \end{pmatrix} \rightarrow \begin{pmatrix} \texttt{Window-} \\ \texttt{Coordinates} \end{pmatrix}$$

2D Translation



2D Translation

- Translate points in the **(x, y)** plane to new positions by simply adding translation amounts to the coordinates

- Point $\boldsymbol{P(x, y)}$ is to be translated by amount $\boldsymbol{d_x}$ and $\boldsymbol{d_y}$ to a new location $(\boldsymbol{x', y'})$

$$\boldsymbol{x' = d_x + x}$$
$$\boldsymbol{y' = d_y + y}$$

- Or we can write

$$\boldsymbol{P' = \begin{bmatrix} x' \\ y' \end{bmatrix}}$$

$\boldsymbol{P' = T + P}$ where $\boldsymbol{T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}}$

$$\boldsymbol{P = \begin{bmatrix} x \\ y \end{bmatrix}}$$

2D Scaling

- Points can be scaled by s_x along the x axis and by s_y along the y axis into new points

$$x' = s_x * x$$

$$y' = s_y * y$$

- Or we can write

$$P' = S * P$$

where

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

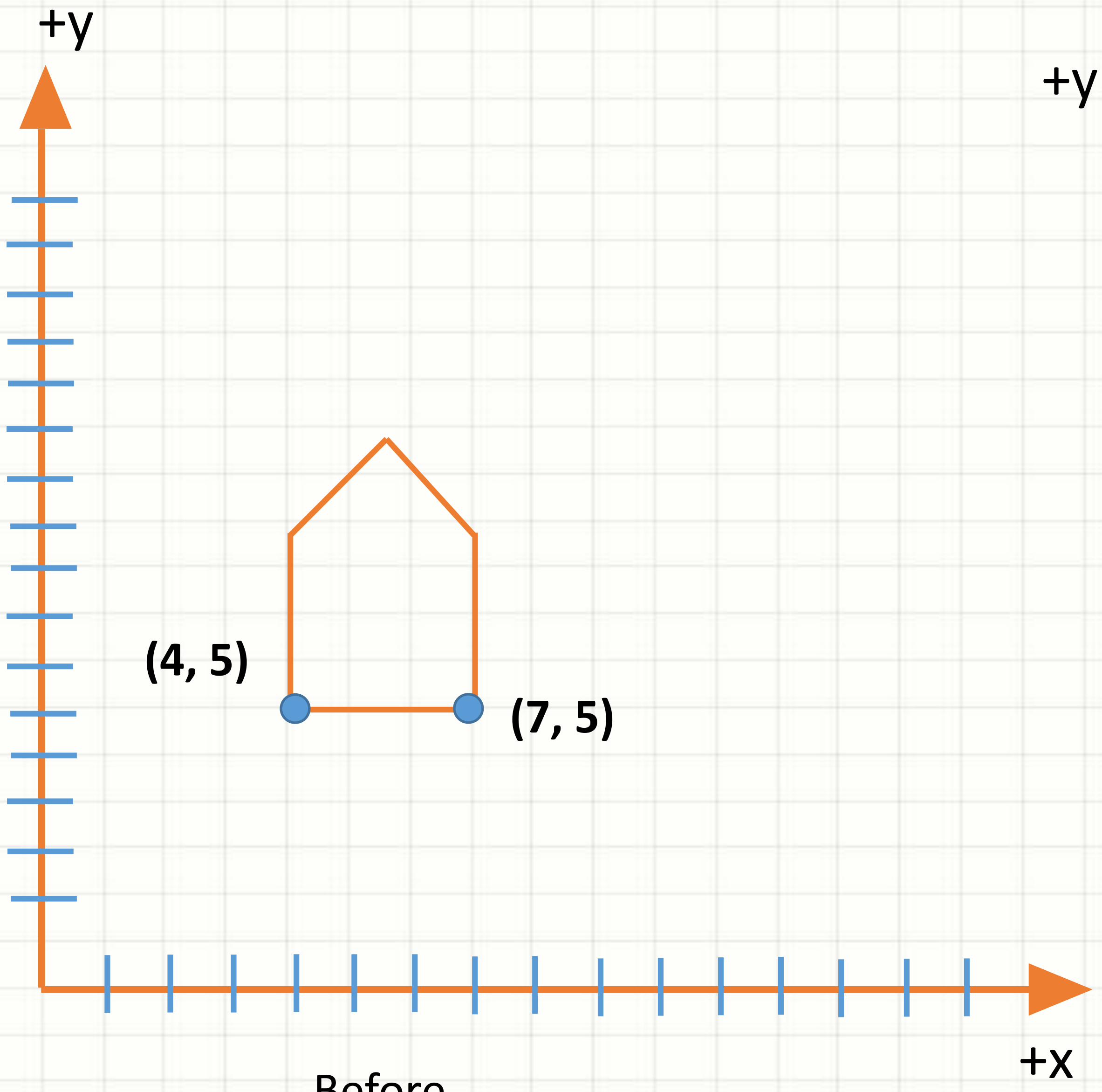
$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix}$$

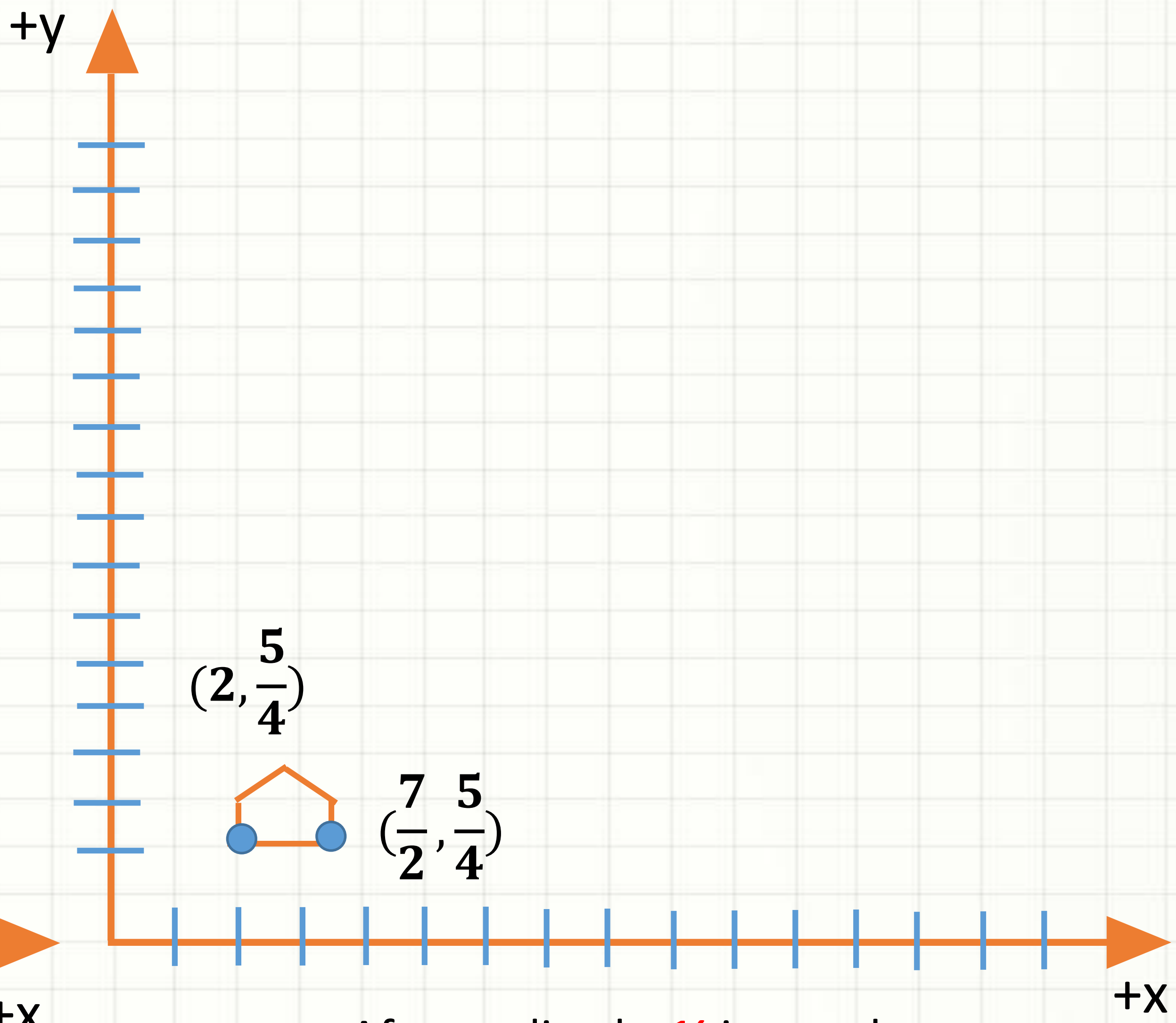
2D Scaling

- Scaling is performed about the origin (0, 0) not about the center of the line/polygon/whatever
- **Scale factor > 1** enlarge the object and move it away from the origin
- **Scale factor = 1** leave the object alone
- **Scale factor < 1** shrink the object and move it towards the origin
- Uniform scaling: $s_x = s_y$
- Differential scaling: $s_x \,! = s_y$ ➔ alters proportions

2D Scaling



Before scaling



After scaling by ½ in **x** and ¼ in **y**

2D Rotation

- Point $P(x, y)$ is to be rotated about the origin by angle theta to location (x', y')

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

(R1)

- Note that this does sin and cos which are much more costly than addition or multiplication

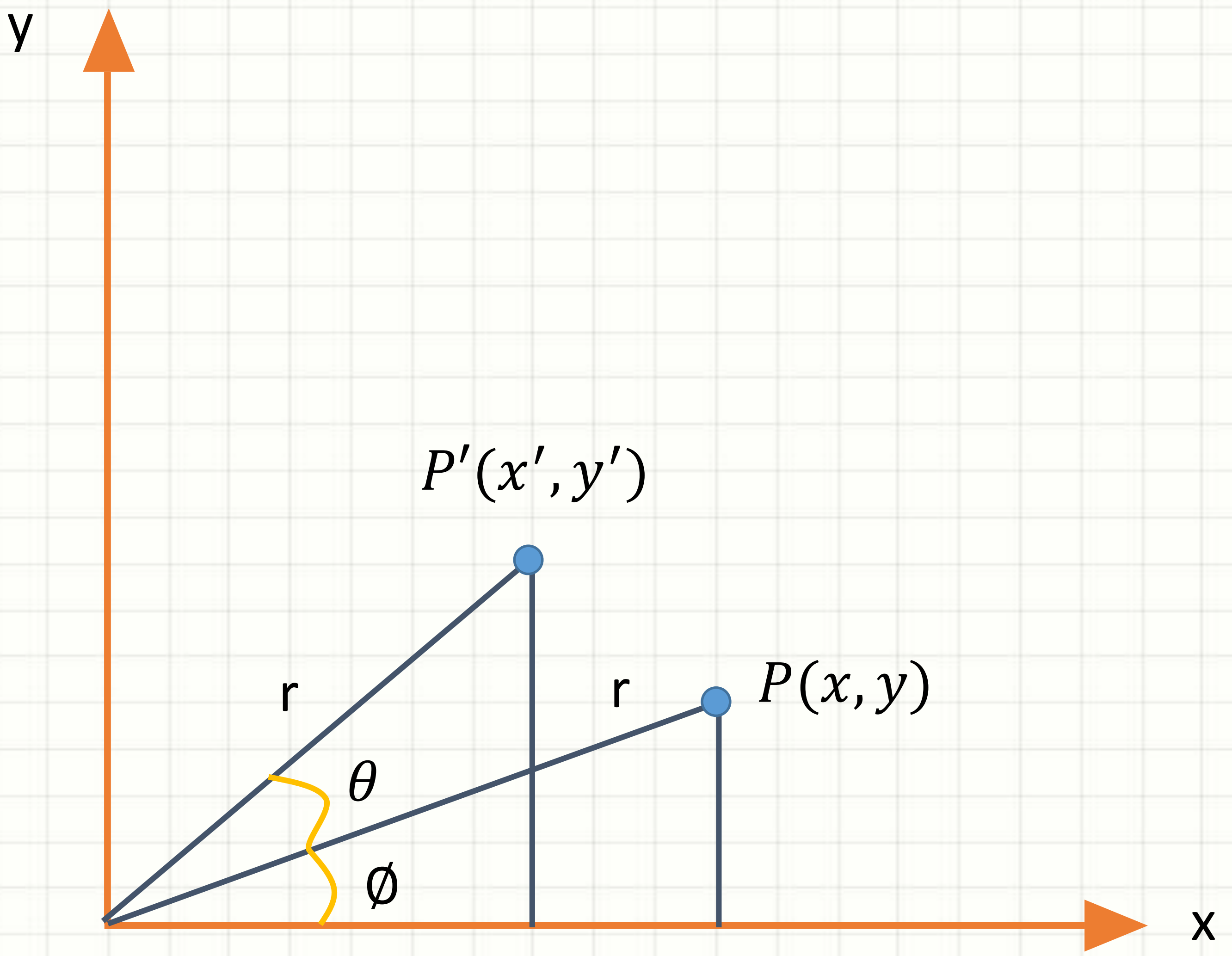
$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$P' = R\ *\ P$ where $R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$

$$P\ = \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Rotation

- Where the equation ***R1*** derive from?



2D Transformation

- The matrix representations for translation, scaling, and rotation are respectively

$$P' = T + P$$

$$P' = S \,.\, P$$

$$P' = R \,.\, P$$

- Scaling and rotations are both handled using matrix multiplication, which can be combined
- Unfortunately, the translations cause a difficulty since they use addition instead of multiplication
- The solution is to express points in ***homogeneous coordinates***
- Point is expressed in ***homogeneous coordinates*** by simply adding a third coordinate ***(x, y, w)***

Homogeneous Coordinates

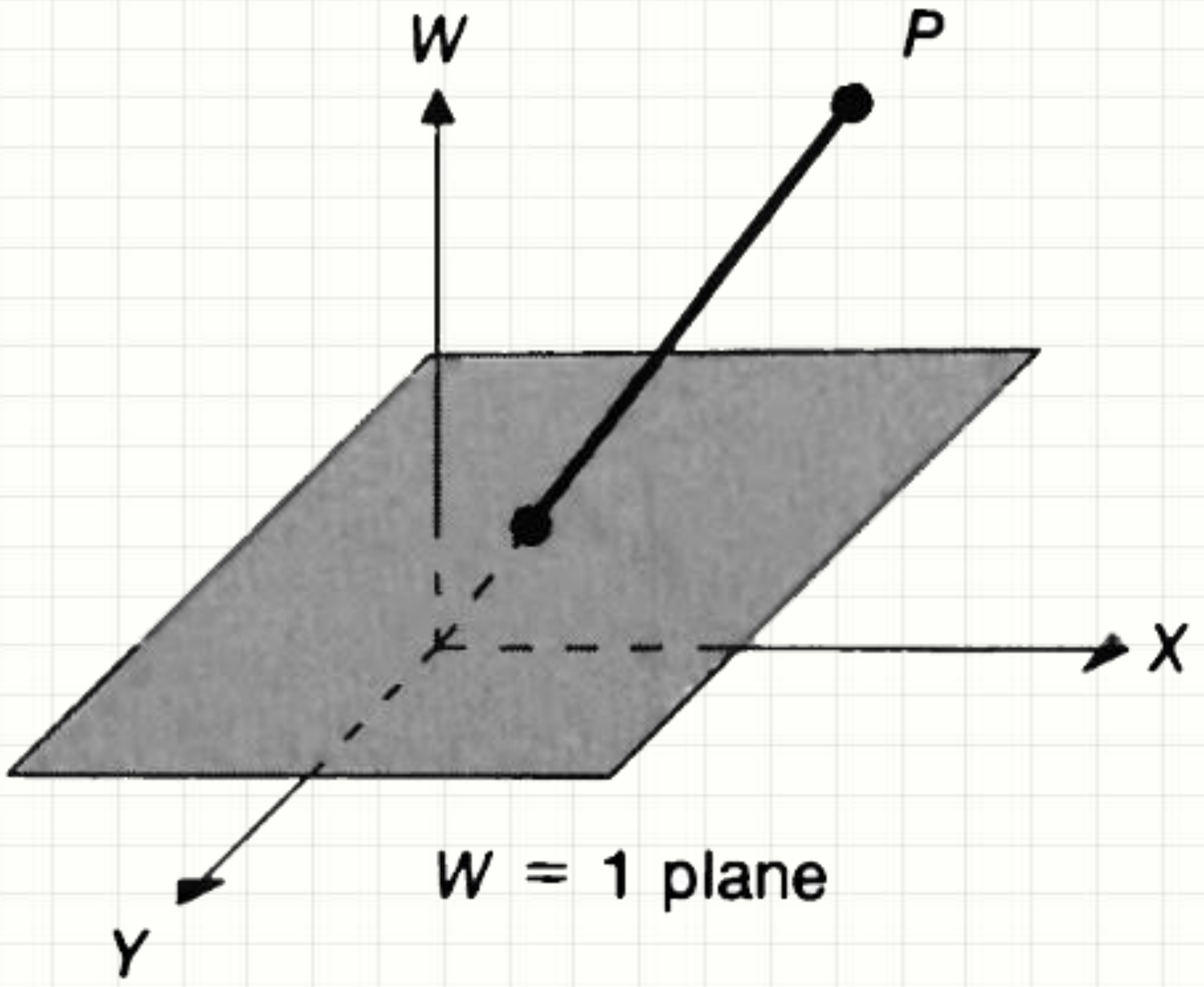
- Expressing points in ***homogeneous coordinates*** allow translations to be handled as a multiplication
- Note that we are not really moving into the third dimension yet
- The third coordinate is being added to the mathematics solely in order to combine the addition and multiplication of 2D coordinates
- Two triples (x, y, w) and (x', y', w') represent the same point if they are multiples of each other
  - **Ex: (1, 2, 3)** and **(2, 4, 6)**

Homogeneous Coordinates

- That is, each point has many different homogeneous coordinate representations
- At least one of the homogeneous coordinates must be nonzero: **(0, 0, 0)** is not allowed
- If **W** is **0** then the point is at infinity. This situation will rarely occur in practice in computer graphics
- If **W** is nonzero we can divide the triple by **W** to get the Cartesian coordinates of **X** and **Y** of the identical point **(X/W, Y/W, 1)**
- This step can be considered as mapping the point from 3D space onto the plane **W=1**

Homogeneous Coordinates

- Conversely, if the 2D Cartesian coordinates of a point are known as **(X, Y)**, then the homogenous coordinates can be given as **(X, Y, 1)**



Translation of 2D Homogenous Coordinates

❖ Point $P(x, y)$ is to be translated by amount Dx and Dy to location (x', y')

$$x' = Dx + x$$

$$y' = Dy + Y$$

$$P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$P' = T * P$$ where

$$T = \begin{bmatrix} 1 & 0 & Dx \\ 0 & 1 & Dy \\ 0 & 0 & 1 \end{bmatrix} = T(Dx, Dy)$$

$$P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling of 2D Homogenous Coordinates

- Points can be scaled by s_x along the x axis and by s_y along the y axis into new points

$$P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$P' = S * P \quad \text{where} \quad S = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} = S(Sx, Sy)$$

$$P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation of 2D Homogenous Coordinates

- Point $P(x, y)$ is to be rotated about the origin by angle theta to location (x', y')

$$P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$P' = R(\theta) \ * \ P$$

where

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

$$P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

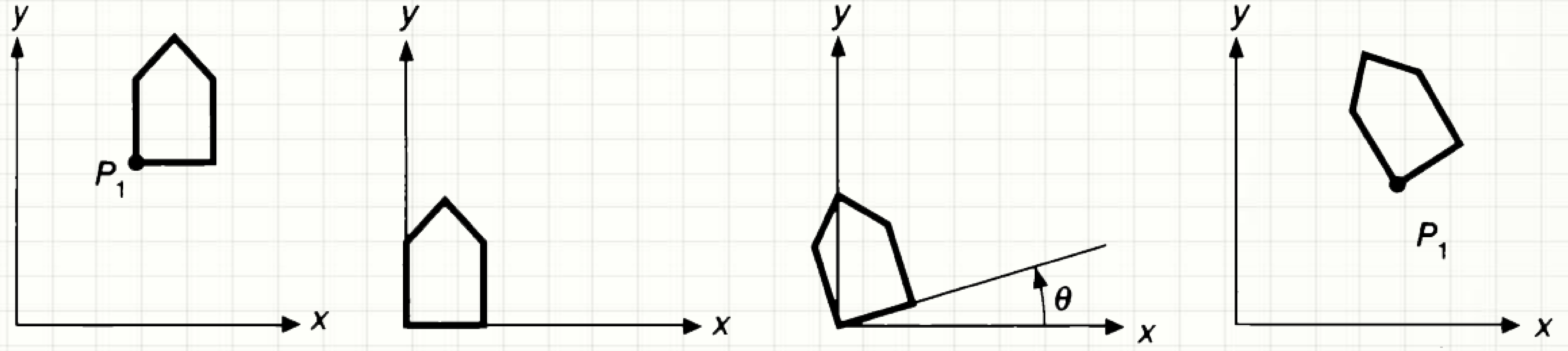
Composition of 2D Transformation

- There are many situation in which the final transformation of a point is a combination of several individual transformation
- It is possible to use composition to combine the fundamental **R, S,** and **T** matrices
- Suppose we need to rotate a polygon about an arbitrary point P_1 rather than around the origin
  - Translate so that P_1 is at the origin $T(-D_x, -D_y)$
  - Rotate $R(\theta)$
  - Translate such that the point at the origin back to P_1 $T(D_x, D_y)$

$$P' = T(D_x, D_y) * R(\theta) * T(-D_x, -D_y) * P$$

Note: The rightmost matrix is the operation that occurs first

Composition of Rotation about Arbitrary Point



Original house

After translation of P_1 to origin

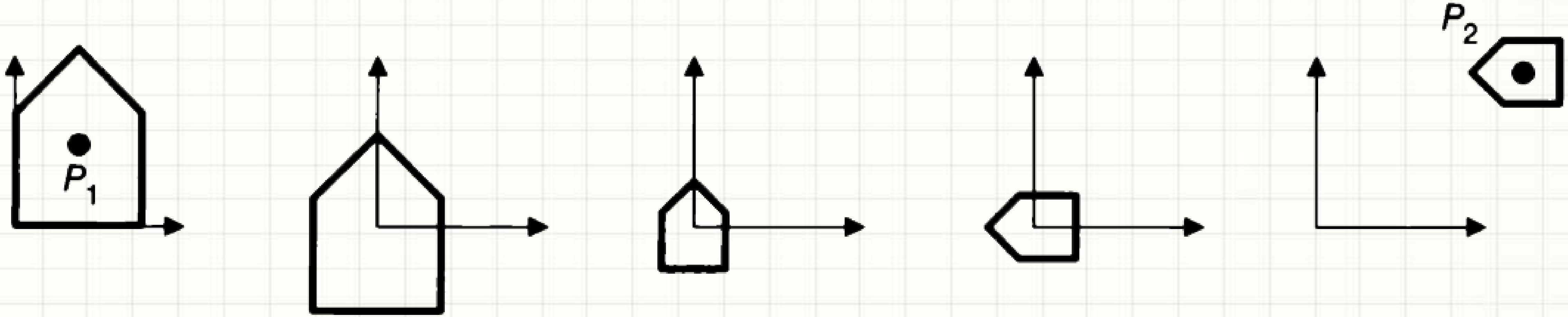
After rotation

After translation to original P_1

$$T(Dx, Dy) * R(\theta) * T(-Dx, -Dy) = \begin{bmatrix} 1 & 0 & D_x \\ 0 & 1 & D_y \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -D_x \\ 0 & 1 & -D_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta & -sin\theta & D_x(1 - cos\theta) + D_y sin\theta \\ sin\theta & cos\theta & D_y\,(1 - cos\theta) - D_x sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Composition of Scaling about an Arbitrary Point



Original house | Translate P_1 to origin | Scale | Rotate | Translate P_1 to final position P_2

$$T(Dx, Dy) * S(S_x, S_y) * T(-Dx, -Dy) = \begin{bmatrix} 1 & 0 & D_x \\ 0 & 1 & D_y \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -D_x \\ 0 & 1 & -D_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & D_x(1 - S_x) \\ 0 & S_y & D_y(1 - S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

Literature

- Foley, J. D., Van Dam, A., Feiner, S.K., Hughes, J. F., & Phillips R. L. (1996). *Introduction to Computer Graphics*.
- Watt A. H. (1990). *Fundamentals of three-dimensional computer graphics*. Addison-Wesley.
- D.H. Eberly, *3D game engine design, a practical approach to real-time computer graphics, Academic Press, Morgan Kaufmann, 2001*
- Hughes, J. F., Van Dam, A., Foley, J. D., & Feiner, S. K. (2013). *Computer graphics: principles and practice.* Pearson Education.
- *Dunn, F., & Parberry, I. (2011). 3D math primer for graphics and game development. CRC Press.*
- *ARB, Dave Shreiner, editor,*
  - *OpenGL programming guide (RED)*