

CS770/870 Spring 2017

Transformations

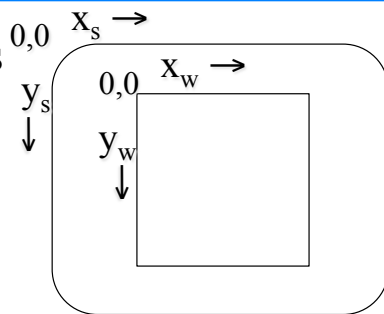
Coordinate systems
2D Transformations
Homogeneous coordinates
Matrices, vectors, points

01/29/2017

1

Coordinate Systems

- Coordinate systems used in graphics
 - *Screen* coordinates: the physical location on a specific display device
 - *Window* coordinates: screen coords relative to window origin
 - *Normalized device coordinates (NDC)*: a device independent abstraction of display window
 - *World coordinates*: a generic coordinate system that any application can use to define objects for display



2

Screen Coordinates

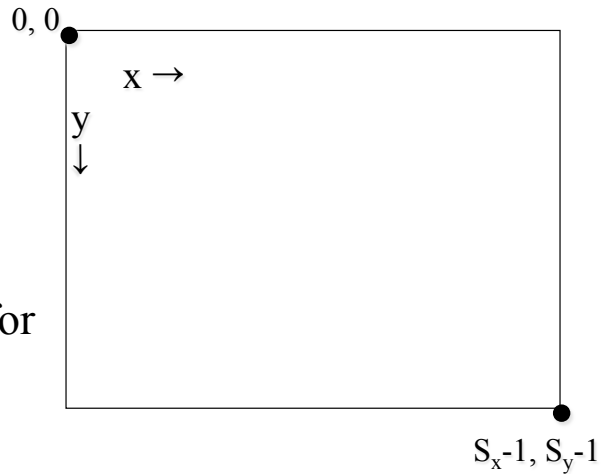
- Physical device coords

- Location of pixels

- S_x by S_y pixels

- 640x480, 1024x768,
1280x800, 1440x900,
etc.

- Variation is difficult for application program; leads to device dependent code



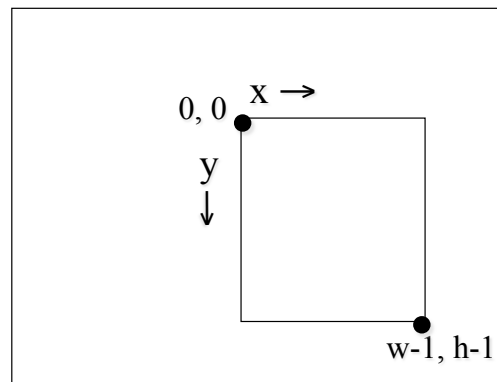
3

GUI Window Coordinates

- GUI windows add another complication

- GUI Window system gives the application a portion of the screen that is $w \times h$ pixels, for arbitrary w and h

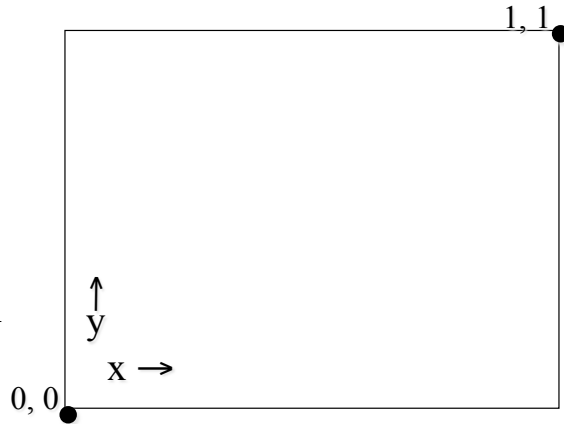
- Window system maps window coordinates to screen coordinates



4

Normalized Device Coordinates

- An *abstraction* of display device coordinates (screen and/or GUI window)
- Floating point (x,y) coordinates from 0 to 1
- y goes up!



5

World Coordinates

- The application should be able to define objects in a way natural for the domain:
 - 0 to 100 feet for a house floor plan
 - 0 to 4000 miles for a map of the US
 - etc.
- *World coordinates* are simply 2D or 3D *float* or *double* values that can represent locations and sizes of object to be displayed

6

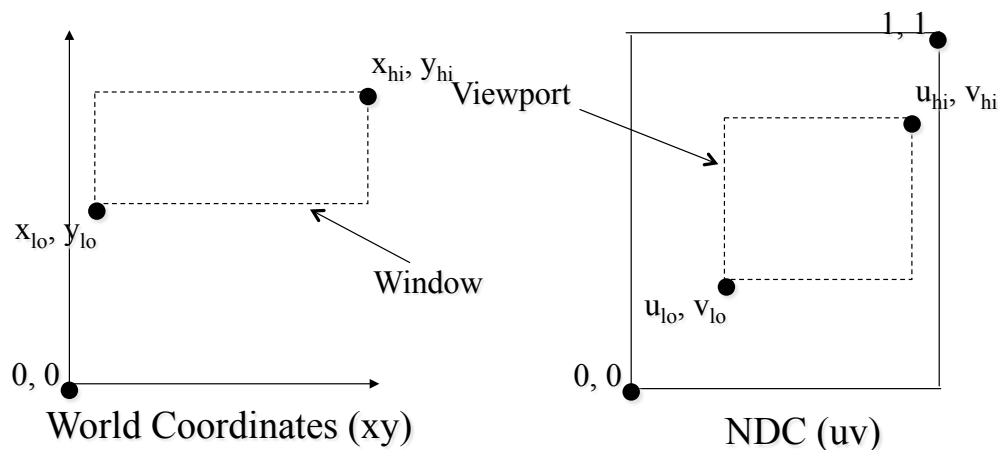
Mapping World Coordinates to Device Coordinates

- Application needs to tell the graphics system:
 - what portion of the world is to be displayed, called the window on the world.
 - where on the screen the window should be mapped, called the viewport.
- In 2D
 - the window is a rectangle in world coordinates
 - the viewport is a rectangle in normalized device coordinates
 - Some graphics systems allow a rotation as well

7

Window to Viewport Mapping

- Window: (x_{lo}, y_{lo}) to (x_{hi}, y_{hi})
- Viewport: (u_{lo}, v_{lo}) to (u_{hi}, v_{hi})



8

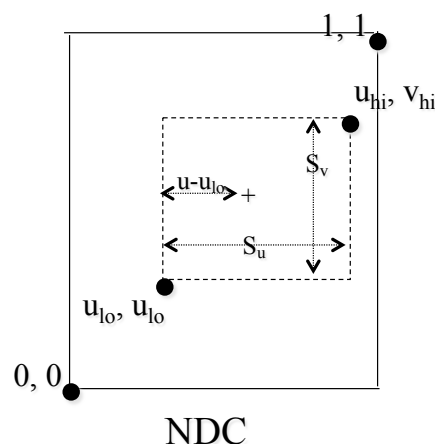
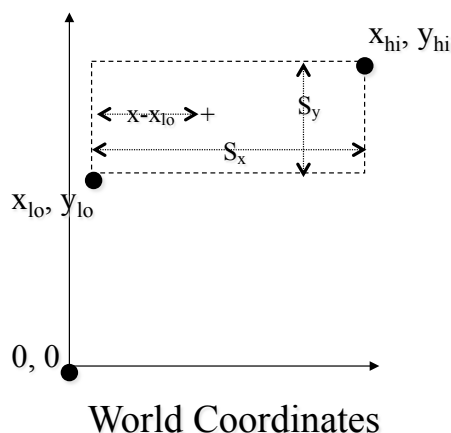
Window to Viewport Mapping

- Given (x,y) in world, what is its (u,v) in ndc?
 - Consider the x coordinate first
 - Let S_x be the width of the window = $x_{hi} - x_{lo}$
 - Let S_u be the width of the viewport = $u_{hi} - u_{lo}$
 - what % of the way from x_{lo} to x_{hi} is x ?
$$xratio = (x - x_{lo}) / S_x$$
 - it is exactly that % of the way from u_{lo} to u_{hi} in ndc:
$$uratio = (u - u_{lo}) / S_u$$
 - so, $(u - u_{lo}) / S_u = (x - x_{lo}) / S_x$
 - or, $u = u_{lo} + (x - x_{lo}) / S_x * S_u$
 - Similarly, $v = v_{lo} + (y - y_{lo}) / S_y * S_v$

9

Window to Viewport Mapping 2

- $u = u_{lo} + (x - x_{lo}) / S_x * S_u$
- $v = v_{lo} + (y - y_{lo}) / S_y * S_v$



10

Clipping

- What if (x,y) is outside the window?
 - it should not be visible in the viewport!
 - can't just ignore a point outside window; it's part of a line or polygon or more complex object
 - must display only the portion of each object that is inside the window
 - must do it efficiently!
 - we'll re-visit clipping later

11

2D Transformations

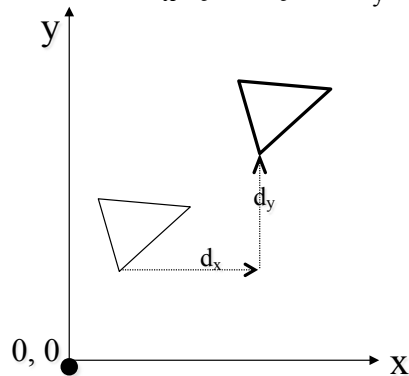
- We'd like to *define* an object with 1 set of coordinate specifications, then *transform* it to another location, size, and/or orientation.
- Translation by (d_x, d_y)
- Scale by (s_x, s_y)
- Rotation by α about the origin $(0,0)$

12

2D Translation

- Translation by (d_x, d_y)
for each (x, y) in the object

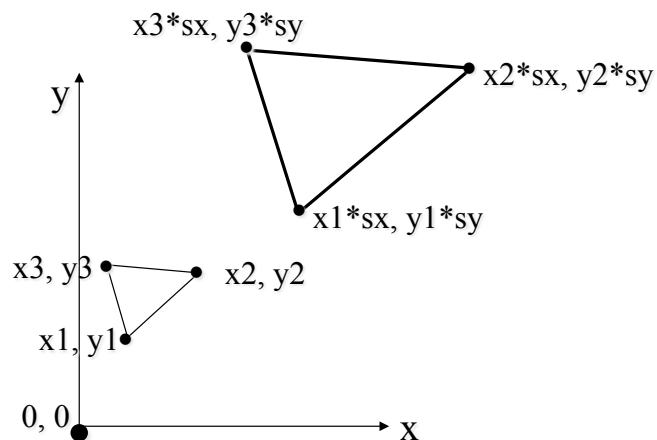
$$x' = x + d_x; y' = y + d_y$$



2D Scale

for each (x, y) in the object

$$x' = x * s_x; y' = y * s_y$$



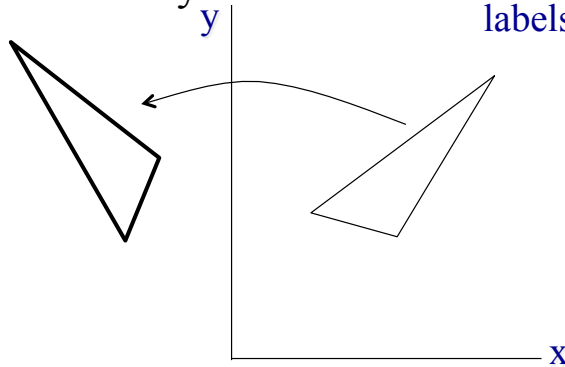
2D Rotation

- Rotation of α degrees about the origin, where $\alpha > 0$ is counter clockwise rotation

$$x' = x \cos \alpha - y \sin \alpha$$

$$y' = x \sin \alpha + y \cos \alpha$$

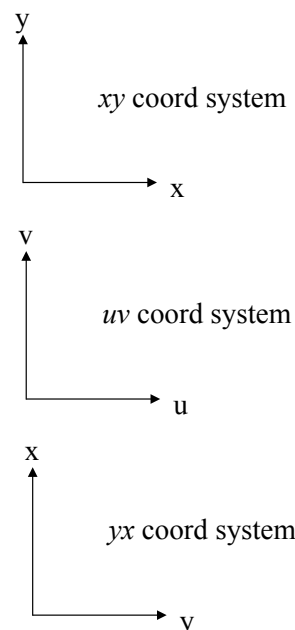
Note: equations only make sense if the axes labels are known!



15

2D Axis Ordering Conventions

- If we say the axes are labeled xy , we assume that x is horizontal, increasing to the right, and y is vertical increasing upward
 - It's pure convention!
- What is a positive angle?
 - if axes are xy , an angle > 0 rotates x -axis towards y
 - another convention.



16

Affine Transformations

- Affine transformations leave the origin (0,0) unchanged
 - Scale and rotation are affine transformations
 - Translation is not affine
- 2D affine transformations can be elegantly represented in matrix format as 2x2 matrices
 - unifies the representation of scale and rotation
 - provides an easy implementation of composition of transformations: rotate, then scale, then rotate

17

Linear Algebra

- Linear algebra isn't scary: it's notation. Consider the 2 linear equations on the right
- The “meaning” of the equations is all in the constants (a,b,c,d); the variables (u,v,x,y) are just placeholders. We encapsulate the constants in a 2x2 **matrix** and the variables as 2x1 **vectors**.

$$u = ax + by$$

$$v = cx + dy$$

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

$$\begin{bmatrix} u \\ v \end{bmatrix} \quad \begin{bmatrix} x \\ y \end{bmatrix}$$

18

Linear Algebra Operators

- To utilize this “notation” efficiently, we can define some operators on matrices and vectors:
 - “Multiplication” of a matrix times a vector let’s us recover the underlying equations
 - The “dot” product (also called *inner* product) let’s us combine 2 vectors into a scalar; it’s definition leads to convenient computations of vector lengths and the angle between two vectors
 - The “cross” product (*outer* product) let’s us easily compute a new vector that is orthogonal to 2 vectors

19

Dot (inner) Product

- Given two vectors of the same length, the dot product is the sum of the products of corresponding entries:

$$\begin{bmatrix} a \\ b \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow ax+by$$

which really should be written as

a row vector * a column vector

$$\begin{bmatrix} a & b \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow ax+by$$

20

Matrix times a vector

- Consider the 2x2 matrix and a 2x1 vector:

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

- The product of the matrix * vector is defined as

$$\begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

- Each entry is the dot product of a row of the input matrix with the input vector (thus the row*col form of the dot product definition)

21

Matrix times Matrix

- The row/column dot product convention extends to matrix products:
- Row i of the left operand is dotted with column j of the right operand to get position (i,j) in the result:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} r & s \\ t & u \end{bmatrix} \rightarrow \begin{bmatrix} ar + bt & as + bu \\ cr + dt & cs + du \end{bmatrix}$$

- All this extends easily to 3D and more

22

Matrices for Affine Transforms

- Scale: $x' = xS_x$, $y' = yS_y$

$$\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$
- Rotation: $x' = x \cos \alpha - y \sin \alpha$, $y' = x \sin \alpha + y \cos \alpha$

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

Assumes axes are labeled xy .
 +90 rotation: $\cos 90 = 0$, $\sin 90 = 1$

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

x axis $(1,0) \Rightarrow y$ axis $(0,1)$
 y axis $(0,1) \Rightarrow -x$ axis $(-1,0)$

23

Matrix Multiplication

- Matrix multiplication is just a series of matrix-vector multiplications
 - Each column of the right hand matrix is multiplied by the left to get the next column of result

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} u & x \\ w & y \end{bmatrix} \rightarrow \begin{bmatrix} au + bw & ax + by \\ cu + dw & cx + dy \end{bmatrix}$$

- Matrix multiplication is associative
 $(M1 * (M2 * V)) = (M1 * M2) * V$

24

Transformation Composition

- Suppose you want to scale an object by (s, t) , then rotate by α degrees
 - Create S and R matrices
 - Let $M = R * S$
 - foreach point in object, p
$$\text{newP} = M * p$$
- Simpler and more efficient than first transforming all points by S and then transforming the results by R . Why?

25

Are Affine Transforms Symmetric?

- Given two affine transforms, M_1 and M_2
 - Does $M_1 * M_2 = M_2 * M_1$?
 - Consider $S = \text{scale by } (1, 2)$, $R = \text{rotate by } 90$
 - Does $S * R = R * S$
 - Surely, not.
 - What about 2 rotations?
 - yes (in 2D, but not 3D)
 - What about 2 scales?
 - yes
 - Rotation and uniform scale?
 - yes

26

THE Problem

- Translation is not affine
 - but we can't live without it!
- How can we combine translation with scale and rotate into one unified model?
- Homogeneous coordinates
 - We'll present it as a convenient engineering “trick”
 - There is a rigorous mathematical explanation

27

Translation Transformation

- $x' = x + d_x, \quad y' = y + d_y$
- Consider the 3x3 matrix and the 3x1 vector:
$$\begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
- The product of this matrix times the vector is:
$$\begin{bmatrix} x + d_x \\ y + d_y \\ 1 \end{bmatrix}$$

It's just what we want (if we ignore the extra coord)!

But! what about scale and rotation?

28

Revised Scale/Rotate Matrices

- Expand the Scale and Rotate matrices to 3x3

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

$$\begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

You should verify that multiplying these matrices by $[x \ y \ 1]$ produce same results as before.

29

Homogeneous Coordinates

- The 3x3 matrix representation for 2D transformations is called *homogeneous coordinates*
- The 3rd coordinate is called the homogeneous coordinate and is usually designated with a w
- In this model, rotation, scale and translation can be represented in a single uniform notation and can be composed in arbitrary order

30

Why does it work?

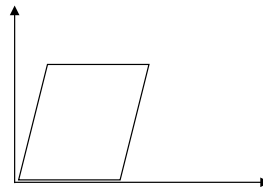
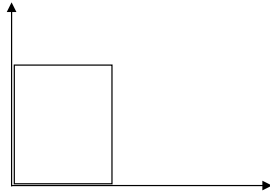
- There is a complex mathematical explanation and
- an intuitive geometric one.

Which do you want?

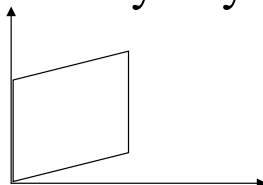
31

2D Shear Transform

- Shear of x by a factor of y: $x' = x + y * h_{xy}$
 - turns a rectangle into a parallelogram



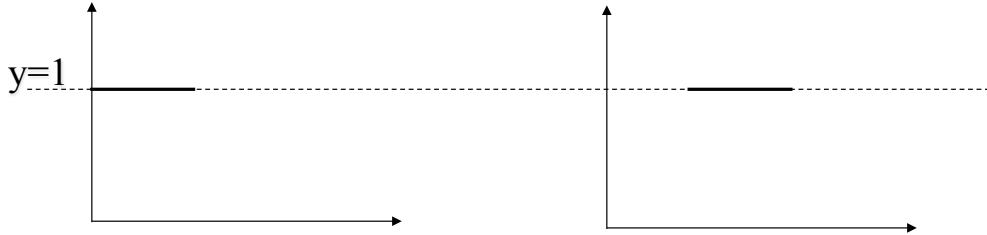
- Shear of y by a factor of x: $y' = y + x * h_{yx}$
 - similar result



32

What is a 2D shear in 1D?

- Suppose you represent all your objects with $y=1$; this becomes a 1D world with scale and translation operations.
- A 2D shear becomes a 1D translation at $y=1$



- Similarly, a 3D shear at $w=1$ is a 2D translation in the $w=1$ plane

33

2D Shear Matrix

- 2x2 shear matrix

$$\begin{bmatrix} 1 & h_{xy} \\ h_{yx} & 1 \end{bmatrix}$$

- 3x3 homogeneous shear matrix for 2D shear

$$\begin{bmatrix} 1 & h_{xy} & 0 \\ h_{yx} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Verify that these are correct

34

3D Shear Matrix

- A complete 3D shear matrix is

$$\begin{bmatrix} 1 & h_{xy} & h_{xz} \\ h_{yx} & 1 & h_{yz} \\ h_{zx} & h_{zy} & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & h_{xw} \\ 0 & 1 & h_{yw} \\ 0 & 0 & 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$$

- But, when we use the matrix as a 2D homogeneous matrix, all the factors except h_{xz} and h_{yz} are 0
 - and those should be renamed as h_{xw} and h_{yw}
 - or even d_x , d_y

35

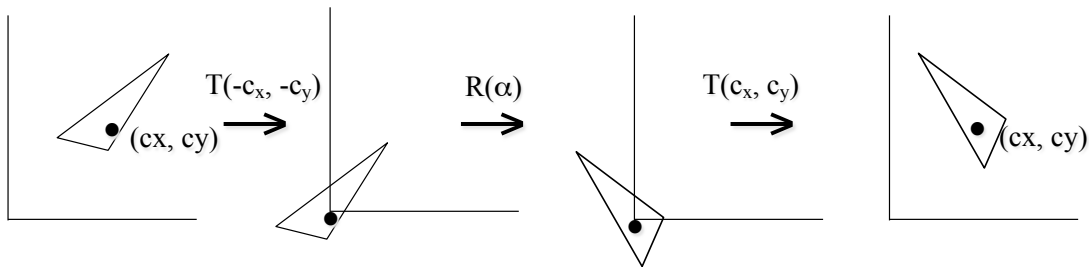
Scale/Rotation about a Point

- Scale/Rotation leave the origin fixed
 - we say that they occur “about the origin”
- Often, want the transformation to occur about an arbitrary point, perhaps the object's center
- Can do this with a composite transformation that:
 - translates the desired center point to the origin
 - applies the scale or rotate (or both)
 - translates the origin back to the original center point
- The composite is applied to all points

36

Arbitrary Rotation

- Rotate by α about (c_x, c_y)
 - $T(-c_x, -c_y)$: Translate (c_x, c_y) to $(0, 0)$
 - $R(\alpha)$: Rotation by α
 - $T(c_x, c_y)$: Translate $(0, 0)$ to (c_x, c_y)



37

Arbitrary Rotation Matrices

- Rotate by α about (c_x, c_y)
 - T_1 , Translate (c_x, c_y) to $(0, 0)$

$$T_1 = \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{bmatrix}$$
 - R , Rotation
$$R = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
 - T_2 , Translate $(0, 0)$ to (c_x, c_y)

$$T_2 = \begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix}$$
- Composite $C = T_2 R T_1$
 - for each point P : $P' = CP$

38

Multiplication Exercise

• $C = T_2 R T_1$

1. $T_2 * R$
 $r_0 * c_0$
 $r_0 * c_1$
 $r_0 * c_2$
 etc

2. $(T_2 R) * T_1$

$$C = \begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos \alpha & -\sin \alpha & c_x \\ \sin \alpha & \cos \alpha & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos \alpha & -\sin \alpha & -c_x \cos \alpha + c_y \sin \alpha + c_x \\ \sin \alpha & \cos \alpha & -c_x \sin \alpha - c_y \cos \alpha + c_y \\ 0 & 0 & 1 \end{bmatrix}$$

39

Review/Preview

- Review
 - Coordinate systems
 - 2D Transformations
 - Homogeneous coordinates
 - Matrices, vectors, points
- Preview
 - 3D Transformations
 - Homogeneous coords
 - Matrices, vectors, points

40