Lecture 05:

Transform 2

COMP 175: Computer Graphics

February 21, 2016

Refresher

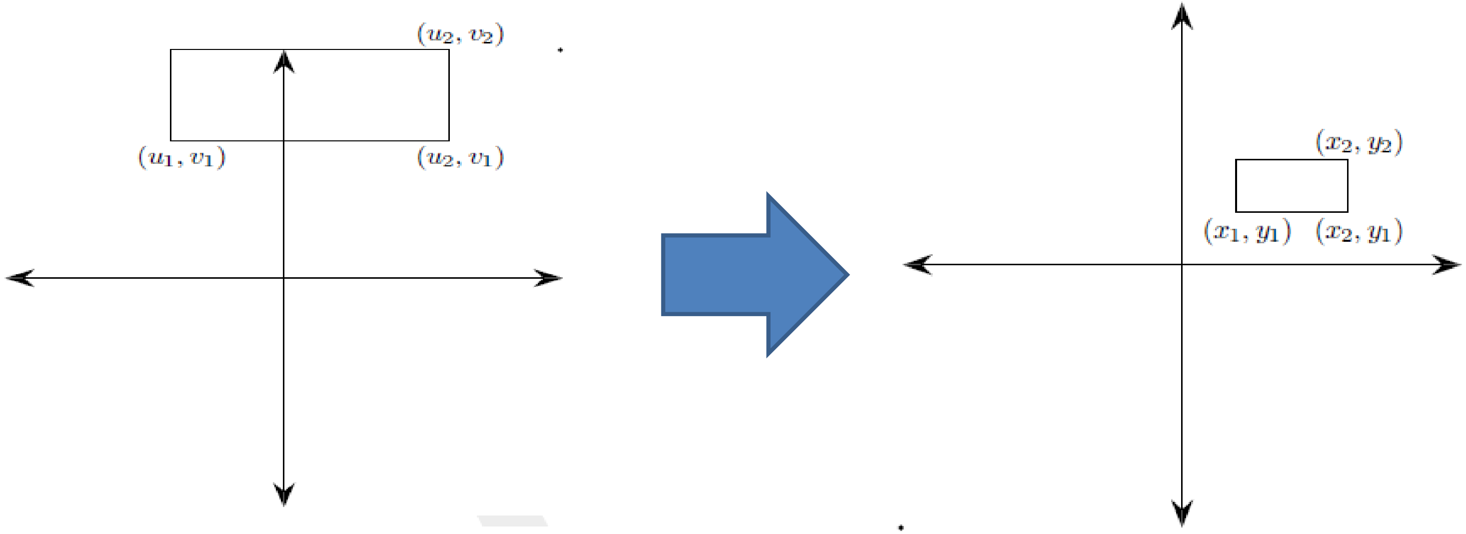
- How do you multiply a (3x3) matrix by 3D vector?
- How do you multiply two 3x3 matrices?
- Given this matrix: $\begin{bmatrix} 0.20 & 0.93 & 0.64 & 1.20 \\ 0.65 & 0.82 & 0.75 & 1.40 \\ 0.95 & 1.10 & 0.52 & 3.20 \\ 1.15 & 0.20 & 1.25 & 2.25 \end{bmatrix}$, what are the 4 basis vectors?
  - How did you find those?
- Matrix as a coordinate transform
  - What happens if you have a 2x3 matrix (2 rows, 3 columns) and we multiply it by a 3D vector?
  - What happens if we have a 3x2 matrix (3 rows, 2 columns) and we multiply it by a 3D vector?

Refresher

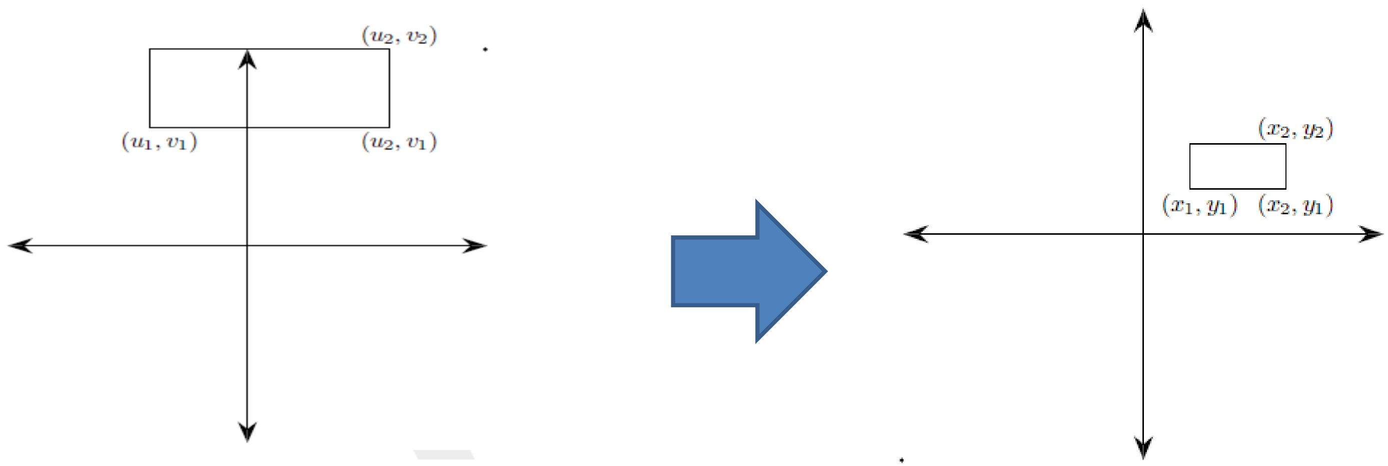
- Write out the basic forms for a 2D:
  - Scaling matrix
  - Rotation matrix
  - Translation matrix
- Why is Translation transform affine but not linear?
- Why do we use 4x4 Matrices in 3D?
- How are Points and Vectors represented differently, and why?

Exercise

- Window Transformations:
  - Given a window (rectangle) with bounding coordinates $(u_1, v_2), (u_2, v_2)$
  - Create a matrix that can both move and scale this matrix so that the new bounding box is $(x_1, y_1), (x_2, y_2)$



Exercise - Solution



- $$\begin{bmatrix} (x_2 - x_1)/(u_2 - u_1) & 0 & (x_1 u_2 - x_2 u_1)/(u_2 - u_1) \\ 0 & (y_2 - y_1)/(v_2 - v_1) & (y_1 v_2 - y_2 v_1)/(v_2 - v_1) \\ 0 & 0 & 1 \end{bmatrix}$$
- This is important! Think about what this means for 2D graphics and visualization.
- In 3D graphics, this is also commonly used to create viewports. So this matrix is known as the "window-to-viewport" transformation

Questions?

Matrix Inverse

- For Scaling, we have:
  - $v' = Sv$
    - where $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- If I want to go backwards, that is, if I was given x', y', how would I find x and y?
  - In other words, I want:
  - $v = S^{-1}v'$
  - Find S^{-1}

Matrix Inverse

- If I want to go backwards, that is, if I was given x', y', how would I find x and y?
  - In other words, I want:
  - $v = S^{-1}v'$
  - Find S^{-1}
- If we look at it at a component level:
  - $x' = S_x x, \quad and \; y' = S_y y$, then
  - $x = \frac{1}{S_x}x', \quad and \; y = \frac{1}{S_y}y'$

Matrix Inverse

- If we look at it at a component level:
  - $x' = S_x x, \quad and\ y' = S_y y$, then
  - $x = \frac{1}{S_x} x', \quad and\ y = \frac{1}{S_y} y'$
- Let's put that back into a matrix form:
  - $v = S^{-1} v'$
    - where $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$

Matrix Inverse

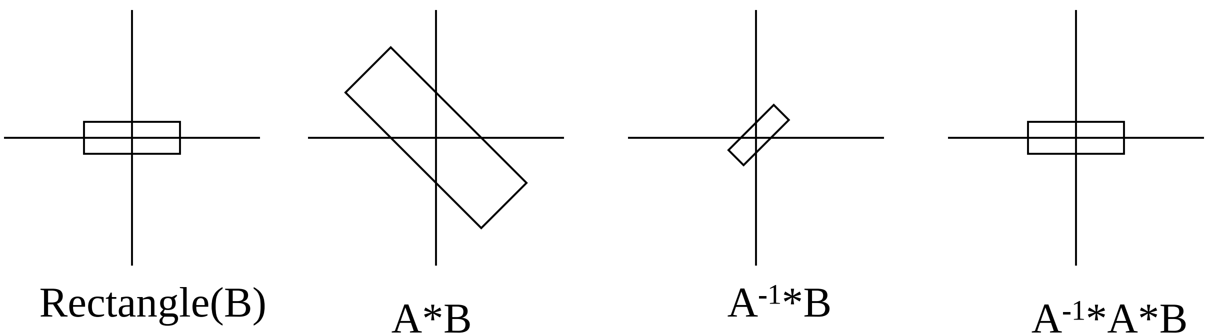
- $S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad S^{-1} = \begin{bmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Notice that if we were to multiply the two together:
  - SS^{-1} or $S^{-1}S$, we get back the identity matrix
    - $I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- In other words: $SS^{-1} = I, = S^{-1}S$

Matrix Inverse

- Definition:
  - $AA^{-1} = I = A^{-1}A$
- Inverting composed matrices:
  - $(AB)^{-1} = B^{-1}A^{-1}$
    - Note, $(AB)^{-1} \neq A^{-1}B^{-1}$
- It is important to note that a matrix is not always invertable. A matrix will not be invertible if:
  - It is not a square matrix (nxn matrix)
  - It has row/column of all zeros (because the row/col can be deleted)
  - If any row/col is a multiple of any other row/col (if a row is not linearly independent)
- Matrices for Rotation, Scaling, Translation (using homogeneous coordinates) will always have inverses!

One Way To Think About Inverses…

- Is to think of an inverse as an “undo”
- For example, if A scales by a factor of 2 and rotates 135 degrees, then A^{-1} will rotate by -135 degrees and scale by 0.5



Finding Inverse Matrices…

- We have found the inverse matrix of a Scaling matrix.
  - $S^{-1} = \begin{bmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Let’s find the inverse matrix of a Translation matrix
  - $T = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}$

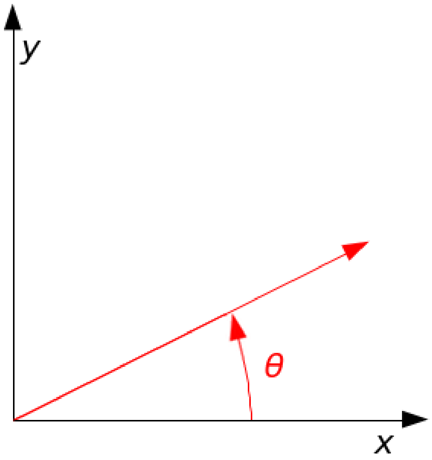
Finding Inverse Matrices…

- This is pretty simple, we just want to "subtract" the change…
  - $T^{-1} = \begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix}$

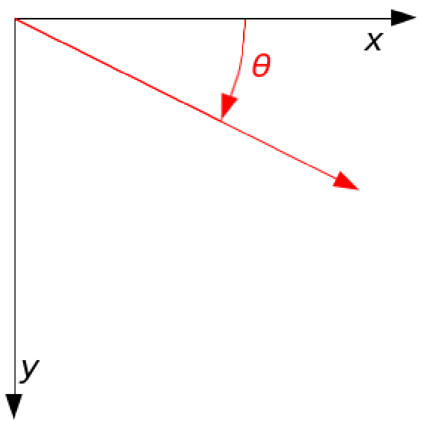
- What about Rotation matrix?
  - $R_\theta = \begin{bmatrix} \cos\theta & -sin\theta & 0 \\ sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Inverse Rotation Matrix

- Regular Rotation Matrix:
  - $R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$



- Inverse Rotation Matrix:
  - $R_\theta^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$



Recap of Inverses

- For Scaling, we have:
  - $v = S^{-1}v'$
    - where $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$
- For Rotation, we have:
  - $v = R_\theta^{-1}v'$
    - where $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & sin\theta & 0 \\ -sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$
- For Translation, we have:
  - $v = T^{-1}v'$
    - where $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$

Questions?

Matrix Transpose

- What is the transpose of a matrix A?

- It's making the rows of the matrix its columns, and its columns become rows
  - Or, you can think of it as: "turning the matrix by 90 degrees

- $A = \begin{bmatrix} v1_x & v1_y & v1_z \\ v2_x & v2_y & v2_z \\ v3_x & v3_y & v3_z \end{bmatrix}, A^T = \begin{bmatrix} v1_x & v2_x & v3_x \\ v1_y & v2_y & v3_y \\ v1_z & v2_z & v3_z \end{bmatrix}$

Neat Fact about Rotation Matrix

- The inverse of a Rotation Matrix R_θ is the same as its transpose R_θ^T.
  - In other words, $R_\theta^{-1} = R_\theta^T$
- Let's prove this… First, we note the properties of $R_\theta = [v_1 \quad v_2 \quad v_3]$:
  - Columns are orthogonal to each other (e.g., $v_1 \cdot v_2 = 0$)
  - Columns represent unit vectors: $||v_i|| = 0$
- Let's multiply R_θ^T by R_θ:

$$\begin{bmatrix} v1_x & v1_y & v1_z \\ v2_x & v2_y & v2_z \\ v3_x & v3_y & v3_z \end{bmatrix} \begin{bmatrix} v1_x & v2_x & v3_x \\ v1_y & v2_y & v3_y \\ v1_z & v2_z & v3_z \end{bmatrix} = \begin{bmatrix} v1 \cdot v1 & v1 \cdot v2 & v1 \cdot v3 \\ v2 \cdot v1 & v2 \cdot v2 & v2 \cdot v3 \\ v3 \cdot v1 & v3 \cdot v2 & v3 \cdot v3 \end{bmatrix}$$

Neat Fact about Rotation Matrix

- Let's multiply R_θ^T by R_θ:

$$\begin{bmatrix} v1_x & v1_y & v1_z \\ v2_x & v2_y & v2_z \\ v3_x & v3_y & v3_z \end{bmatrix} \begin{bmatrix} v1_x & v2_x & v3_x \\ v1_y & v2_y & v3_y \\ v1_z & v2_z & v3_z \end{bmatrix} = \begin{bmatrix} v1 \cdot v1 & v1 \cdot v2 & v1 \cdot v3 \\ v2 \cdot v1 & v2 \cdot v2 & v2 \cdot v3 \\ v3 \cdot v1 & v3 \cdot v2 & v3 \cdot v3 \end{bmatrix}$$

- Based on our rules, the right hand side comes out to: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, which is the identity matrix. This means that $R_\theta^T R_\theta = I$, by definition, $R_\theta^T = R_\theta^{-1}$

Questions?

Composition of Transformations!

- For Scaling, we have:
  - $v' = Sv$
- $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- For Rotation, we have:
  - $v' = R_\theta v$
- $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -sin\theta & 0 \\ sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- For Translation, we have:
  - $v' = Tv$
- $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Composition of Transformations!

- So, if I want to combine the 3 transformations…
  - $v' = Sv$
  - $v'' = R_\theta v'$
  - $v''' = Tv''$
- This means:
  - $v''' = Tv''$
  - $v''' = T(R_\theta v')$
  - $v''' = T\big(R_\theta(Sv)\big)$
  - $v''' = TR_\theta Sv$

Composition of Transformations!

- $v''' = TR_\theta Sv$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Recall that matrix operations are associative. Meaning that:
  - (1+2)+3 = 1+(2+3),
- But it is not commutative:
  - $1+2+3 \neq 3+2+1$

- This means that I can pre-multiply $TR_\theta S = M$, so that
  - $v''' = Mv$, where $M = \begin{bmatrix} S_x\cos\theta & S_y(-sin\theta) & dx \\ S_x sin\theta & S_y cos\theta & dy \\ 0 & 0 & 1 \end{bmatrix}$

Composition of Transformations!

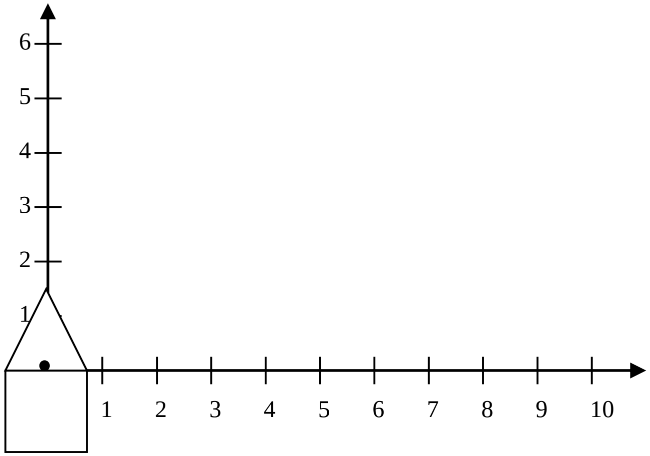
- Remember, **ORDER MATTERS!!**

- So
  - $v' = TR_\theta v \neq R_\theta T v$

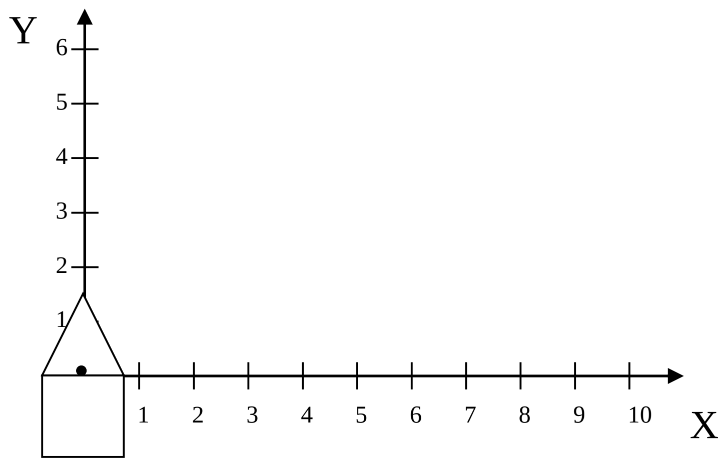
- For Example….

Not commutative

Translate by x=6, y=0 then rotate by 45º



Rotate by 45º then translate by x=6, y=0

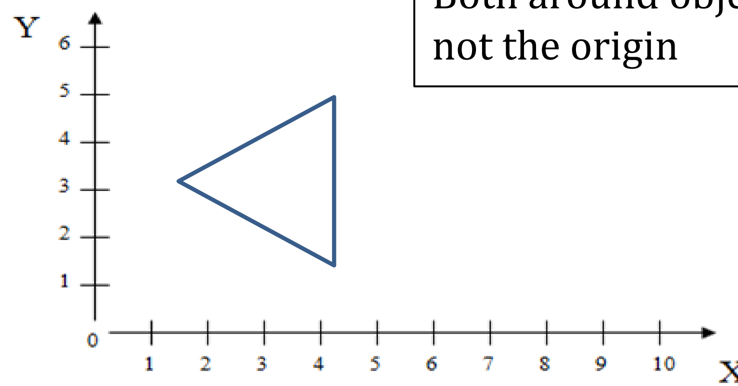


Composition (an example) (2D)

- Start:

Goal:

Rotate 90°
Uniform Scale 3x
Both around object's center,
not the origin



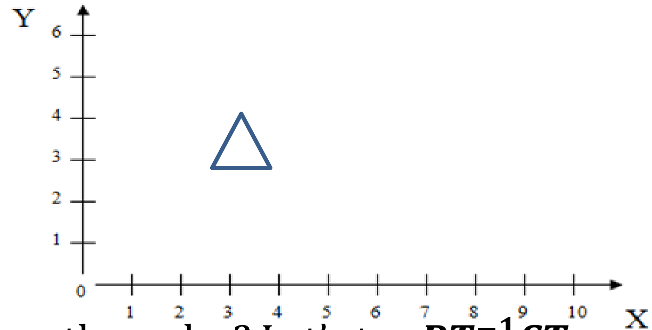
- Important concept: Make the problem simpler
- Translate object to origin first, scale , rotate, and translate back
  $\boldsymbol{T^{-1}RST}$

$$\begin{bmatrix}1 & 0 & 2\\ 0 & 1 & 2\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos 90 & -sin90 & 0\\ sin90 & cos90 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}3 & 0 & 0\\ 0 & 3 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -2\\ 0 & 1 & -2\\ 0 & 0 & 1\end{bmatrix}$$

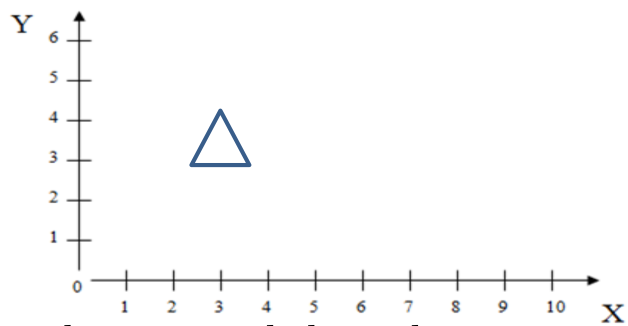
- Apply to all vertices

Composition (an example) (2D) (2/2)

- $\boldsymbol{T^{-1}RST}$



- But what if we mixed up the order? Let's try $\boldsymbol{RT^{-1}ST}$
  - $$\begin{bmatrix} \cos 90 & -sin90 & 0 \\ sin90 & cos90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$



- Oops! We managed to scale it properly but when we rotated it we rotated the object about the origin, not its own center, shifting its position…**Order Matters!**

- http://www.cs.brown.edu/exploratories/freeSoftware/repository/edu/brown/cs/exploratories/applets/transformationGame/transformation_game_guide.html (Transformations applet)

Questions?

Inverse Composite Matrix

- Recall that:
  - $(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$
- Why is that?
- For example:
  - Let $M = TR_\theta$, then $M^{-1} = (TR_\theta)^{-1} = R_\theta^{-1}T^{-1}$
  - In terms of operations, it makes sense:
    - 1) Rotate
    - 2) Translate
  - In reverse, I would want to
    - 1) Reverse Translate
    - 2) Reverse Rotate

Inverses Revisited

- What is the inverse of a sequence of transformations?

$$(\boldsymbol{M_1 M_2 \dots M_n})^{-1} = \boldsymbol{M_n^{-1} M_{n-1}^{-1} \dots M_1}$$

- Inverse of a sequence of transformations is the composition of the inverses of each transformation in reverse order

- Say from our from our previous example we wanted do the opposite, what will our sequence look like?

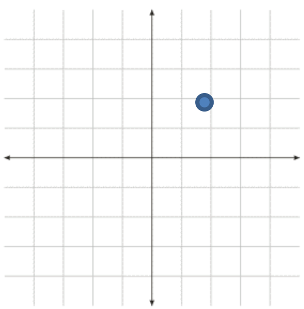
$$\left(\boldsymbol{T^{-1}RST}\right)^{-1} = \boldsymbol{T^{-1}S^{-1}R^{-1}T}$$

- $\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90 & sin 90 & 0 \\ -sin 90 & cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$
- We still translate to origin first, then translate back at the end!

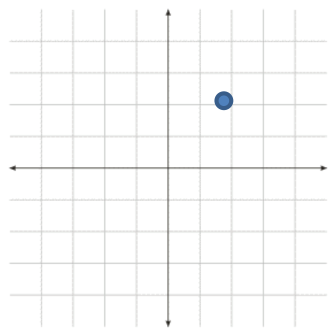
Questions?

Transforming Coordinate Axes

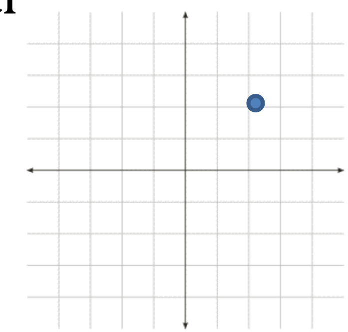
- We understand linear transformations as changing the position of vertices relative to the standard axes
- Can also think of transforming the coordinate axes themselves



Rotation



Scaling



Translation

- Just as in matrix composition, be careful of which order you modify your coordinate system

Mapping It to 3D

- We have been doing everything in 2D. What happens in 3D Cartesian Coordinate System?

Composition of Transformations!

- For Scaling, we have:
  - $v' = Sv$

- $$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- For Rotation, we have:
  - $v' = R_\theta v$

- $$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$ (will be explained)

- For Translation, we have:
  - $v' = Tv$

- $$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation

- Rotation by angle θ around vector $\boldsymbol{w}$ = $\begin{bmatrix} \boldsymbol{w_x} \\ \boldsymbol{w_y} \\ \boldsymbol{w_z} \end{bmatrix}$
- Here's a not so friendly rotation matrix:
- $$\begin{vmatrix} w_x^2 + cos\theta(w_y^2 + w_z^2) & w_x w_y(1 - cos\theta) + w_z sin\theta & w_x w_z(1 - cos\theta) + w_y sin\theta & 0 \\ w_x w_y(1 - cos\theta) + w_z sin\theta & w_y^2 + cos\theta(w_x^2 + w_z^2) & w_z w_y(1 - cos\theta) - w_x sin\theta & 0 \\ w_x w_z(1 - cos\theta) - w_y sin\theta & w_z w_y(1 - cos\theta) + w_x sin\theta & w_z^2 + cos\theta(w_y^2 + w_x^2) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$
- This is called the coordinate form of Rodrigues's formula
- Let's try a different way…

Rotating axis by axis (1/2)

- Every rotation can be represented as the composition of 3 different angles of **CLOCKWISE** rotation around 3 axes, namely
  - x-axis in the yz plane by ψ
  - y-axis in the xz plane by θ
  - z-axis in the xy plane by ϕ
- Also known as Euler angles, makes the problem of rotation much easier to deal with

$R_{xy}(\phi)$	$R_{yz}(\psi)$	$R_{xz}(\theta)$
$\begin{bmatrix} cos\phi & -sin\phi & 0 & 0 \\ sin\phi & cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\psi & -sin\psi & 0 \\ 0 & sin\psi & cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- $\boldsymbol{R_{yz}}$: rotation around the x axis, $\boldsymbol{R_{xz}}$: rotation about the y axis,
  $\boldsymbol{R_{xy}}$: rotation about the z axis
- You can compose these matrices to form a composite rotation matrix

Rotating axis by axis (2/2)

- It would still be difficult to find the 3 angles to rotate by given arbitrary axis w and specified angle ψ
- Solution? Make the problem easier
- **Step 1:** Find a θ to rotate around y axis to put w in the xy plane
- **Step 2:** Then find a $\boldsymbol{\phi}$ to rotate around the z axis to align w with the$x$ axis
- **Step 3:** Rotate by ψ around x axis = w axis
- **Step 4:** Finally, undo the alignment rotations (inverse)
- Rotation Matrix: $\boldsymbol{M} = \boldsymbol{R}_{\boldsymbol{xz}}^{-1}(\boldsymbol{\theta})\boldsymbol{R}_{\boldsymbol{xy}}^{-1}(\boldsymbol{\phi})\boldsymbol{R}_{\boldsymbol{yz}}(\boldsymbol{\psi})\boldsymbol{R}_{\boldsymbol{xy}}(\boldsymbol{\phi})\boldsymbol{R}_{\boldsymbol{xz}}(\boldsymbol{\theta})$

Inverses and Composition in 3D!

- Inverses are once again parallel to their 2D versions…

Transformation	Matrix Inverse
Scaling	$\begin{bmatrix} 1/s_x & 0 & 0 & 0 \\ 0 & 1/s_y & 0 & 0 \\ 0 & 0 & 1/s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Rotation	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\psi & sin\psi & 0 \\ 0 & -sin\psi & cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\phi & sin\phi & 0 & 0 \\ -sin\phi & cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & -sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Translation	$\begin{bmatrix} 1 & 0 & 0 & -dx \\ 0 & 1 & 0 & -dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- Composition works exactly the same way…

Example in 3D!

- Let's take some 3D object, say a cube, centered at (2,2,2)
- Rotate in object's space by 30° around x axis, 60° around y and 90° around z
- Scale in object space by 1 in the x, 2 in the y, 3 in the z
- Translate by (2,2,4)
- Transformation Sequence: $\boldsymbol{T}\boldsymbol{T}_{\mathbf{0}}^{-1}\boldsymbol{R}_{xy}\boldsymbol{R}_{xz}\boldsymbol{R}_{yz}\boldsymbol{S}\boldsymbol{T}_{\mathbf{0}}$, where $\boldsymbol{T}_{\mathbf{0}}$ translates to (0,0)
- $$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos90 & sin90 & 0 & 0 \\ -sin90 & cos90 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos60 & sin60 & 0 \\ 0 & -sin60 & cos60 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos30 & 0 & -sin30 & 0 \\ 0 & 1 & 0 & 0 \\ sin30 & 0 & cos30 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Questions?

How to Invert a Matrix

- We're going to use Gauss-Jordan elimination
- Finding A^{-1} with Gauss-Jordan elimination is done by augmenting A with I to get [A|I], then reducing the new matrix into reduced row echelon form (rref) to get a new matrix. This new matrix will be of the form $[I|A^{-1}]$
- What does rref really mean?
  - If a row does not consist entirely of zeros, then the first nonzero number in the row is a 1. (Call this a leading 1)
  - If there are any rows that consist entirely of zeros, then they are grouped together at the bottom of the matrix.
  - If any two successive rows do not consist entirely of zeros, the leading 1 in the lower row occurs farther to the right than the leading 1 in the higher row
  - Each column that contains a leading 1 has zeros everywhere else.

How to Invert a Matrix

- To transform a matrix into rref we are allowed to perform any of the three elementary row operations. These are:
  - Multiply a row by a nonzero constant
  - Interchange two rows
  - Add a multiple of one row to another row

How to Invert a Matrix, Example

- Given: $A = \begin{bmatrix} 11 & 13 \\ 17 & 19 \end{bmatrix}$, let's find A^{-1}:

1. Augment this with the identity:
   - $[A|I] = \begin{bmatrix} 11 & 13 & 1 & 0 \\ 17 & 19 & 0 & 1 \end{bmatrix}$
2. Row operation 1: multiply row 1 by 1/11
   - $\begin{bmatrix} 1 & \frac{13}{11} & \frac{1}{11} & 0 \\ 17 & 19 & 0 & 1 \end{bmatrix}$
3. Row operation 3: multiply row 1 by -17 and add it to row 2:
   - $\begin{bmatrix} 1 & \frac{13}{11} & \frac{1}{11} & 0 \\ 0 & -\frac{12}{11} & -\frac{17}{11} & 1 \end{bmatrix}$

How to Invert a Matrix, Example

4. Row operation 1, multiply row 2 by –11/12
   - $\begin{bmatrix} 1 & \frac{13}{11} & \frac{1}{11} & 0 \\ 0 & 1 & \frac{17}{12} & -\frac{11}{12} \end{bmatrix}$
5. Row operation 3: multiply row 2 by –13/11 and add to row 1
   - $[I|A] = \begin{bmatrix} 1 & 0 & -\frac{19}{12} & \frac{13}{12} \\ 0 & 1 & \frac{17}{12} & -\frac{11}{12} \end{bmatrix}$
6. Therefore:
   - $A^{-1} = \begin{bmatrix} -\frac{19}{12} & \frac{13}{12} \\ \frac{17}{12} & -\frac{11}{12} \end{bmatrix}$

Questions?

Addendum – Matrix Notation

- The application of matrices in the row vector notation is executed in the reverse order of applications in the column vector notation:
  - $\begin{bmatrix} x \\ y \\ z \end{bmatrix} \leftrightarrow \begin{bmatrix} x & y & z \end{bmatrix}$
- Column format: vector follows transformation matrix:
  - $\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
- Row format: vector precedes matrix and is post-multiplied by it:
  - $\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$
- By convention, we always use **Column Format**

Addendum – Matrix Notation

- Uh… A problem:
  - $$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$
- While:
  - $$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} ax + dy + gz & bx + ey + hz & cx + fy + iz \end{bmatrix}$$

Addendum – Matrix Notation

- In order for both types of notations to yield the same result, a matrix in the row system must be the transpose of the matrix in the column system
  - Recall: $M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$, $M^T = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$
- In order to make the two notations line up:
  - $[x \quad y \quad z] \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} = [ax + by + cz \quad dx + ey + fz \quad gx + hy + iz] \leftrightarrow$

    $\begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

- Be careful! **Different textbooks and different graphics packages use different notations**! First understand which one they use!!

Why Column-Format Notation?

- Because it fits more naturally to OpenGL's stacks
- For example, for a series of matrix operations: $\boldsymbol{T^{-1}RST}$, how would you write this using GL calls?
- What happens when I have multiple objects?
  - For example, the solar system?...

Questions?

Converting Math to OpenGL Code

- Let's say that you have a set of transforms:
  - $M = T^{-1}RST$

- Writing this in OpenGL would look something like:

```
glTranslate3f(-xtrans, -ytrans, -ztrans);
glRotate3f (angle, x_axis, y_axis, z_axis);
glScale3f (xscale, yscale, zscale);
glTranslate3f (xtrans, ytrans, ztrans);
DrawObject();
```

- Or, you can do this in software (SLOW) using the Algebra.h library.

```
Matrix t_invM = inv_trans_mat (transVec);
Matrix rotM = rot_mat (rotVec, angle);
Matrix scaleM = scale_mat(scaleVec);
Matrix tM = trans_mat (transVec);
Matrix composite = t_invM * rotM * scaleM * tM;
for (each vertex in object) {
        Point newPos = composite * vertex->getPosition();
}
```