07-0: Matrices and Translations

- Matrices are great for rotations, reflections, scale
- Can't do translations
 - Matrices can only do linear transformations
 - Translations aren't linear
- Like to do *everything* with matrices
- Solution: Add a dimension

07-1: 4D Homogenous Space

- Extend 3D coordinates (x, y, z) to 4D homogenous coordinates (x, y, z, w)
 - 4th dimension is *not time*
 - Start with extending 2D coordinates (x, y) to 3D homogenous coordinates (x, y, w)

07-2: 3D Homogenous Space

- To convert a point (x, y, w) in 3D Homogenous space into 2D (x, y) space:
 - Place a plane at w = 1
 - (x, y, w) maps to the (x, y) position on the plane where the ray (x, y, w) intersects the plane



07-3: 3D Homogenous Space



07-4: **3D Homogenous Space** 07-5: **3D Homogenous Space**

- Converting from a point in 3D homogenous space to 2D space is easy
 - Divide the x and y coordinates by w
 - What happens when w = 0?

07-6: 3D Homogenous Space

- Converting from a point in 3D homogenous space to 2D space is easy
 - Divide the x and y coordinates by w
 - What happens when w = 0?
 - "Point at infinity"
 - Direction, but not a magnitude

07-7: **3D Homogenous Space**

- For a given (x, y, w) point in 3D Homogenous space, there is a single corresponding point in "standard" 2D space
 - Though when w = 0, we are in a bit of a special case
- For a single point in "standard" 2D space, there are an infinite number of corresponding points in 3D Homogenous space

07-8: 4D Homogenous Space

- We can now extend to 3 (4!) dimensions
- A point in 4D Homogeous space (x, y, z, w) transforms to a point in 3D space by dividing x, y and z by w
- That is, where the line defined by points (0,0,0,0) and (x, y, z, w) intersects the hyperplane at w = 1

07-9: 4x4 Transfromation matrices

- In the 3x3 case, a matrix is a transformation of a 3D vector
- In the 4x4 case, a matrix is a transformation of a 4D vector (which we wil then project back into 3D space)
- Let's look at what happens when we restrict w to be 1:

07-10: 4x4 Transfromation matrices

• Given any 3x3 transformation marix, we can convert it to 4D as follows:

m 1 1	m 10	m 1 0	1	m ₁₁	m_{12}	m_{13}	0	1
m 11 m 21	m 12 m 22	m 13 m 22		^m 21	m_{22}	m_{23}	0	L
m ₃₁	m32	m ₃₃		m ₃₁	m_{32}	m_{33}	0	L
01	02	00	-	L 0	0	0	1.	1

07-11: 4x4 Transfromation matrices

- Now, take any 3D vector $\mathbf{v} = [x, y, z]$, and matrix \mathbf{M}
 - Convert **v** to 4D vector with w = 1
 - Convert M to 4D matrix as above
 - Transform vector using the new matrix
 - Transform back to 3D space
 - Get the same vector as if we had not gone into 4D homogenous space at all

07-12: 4x4 Transfromation matrices

[x, y, z]	$\begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \end{bmatrix}$	$m_{12} \\ m_{22} \\ m_{32}$	$m_{13} \\ m_{23} \\ m_{33}$]
$= [xm_{11} + ym_{21} + zm_{31}, x$	$m_{12} + y$	m ₂₂ + -	zm_{32}, z	$zm_{13} + ym_{23} + zm_{33}$]
$[x,y,z,1]\left[$	$m_{11} \\ m_{21} \\ m_{31} \\ 0$	$m_{12} \\ m_{22} \\ m_{32} \\ 0$	$m_{13} \\ m_{23} \\ m_{33} \\ 0$	0 0 0 1
$= [xm_{11} + ym_{21} + zm_{31}, xm_{31}]$	$x_{12} + yr$	$n_{22} + z$	m32, x1	$n_{13} + ym_{23} + zm_{33}, 1$

07-13: 4x4 Transfromation matrices

- As long as the w component is 1 going in, it will be 1 coming out
 - Easy to go back and forth between 3D coordinates and homogenous 4D coordinates
- We've transformed 3D problem into an equivant 4D problem
 - Why?

07-14: Translation

• Consider the matrix:

Γ	1	0	0	0
	0	1	0	0
	0	0	1	0
	Δx	Δy	Δz	1

• What happens when we put a vector [x, y, z, 1] through this matrix?

07-15: Translation

 $[x, y, z, 1] \left[\begin{array}{rrrr} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{array} \right] = [x + \Delta x, y + \Delta y, z + \Delta z, 1]$

• We can now use matrices to do translations!

07-16: Translation

- But ...
 - We're still just doing matrix multiplication
 - Matrix multiplication does linear transforms
 - Translation is not linear
- What's going on?

07-17: Translation

- We are still doing a linear trasformation of the 4D vector
- We are *shearing* the 4D space
- The resulting projection back to 3D is seen as a translation

07-18: Translation 2D Shape

Transform to 3D Homogenous Space



Shear operation in 3D space

Back to 2D





07-19: Translation

• Recall our matricies for shearing in 3D:

$$\left[\begin{array}{rrrr} 1 & 0 & 0 \\ 0 & 1 & 0 \\ s & t & 1 \end{array}\right]$$

• This is precisely what we are doing when translating!

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix}$$

07-20: Combining Transforms

- Since matrix multiplication is associative, we can combine translation and rotation into a single matrix
- First do a rotation, and then a translation
 - Order is important!
 - Why?



07-21: Combining Transforms bining Transforms

- First rotate, and then translate
- $(\mathbf{v}\mathbf{M}_R)\mathbf{M}_T = \mathbf{v}(\mathbf{M}_r\mathbf{M}_T)$
- What is $\mathbf{M}_r \mathbf{M}_T$?



07-23: Combining Transforms

- First rotate, and then translate
- $(\mathbf{v}\mathbf{M}_R)\mathbf{M}_T = \mathbf{v}(\mathbf{M}_r\mathbf{M}_T)$
- What is $\mathbf{M}_r \mathbf{M}_T$?

$M_R =$	$\begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \\ 0 \end{bmatrix}$	$r_{12} \\ r_{22} \\ r_{32} \\ 0$	$r_{13} \\ r_{23} \\ r_{33} \\ 0$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \\ \end{bmatrix}$	$\begin{array}{c}1\\0\\0\\\Delta x\end{array}$	$\begin{array}{c} 0 \\ 1 \\ 0 \\ \Delta y \end{array}$	$\begin{array}{c} 0\\ 0\\ 1\\ \Delta z \end{array}$	$ \begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \end{array} $] = [r_{11} r_{21} r_{31} Δx	r_{12} r_{22} r_{32} Δy	r_{13} r_{23} r_{33} Δz	0 0 1	
---------	---	-----------------------------------	-----------------------------------	---	--	--	---	---	-------	---------------------------------------	--	--	-------------	--

07-24: Combining Transforms

- Any 4x4 Homogenous matrix can be split into a rotational component and a translation component
 - Upper 3x3 matrix is rotation (which is done first)
 - Bottom row is translation (done second)
- But wait rotation is not always done first!
 - True, but any series of rotations and translations is equivalent to a single rotation followed by a single translation

07-25: Combining Transforms

- Let's look at an example
 - First rotate by $\pi/2$ (90 degrees) counterclockwise
 - Then translate x by +1



07-27: Combining Transforms

- Another example
 - First translate x by +1
 - Then rotate by $\pi/2$ (90 degrees) counterclockwise



• Same as rotating, and then moving up + y





07-30: Combining Transforms

• Rotating by $\pi/4$, then translating 1 unit +x

$\begin{bmatrix} \cos \Theta \\ -\sin \Theta \\ 0 \end{bmatrix}$	$sin \Theta$ $cos \Theta$ 0	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\left[\begin{array}{c}1\\0\\1\end{array}\right]$	$\begin{array}{c} 0 \\ 1 \\ 0 \end{array}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} =$	$\left[\begin{array}{c}\cos\Theta\\-\sin\Theta\\1\end{array}\right]$	$sin \Theta$ $cos \Theta$ 0	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	=	$\begin{array}{c} 1/\sqrt{2} \\ -1/\sqrt{2} \\ 1 \end{array}$	$\frac{1/\sqrt{2}}{1/\sqrt{2}}$ 0	0 0 1]
--	-----------------------------------	---	---	--	---	--	-----------------------------------	---	---	---	--------------------------------------	-------------	---

07-31: Combining Transforms

• Translating 1 unit +x, then rotating by $\pi/4$

Γ	1	0	0] [$\cos \Theta$	$\sin \Theta$	0] [$\cos \Theta$	$\sin \Theta$	0]		$1/\sqrt{2}$	$1/\sqrt{2}$	0]
	0	1	0	$-\sin\Theta$	$\cos \Theta$	0	=	$-\sin\Theta$	$\cos \Theta$	0	=	$-1/\sqrt{2}$	$1/\sqrt{2}$	0
L	1	0	1] [0	0	1	J	$\cos \Theta$	$\sin \Theta$	1		$1/\sqrt{2}$	$1/\sqrt{2}$	1

• Same as rotating $\pi/4$ counterclockwise, and then translating over (+x) $1/\sqrt{2}$ and up (+y) $1/\sqrt{2}$

07-32: Non-Standard Axes

- We want to rotate around an axis that does not go through the origin
- 2D Case: Rotate around point at 1,0
- Create the approprate 3x3 vector

07-33: Non-Standard Axes

Rotate $\pi/4$ around (1,0)



07-37: Non-Standard Axes

- First, translate to the origin
- Then, do the rotation

• Finally, translate back

$$\left[\begin{array}{rrrr} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right]$$

07-38: Non-Standard Axes

• Final matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & -1/\sqrt{2} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & -1/\sqrt{2} & 1 \end{bmatrix}$$

07-39: Non-Standard Axes

Rotate $\pi/4$ around (1,0)



Translate to origin, Rotate $\pi/4$ around (0,0), Translate back



07-40: Non-Standard Axes

Rotate $\pi/4$ around (1,0)



Rotate p/4 around (0,0), then translate over $1 - 1 / \sqrt{2}$ and down $1 / \sqrt{2}$

07-41: Non-Standard Axes

- Note that the *rotation* component (upper right 2x2 matrix) is the same as if we were rotating around the origin
- Only the *position* component is altered.
- In general, whenever we do a rotation and a number of translations, the rotation component will be unchanged

07-42: Non-Standard Axes 3D

- To rotate in 3D around an axis whose center point does not go through the origin
 - Let $\mathbf{p} = [p_x, p_y, p_z]$ be some point on the axis of rotation
 - Let R_{3x3} be a 3x3 matrix that does the rotation, assuming the axis goes through the origin
- We can write the rotation as $TR_{4x4}T^{-1}$, where T, R_{4x4} , and T^{-1} are defined as:

07-43: Non-Standard Axes 3D

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -p_x & -p_y & -p_z & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & 0 \\ -\mathbf{p} & 1 \end{bmatrix}$$
$$\mathbf{R}_{4x4} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{3x3} & 0 \\ 0 & 1 \end{bmatrix}$$
$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p_x & p_y & p_z & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{p} & 1 \end{bmatrix}$$

07-44: Non-Standard Axes 3D

$$TRT^{-1} = \begin{bmatrix} \mathbf{I} & 0 \\ -\mathbf{p} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & 0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{p} & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{R}_{3x} & 0 \\ -\mathbf{p}\mathbf{R}_{3x3} + \mathbf{p} & 1 \end{bmatrix}$$

07-45: Non-Standard Axes 3D

- Let's take a closer look:
 - First, rotate around axis that goes through origin (this will rotate the object's position through space we want to undo this)
 - Move the object from its new (rotated) position back to the origin
 - Translate back to the original position

$$TRT^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{p} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{p} & \mathbf{1} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{R}_{3x} & \mathbf{0} \\ -\mathbf{p}\mathbf{R}_{3x3} + \mathbf{p} & \mathbf{1} \end{bmatrix}$$

07-46: Non-Standard Axes 3D

- This doesn't just work for rotating it works for any linear transform (scaling, reflecting, shearing, etc)
 - Move object to origin
 - Do the transformation
 - Move the object back

07-47: Non-Standard Axes 3D

- This doesn't just work for rotating it works for any linear transform (scaling, reflecting, shearing, etc)
 - Do the transformation, assuming axis runs through origin
 - Move the object to the origin (using transformed position)

07-48: Non-Standard Axes 3D

- This doesn't just work for rotating it works for any linear transform (scaling, reflecting, shearing, etc)
 - Do the transformation, assuming axis runs through origin
 - Move the object to the origin (using transformed position)
 - Move the object back to the original position

$$TRT^{-1} = \begin{bmatrix} \mathbf{I} & 0 \\ -\mathbf{p} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & 0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{p} & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{R}_{3x} & 0 \\ -\mathbf{p}\mathbf{R}_{3x3} + \mathbf{p} & 1 \end{bmatrix}$$

07-49: Homogenous Dimension = 0

• Consider a vector in homogenous 4-space

• [x, y, z, w]

• What happens when w = 0?

07-50: Homogenous Dimension = 0

- Consider a vector in homogenous 4-space
 - [x, y, z, w]
- What happens when w = 0?

- x, y, and z components are divided by w
- "Point at infinity"
- Direction only, not magnitude

07-51: Homogenous Dimension = 0

• What happens when multiply a vector with w = 0 by a transform that contains no translation?

$$[x, y, z, 0] \begin{bmatrix} m_{11} & m_{12} & m_{13} & 0\\ m_{21} & m_{22} & m_{23} & 0\\ m_{31} & m_{32} & m_{33} & 0\\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

 $[xm_{11}+ym_{21}+zm_{31},xm_{12}+ym_{22}+zm_{32},xm_{13}+ym_{23}+zm_{33},0]\\$

• Standard transformation – just as if w = 1

07-52: Homogenous Dimension = 0

• What happens when multiply a vector with w = 0 by a transform that does contiain translation?

$$[x, y, z, 0] \begin{bmatrix} m_{11} & m_{12} & m_{13} & 0\\ m_{21} & m_{22} & m_{23} & 0\\ m_{31} & m_{32} & m_{33} & 0\\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix} =$$

07-53: Homogenous Dimension = 0

• What happens when multiply a vector with w = 0 by a transform that does contiain translation?

$$[x, y, z, 0] \begin{bmatrix} m_{11} & m_{12} & m_{13} & 0\\ m_{21} & m_{22} & m_{23} & 0\\ m_{31} & m_{32} & m_{33} & 0\\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix} =$$

 $[xm_{11}+ym_{21}+zm_{31},xm_{12}+ym_{22}+zm_{32},xm_{13}+ym_{23}+zm_{33},0]\\$

• Rotation occurs as before - but translation is ignored

07-54: "Point at Infinity"

• If we have "point at infinity", then having the vector be affected by rotation (and non-uniform scaling, and shearing, etc.), but not translation makes sense

07-55: "Point at Infinity"



07-58: Homogenous Dimension = 0

- We can "Turn off" translation by setting w = 0
- Handy for when we want direction only, not position
 - Surface normals are an excellent example of when we want rotation to affect the vector, but not translation

07-59: Review

- We can describe the orientation of an object using a rotation matrix
 - Describes how to transform (rotate) points in the object from object space to inertial space
 - Example: Rotate 45 degrees around the Z-axis

$$\begin{bmatrix} \cos \pi/4 & \sin \pi/4 & 0 \\ -\sin \pi/4 & \cos \pi/4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

07-60: Review

- We have a point at position $[x_1, y_1, z_1]$ in object space (That's how points in the mesh are stored)
- We need to know the position of the point in world space before rendering (assume no translation yet our model is at the origin)
- We can do a simple multiply:

$$[x_1, y_1, z_1] \begin{bmatrix} \cos \pi/4 & \sin \pi/4 & 0 \\ -\sin \pi/4 & \cos \pi/4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

07-61: Review

- So, our rotation matrix gives us a way to transform points from object space into world space
- Rotation matrix also tells us where our object is facing in world space, and what the up vector of our object is in world space
- How?

07-62: **Review**

- So, our rotation matrix gives us a way to transform points from object space into world space
- Rotation matrix also tells us where our object is facing in world space, and what the up vector of our object is in world space
 - We know the direction our object is facing in local space: [0, 0, 1]
 - If we transfrom this by a matrix, what do we get?

07-63: **Review**

- So, our rotation matrix gives us a way to transform points from object space into world space
- Rotation matrix also tells us where our object is facing in world space, and what the up vector of our object is in world space

- We know the direction our object is facing in local space: [0, 0, 1]
- If we transfrom this by a matrix we get the bottom row of the matrix

07-64: **Review**

- Of course, our objects are not always at the origin
- In addition to the rotational matrix, we also have a position location of the center of the model
- Now, to transform a point, we first rotate it, and then translate it
 - Rotation matrix for our model: M_R
 - Position of our object (displacement from the origin): $\mathbf{pos} = [x_m, y_m, z_m]$
- How can we transform a point [x, y, z] in the object space of this model into world space?

07-65: Review

- Rotation matrix for our model: M_R
- Position of our object (displacement from the origin): $\mathbf{pos} = [x_m, y_m, z_m]$
 - How can we transform a point $\mathbf{p}_{\mathbf{O}} = [x, y, z]$ in the object space of this model into world space?

World space $\mathbf{p}_{\mathbf{w}} = \mathbf{p}_{\mathbf{O}} \mathbf{M}_{\mathbf{R}} + \mathbf{pos}$ 07-66: **Review**

• As a mathematical trick, we can combine our rotation matrix and position into a single entity

m_{11}	m_{12}	m_{13}	0]
m_{21}	m_{22}	m_{23}	0
m_{31}	m_{32}	m_{33}	0
x	y	z	1

• Now, to transform a point, convert it to a 4-element vector (by adding a 1 at the end), multiply by this matrix, look at first 3 elements of the vector

07-67: **Review**

- A 4x4 matrix represents a rotation, followed by a translation
- We can combine multiple transformations by multiplying matrices together
- Result is a single matrix, which represents a single rotation, followed by a single translation.

07-68: Review

- Example: Finding the end of a tank barrel
 - Tank has a location and rotation in world space (represented by a position vector and 3x3 rotation matrix)
 - Barrel has a location and rotation (represented by a position vector and 3x3 rotation matrix reltaive to the center of the tank
 - End of the tank barrel is at location [0, 0, 3] in barrel space
- What is the location of the end of the tank barrel in world space? (do both 3x3 matrices & positions, and 4x4 matrices)

07-69: Review

- Given:
 - A bullet position in world space $\mathbf{p}_b = [b_x, b_y, b_z]$
 - A bullet position in world space $\mathbf{v}_b = [bv_x, bv_y, bv_z]$
 - A rotation matrix for a tank \mathbf{M}_T , and a position for a tank \mathbf{p}_T
 - What is the position and velocity of the bullet in tank space?
 - Why might that be a useful thing to have?

07-70: Row vs. Column Vectors

- Row Vectors
 - Rows of the matrix represent transform of object (1st row is x, 2nd row is y, 3rd row is z)
 - To transform a vector v by first A, then B, then C: vABC
- Column Vectors
 - Columns of the matrix represent transform of object (1st col is x, 2nd col is y, 3rd col is z)
 - To transform a vector **v** by first **A**, then **B**, then **C**: **CBAv**

07-71: Row vs. Column Vectors

• 4x4 Matrx using Row vectors:

• 4x4 Matrx using Column vectors:

x_1 y_1 z_1 Δx	x_2 y_2 z_2 Δy	x_3 y_3 z_3 Δz	$ \begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \end{array} $]
	$\substack{y_1\\y_2\\y_3\\0}$	$\begin{smallmatrix}z_1\\z_2\\z_3\\0\end{smallmatrix}$	$\begin{array}{c} \Delta x \\ \Delta y \\ \Delta z \\ 1 \end{array}$	

07-72: Row vs. Column Vectors

- Ogre & OpenGL use column vectors
- Direct3D uses row vectors
- How does Ogre do both?
 - Does everything in column vectors
 - Multiplies matrices together using column vector convetion
 - When it's time to send a matrix to D3D, does a quick transpose first

07-73: Rotational Matrix Trick

- To remember how to create rotational matrices for the cardinal axes, you just need to remember: cos, sin, -sin, cos
 - If you forget, do the 2D case
- Create 3x3 rotational matrix with the non-rotating vector in the correct location
- From the one in the non-rotating vector, go down and right, and fill in cos,sin,-sin,cos
 - Wrap around as necessary
 - Examples (column major and row major)