GAME2016
Mathematical Foundation of Game Design and Animation

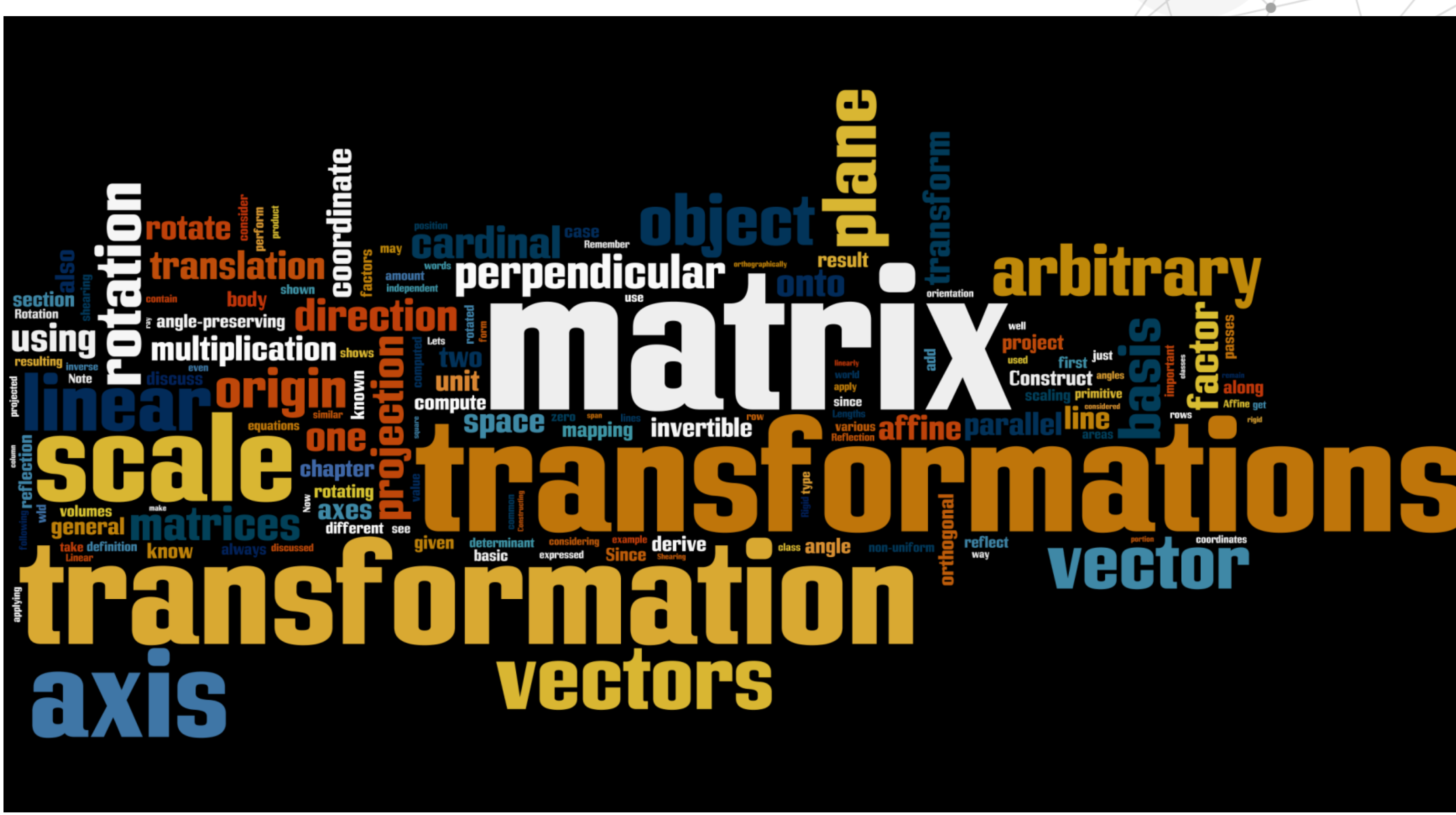
Lecture 4
Matrices and Linear Transformations

Dr. Paolo Mengoni
pmengoni@hkbu.edu.hk
Senior Lecturer @HKBU Department of Interactive Media

Agenda

- Introduction to matrices for primitive linear transformations
  - Rotation
  - Scaling
  - Orthographic projection
  - Reflection
  - Shearing
- Complex transformations and matrix multiplication

Matrices and Linear Transformations

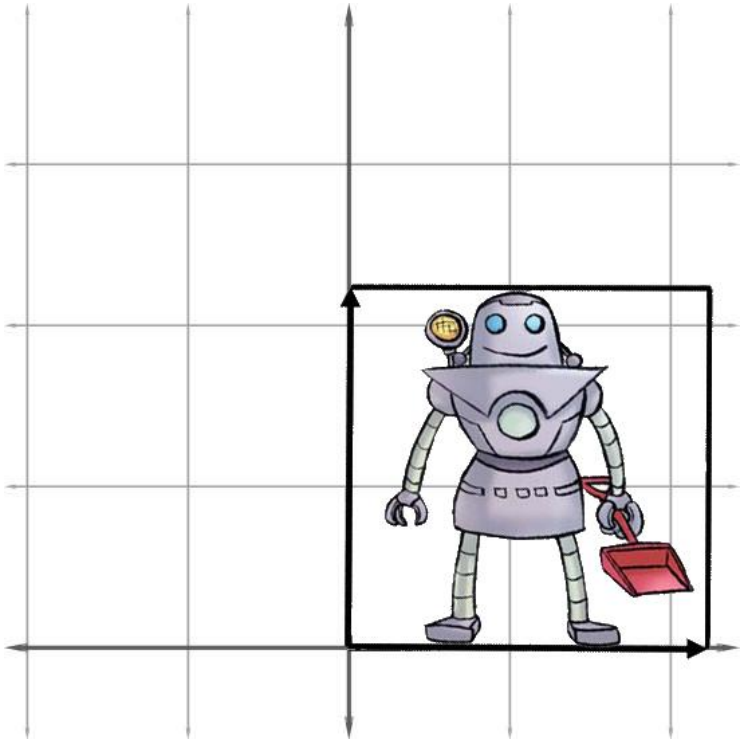

matrix
transformations
transformation
scale
axis
vectors
vector
arbitrary
object
plane
transform
cardinal
perpendicular
rotation
rotate
translation
coordinate
direction
multiplication
origin
linear
projection
basis
factor
affine
parallel
line
space
mapping
invertible
compute
unit
two
one
chapter
rotating
axes
matrices
general
volumes
reflection
using
section
also
body
shown
angle-preserving
shows
resulting
Note
discuss
equations
different
see
given
determinant
basic
considering
expressed
example
Since
derive
class
angle
non-uniform
orthogonal
reflect
way
coordinates
take
definition
know
always
discussed
project
first
just
Construct
primitive
scaling
along
Affine
rows
result
onto
orientation
well
since
various
Reflection
apply
add
case
Remember
may
amount
independent
words
use
Rotation

Rotation

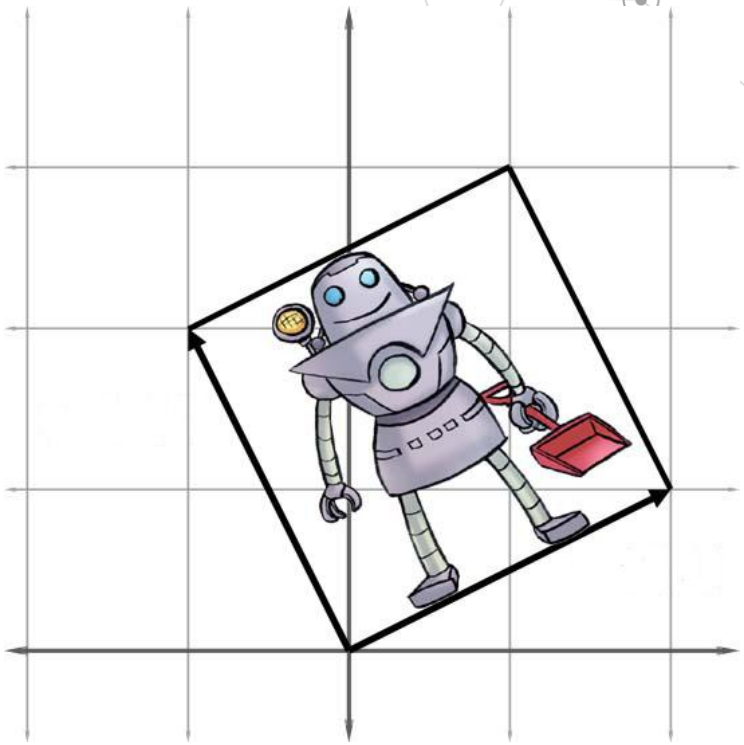
Reminder: Visualize The Matrix

- Each row of a matrix is a *basis vector* after transformation.
- Given an arbitrary matrix, visualize the transformation by its effect on the standard *basis vectors* – the rows of the matrix.
- Given an arbitrary linear transformation, create the matrix by visualizing what it does to the standard *basis vectors* and using that for the rows of the matrix.

2D Rotation Around Point

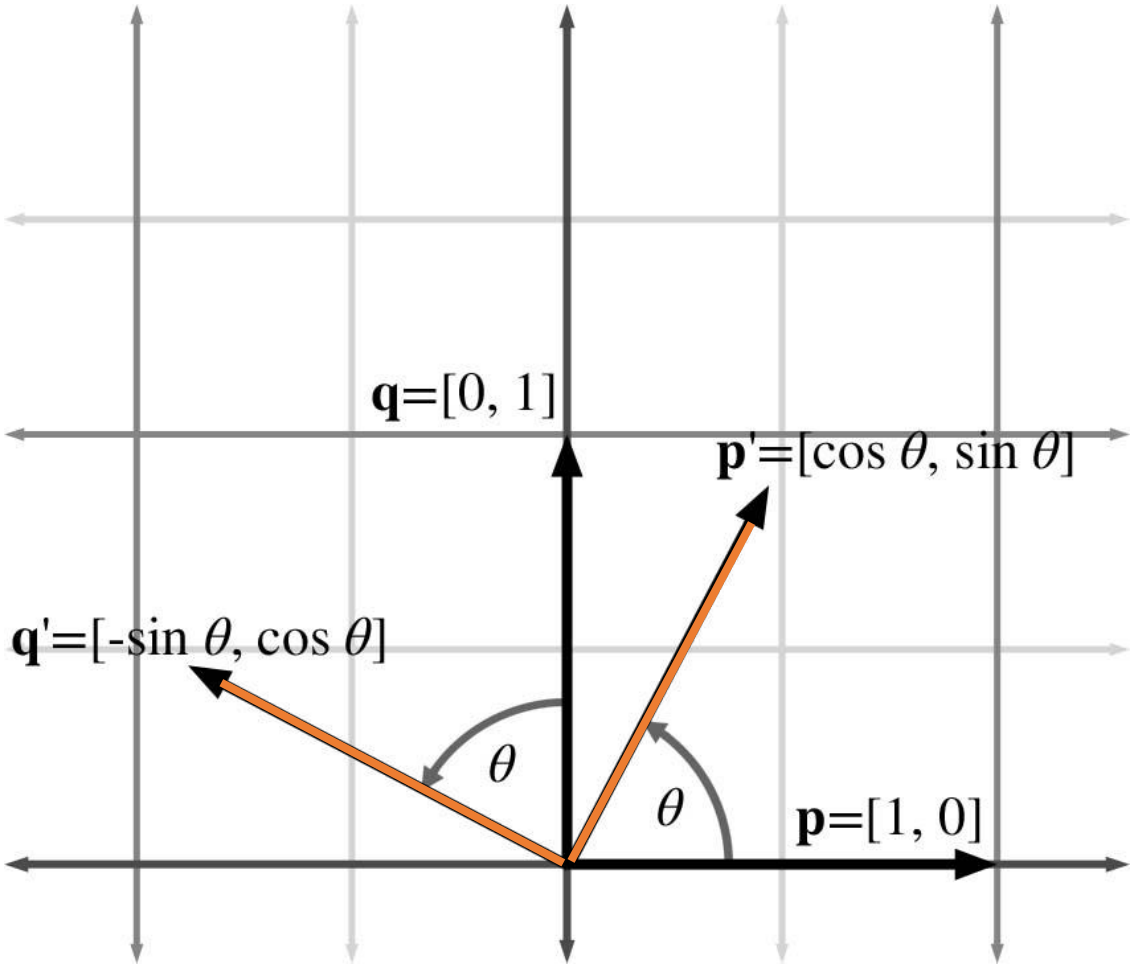


Before



After

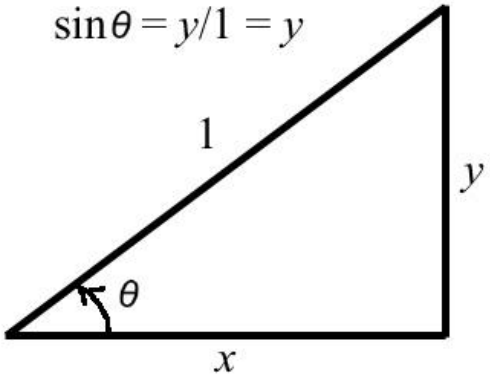
It's All About Rotating Basis Vectors!



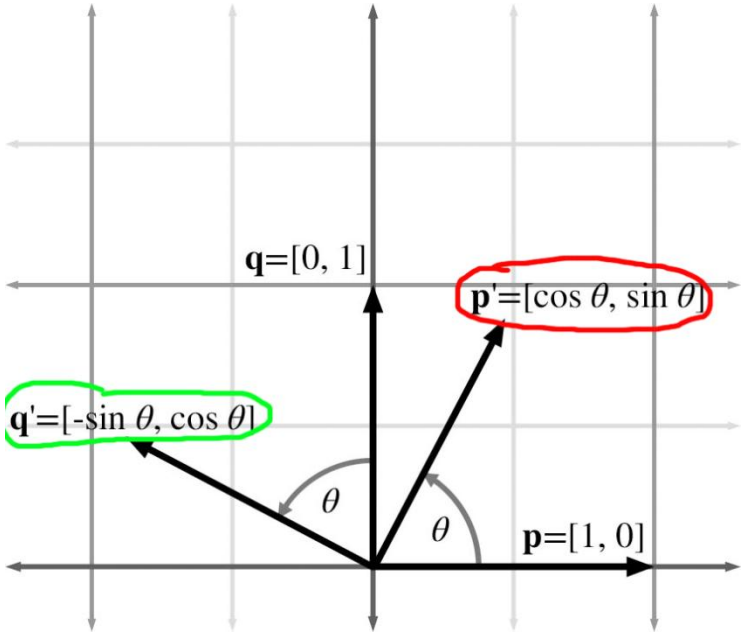
"SohCahToa"

$\cos\theta = x/1 = x$

$\sin\theta = y/1 = y$



Construct Matrix from Basis Vectors



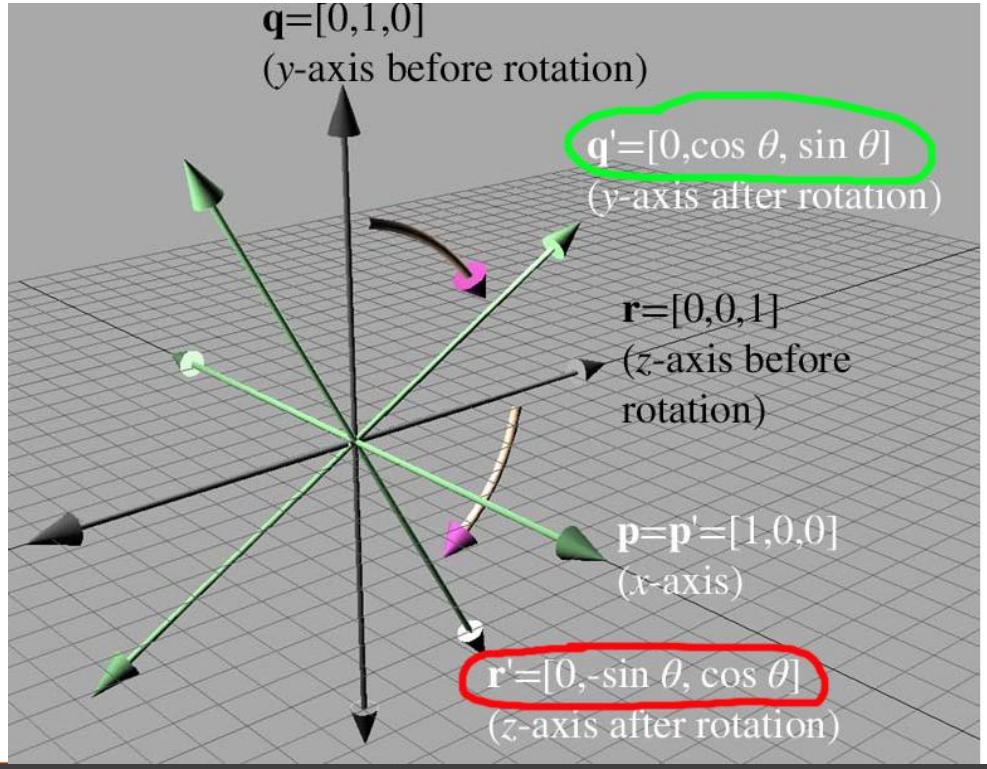
$$\mathbf{R}(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

3D Rotation About Cardinal Axis

- In 3D, rotation occurs about an axis rather than a point as in 2D.
- The most common type of rotation is a simple rotation about one of the cardinal axes.
- We'll need to establish which direction of rotation is “positive” and which is “negative.”
  - Use the left-hand rule

3D Rotate About *x*-axis

- Rotating about the *x*-axis means the *x*-axis doesn't get transformed!
  - The **p** basis vector [1, 0, 0] does not change
  - The *x* component for the **q** and **r** basis vectors doesn't change



$$\mathbf{R}_x(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \\ -\mathbf{r}'- \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

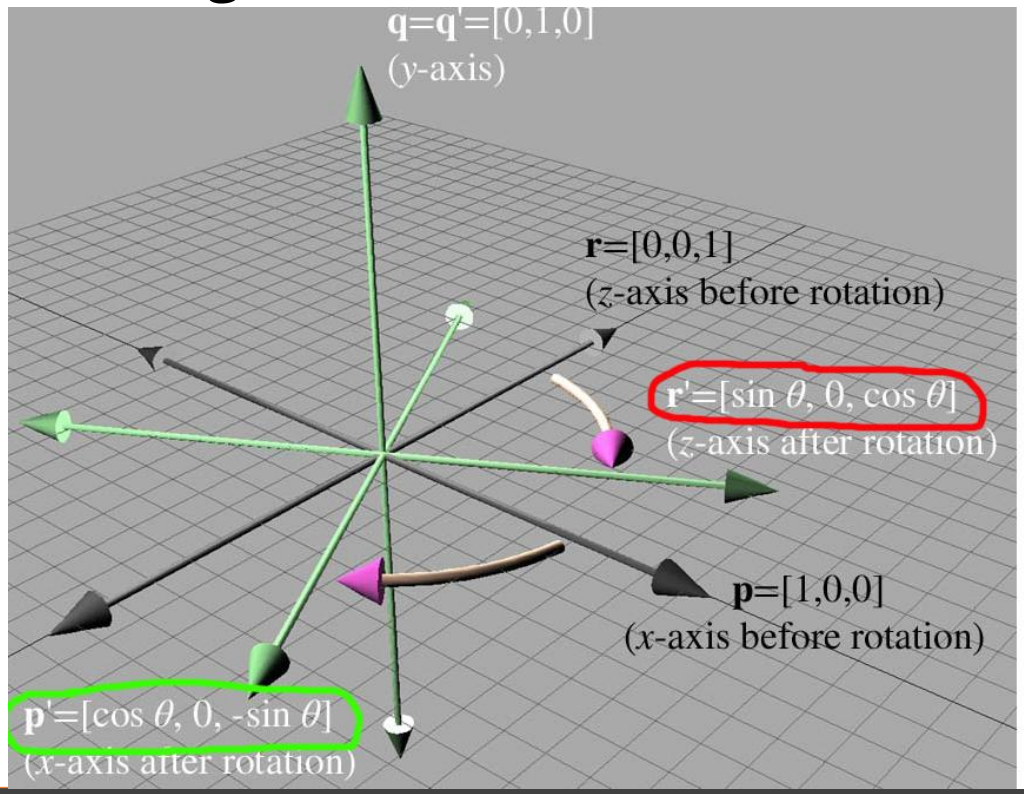
Compare to 2D Case

$$\mathbf{R}(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_x(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \\ -\mathbf{r}'- \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

3D Rotate About *y*-axis

- Rotating about the *y*-axis means the *y*-axis doesn't get transformed!
  - The **q** basis vector [0, 1, 0] does not change
  - The *y* component for the **p** and **r** basis vectors doesn't change



$$\mathbf{R}_y(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \\ -\mathbf{r}'- \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

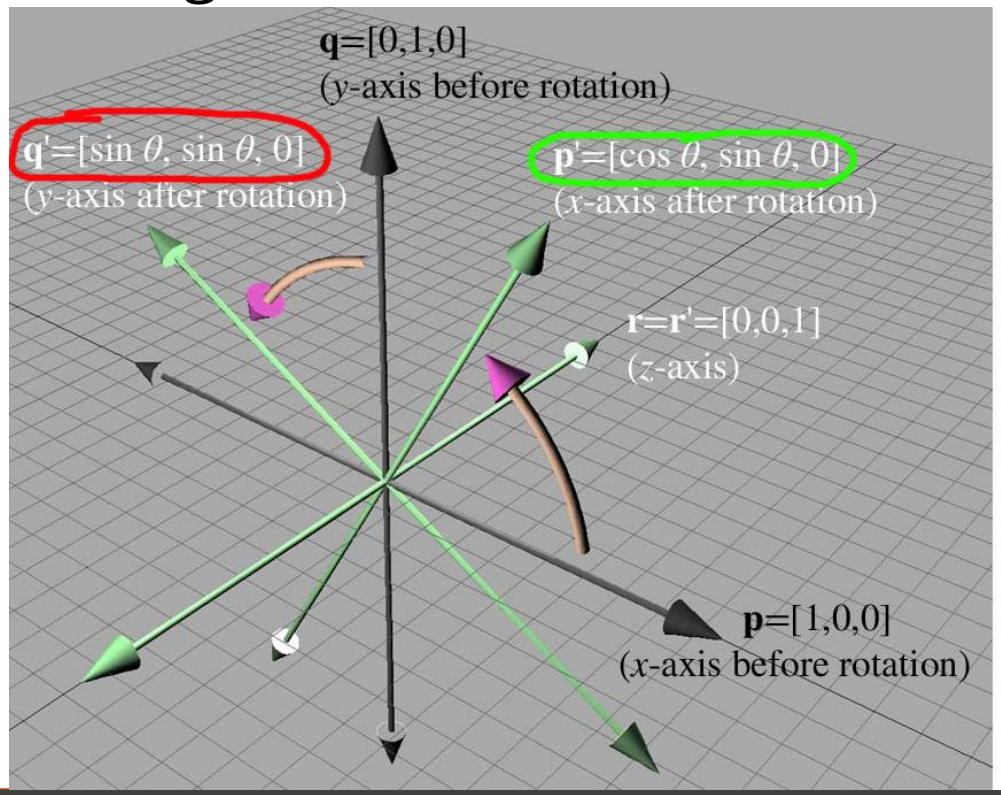
Compare to 2D Case

$$\mathbf{R}(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \\ -\mathbf{r}'- \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

3D Rotate About *z*-axis

- Rotating about the *z*-axis means the *z*-axis doesn't get transformed!
  - The **r** basis vector [0, 0, 1] does not change
  - The *z* component for the **p** and **q** basis vectors doesn't change



$$\mathbf{R}_z(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \\ -\mathbf{r}'- \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compare to 2D Case

$$\mathbf{R}(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

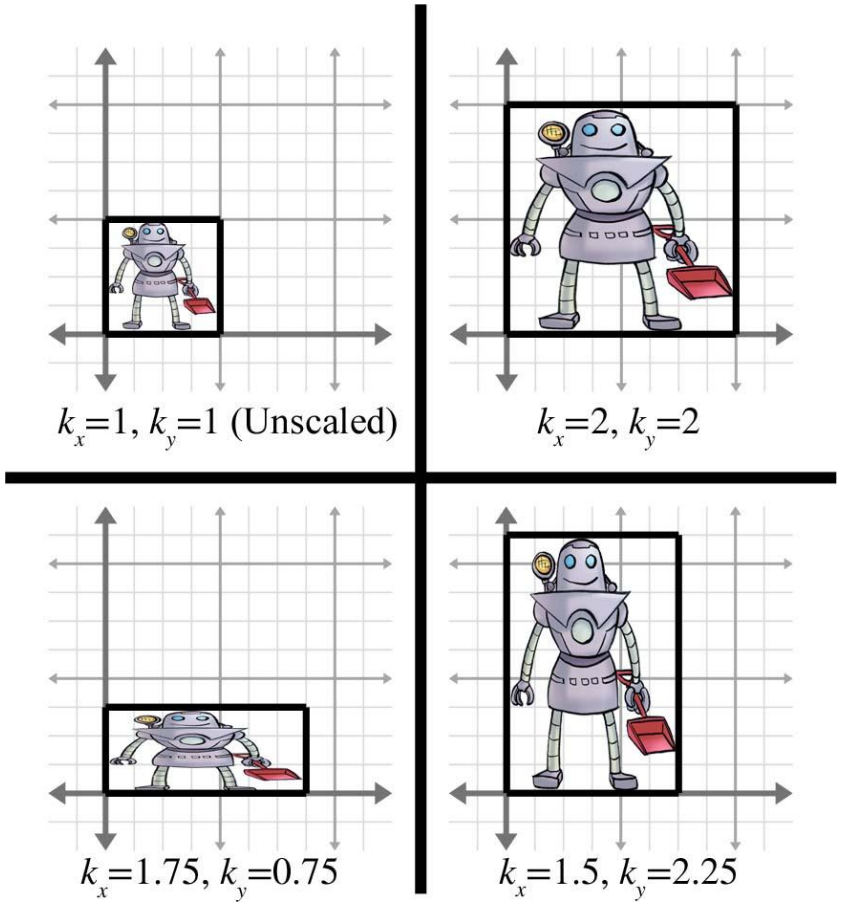
$$\mathbf{R}_z(\theta) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \\ -\mathbf{r}'- \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3D Rotation About Noncardinal Axis

- We can also rotate about an arbitrary axis that passes through the origin.
- This is more complicated and less common than rotating about a cardinal axis.
- Rotation about an arbitrary axis can be expressed as a sum of rotations about cardinal axes, using Euler theorem
  - Details are in the textbook

Scale

Scaling Along Cardinal Axes in 2D



Basis Vectors for Scale

- The basis vectors **p** and **q** are independently affected by the corresponding scale factors:

$$\begin{aligned} \mathbf{p}' &= k_x\mathbf{p} = k_x\begin{bmatrix}1 & 0\end{bmatrix} = \begin{bmatrix}k_x & 0\end{bmatrix} \\ \mathbf{q}' &= k_y\mathbf{q} = k_y\begin{bmatrix}0 & 1\end{bmatrix} = \begin{bmatrix}0 & k_y\end{bmatrix} \end{aligned}$$

2D Scale Matrix

- Constructing the 2D scale matrix $\mathbf{S}(k_x, k_y)$ from these basis vectors:

$$\mathbf{S}(k_x, k_y) = \begin{bmatrix} -\mathbf{p}'- \\ -\mathbf{q}'- \end{bmatrix} = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}$$

3D Scale Matrix

$$\mathbf{S}(k_x, k_y, k_z) = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix}$$

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} = \begin{bmatrix} k_x x & k_y y & k_z z \end{bmatrix}$$

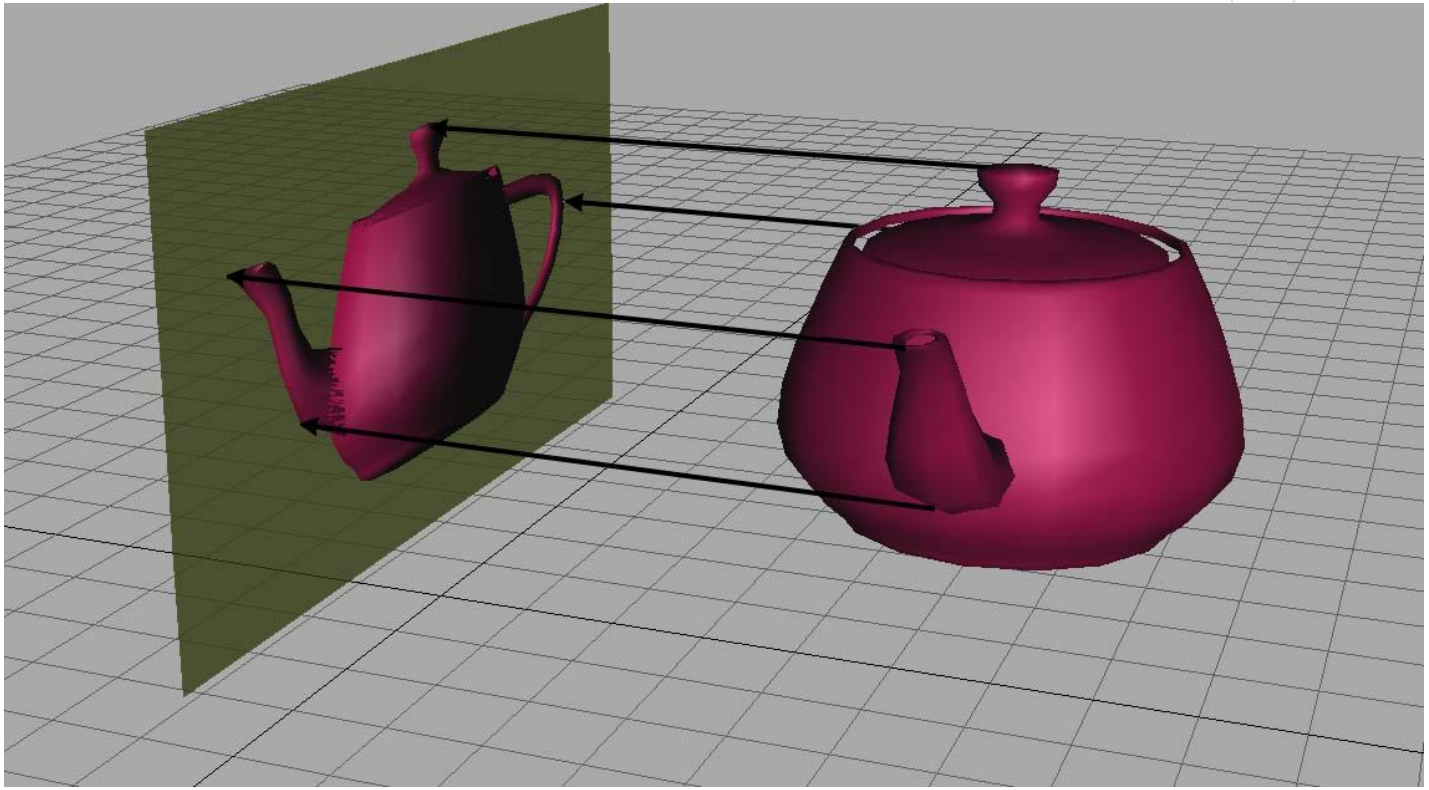
Scale in Arbitrary Direction

- The math for scaling in an arbitrary direction is intricate, but not too tricky.

- For game programmers it is not used very often.
  - Details are in the textbook.

Orthographic Projection

Orthographic Projection



Projecting Onto a Cardinal Axis

- Projection onto a cardinal axis or plane most frequently occurs not by actual transformation, but by simply discarding one of the dimensions while assigning the data into a variable of lesser dimension.

- For example, we may turn a 3D object into a 2D object by discarding the *z* components of the points and copying only *x* and *y*.

- However, we can also project onto a cardinal axis or plane by using a scale value of zero on the perpendicular axis.

Projection Matrices

- For completeness, we present the matrices for these transformations:

$$\mathbf{P}_x = \mathbf{S}\left(\begin{bmatrix}0 & 1\end{bmatrix}, 0\right) = \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}$$

$$\mathbf{P}_y = \mathbf{S}\left(\begin{bmatrix}1 & 0\end{bmatrix}, 0\right) = \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}$$

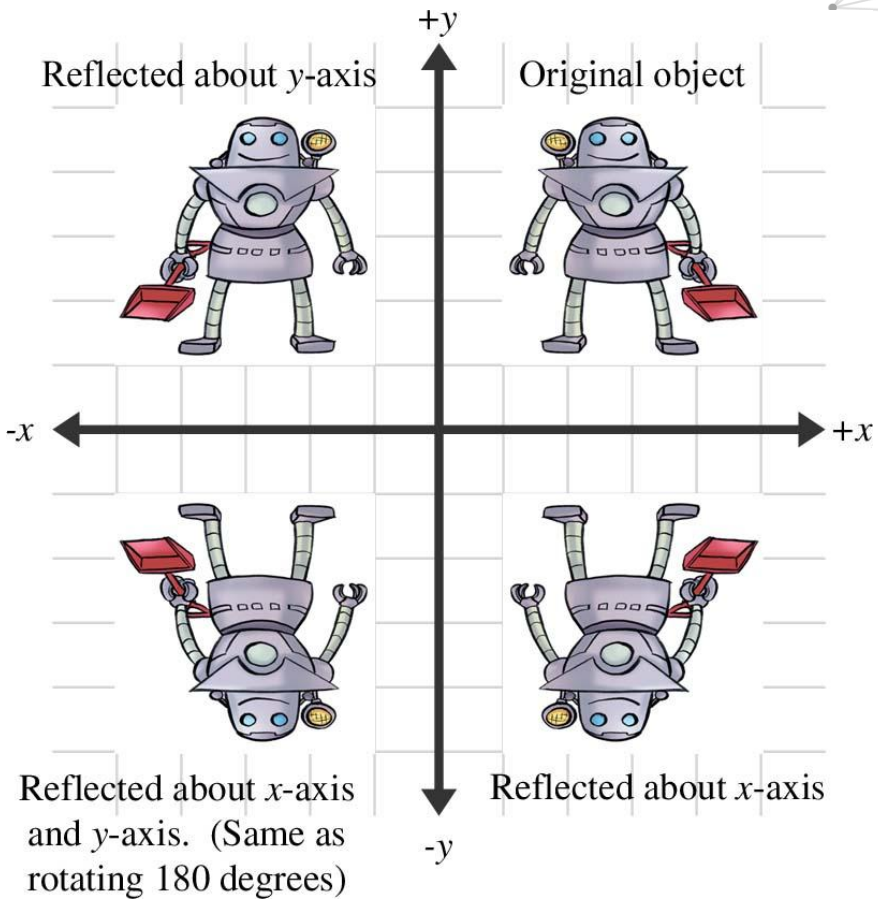
$$\mathbf{P}_{xy} = \mathbf{S}\left(\begin{bmatrix}0 & 0 & 1\end{bmatrix}, 0\right) = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$\mathbf{P}_{xz} = \mathbf{S}\left(\begin{bmatrix}0 & 1 & 0\end{bmatrix}, 0\right) = \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$\mathbf{P}_{yz} = \mathbf{S}\left(\begin{bmatrix}1 & 0 & 0\end{bmatrix}, 0\right) = \begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

Reflection

Reflection


+y
Reflected about y-axis
Original object
-x
+x
Reflected about x-axis and y-axis. (Same as rotating 180 degrees)
Reflected about x-axis
-y

Reflection in 2D

- Reflection can be accomplished by applying a scale factor of -1.
- Let **n** be a 2D unit vector. The following matrix performs a reflection about the axis through the origin perpendicular to **n**:

$$\mathbf{R}(\mathbf{n}) = \mathbf{S}(\mathbf{n}, -1) = \begin{bmatrix} 1 + (-1-1)\,{n_x}^2 & (-1-1)\,n_x n_y \\ (-1-1)\,n_x n_y & 1 + (-1-1)\,{n_y}^2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 - 2{n_x}^2 & -2n_x n_y \\ -2n_x n_y & 1 - 2{n_y}^2 \end{bmatrix}$$

- These matrix is derived from scaling in an arbitrary direction formula
  - Details in the textbook

Reflection in 3D

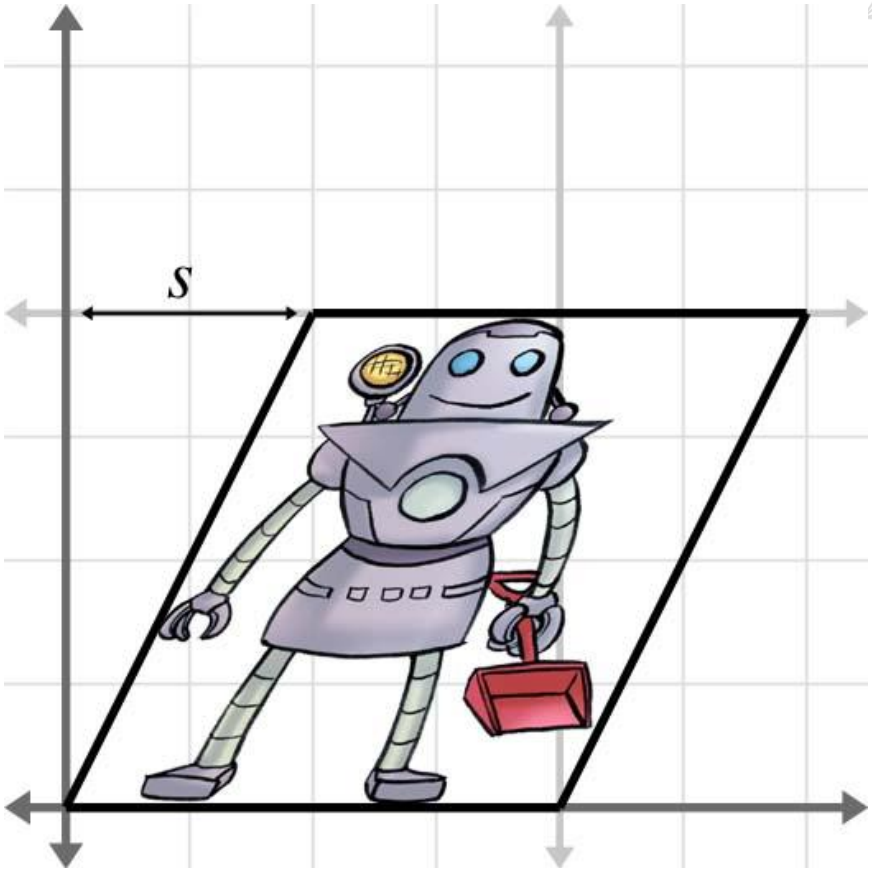
- In 3D, we have a reflecting *plane* instead of an axis
- The following matrix reflects about a plane through the origin perpendicular to the unit vector **n**:

$$\mathbf{R}(\mathbf{n}) = \mathbf{S}(\mathbf{n}, -1) = \begin{bmatrix} 1+(-1-1)\,{n_x}^2 & (-1-1)\,n_x n_y & (-1-1)\,n_x n_z \\ (-1-1)\,n_x n_y & 1+(-1-1)\,{n_y}^2 & (-1-1)\,n_y n_z \\ (-1-1)\,n_x n_z & (-1-1)\,n_y n_z & 1+(-1-1)\,{n_z}^2 \end{bmatrix}$$

$$= \begin{bmatrix} 1-2{n_x}^2 & -2n_x n_y & -2n_x n_z \\ -2n_x n_y & 1-2{n_y}^2 & -2n_y n_z \\ -2n_x n_z & -2n_y n_z & 1-2{n_z}^2 \end{bmatrix}$$

Shearing

Shearing



Shearing in 2D

- Shearing is a transformation that skews the coordinate space, stretching it non-uniformly.
- Angles are not preserved; however, surprisingly, areas and volumes are.
- The basic idea is to add a multiple of one coordinate to the other.
- For example, in 2D, we might take a multiple of y and add it to x, so that $x' = x + sy$.

2D Shear Matrices

- Let $\mathbf{H}_x(s)$ be the shear matrix that shears the x coordinate by the other coordinate, y, by amount s.

$$\mathbf{H}_x(s) = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix},$$

$$\mathbf{H}_y(s) = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}.$$

3D Shear Matrices

- In 3D, we can take one coordinate and add different multiples of that coordinate to the other two coordinates.

- The notation H_{xy} indicates that the *x* and *y* coordinates are shifted by the other coordinate, *z*.

$$\mathbf{H}_{xy}(s,t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ s & t & 1 \end{bmatrix}$$

$$\mathbf{H}_{xz}(s,t) = \begin{bmatrix} 1 & 0 & 0 \\ s & 1 & t \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_{yz}(s,t) = \begin{bmatrix} 1 & s & t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Combining Transformations

Combining Transforms

- Transformation matrices are combined using matrix multiplication.
- One very common example of this is in rendering. Imagine there is an object at an arbitrary position and orientation in the world.
- We wish to render this object given a camera in any position and orientation.
  - Assume we are rendering some sort of triangle mesh

Combining Transforms (cont.)

- To do this, we must take the vertices of the object and transform them from object space into world space.

- This transform is known as the *model transform*, which we'll denote $\mathbf{M}_{obj \rightarrow wld}$

- From there, we transform world-space vertices using the *view transform*, denoted $\mathbf{M}_{wld \rightarrow cam}$ into camera space.

Render Matrix Example

- Math involved

$$\mathbf{p}_{\text{wld}} = \mathbf{p}_{\text{obj}}\,\mathbf{M}_{\text{obj}\to\text{wld}}$$

$$\begin{aligned}\mathbf{p}_{\text{cam}} &= \mathbf{p}_{\text{wld}}\,\mathbf{M}_{\text{wld}\to\text{cam}} \\ &= (\mathbf{p}_{\text{obj}}\,\mathbf{M}_{\text{obj}\to\text{wld}})\,\mathbf{M}_{\text{wld}\to\text{cam}}\end{aligned}$$

- Since matrix multiplication is associative,

$$\begin{aligned}\mathbf{p}_{\text{cam}} &= (\mathbf{p}_{\text{obj}}\,\mathbf{M}_{\text{obj}\to\text{wld}})\,\mathbf{M}_{\text{wld}\to\text{cam}} \\ &= \mathbf{p}_{\text{obj}}(\mathbf{M}_{\text{obj}\to\text{wld}}\,\mathbf{M}_{\text{wld}\to\text{cam}})\end{aligned}$$

- For efficiency, we can concatenate the matrices outside the loop, and only have one matrix multiplication inside the loop (remember there are many vertices):

$$\mathbf{M}_{\text{obj}\to\text{cam}} = \mathbf{M}_{\text{obj}\to\text{wld}}\,\mathbf{M}_{\text{wld}\to\text{cam}}$$

$$\mathbf{p}_{\text{cam}} = \mathbf{p}_{\text{obj}}\,\mathbf{M}_{\text{obj}\to\text{cam}}$$

Geometric Interpretation

- Matrix concatenation works from an algebraic perspective using the associative property of matrix multiplication.
- Let's see if we can't get a more geometric interpretation.
- The rows of a matrix contain the basis vectors after transformation.
  - This is true even in the case of multiple transformations.
- Notice that in the matrix product **AB**, each resulting row is the product of the corresponding row from **A** times the matrix **B**.

Geometric Interpretation

- Let the row vectors $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$ stand for the rows of **A**.

- Then matrix multiplication can alternatively be written like this…

$$\mathbf{A} = \begin{bmatrix} -\mathbf{a}_1- \\ -\mathbf{a}_2- \\ -\mathbf{a}_3- \end{bmatrix} \qquad \mathbf{AB} = \left(\begin{bmatrix} -\mathbf{a}_1- \\ -\mathbf{a}_2- \\ -\mathbf{a}_3- \end{bmatrix} \mathbf{B} \right) = \begin{bmatrix} -\mathbf{a}_1\mathbf{B}- \\ -\mathbf{a}_2\mathbf{B}- \\ -\mathbf{a}_3\mathbf{B}- \end{bmatrix}$$

- This makes it explicitly clear that the rows of the product of **AB** are actually the result of transforming the basis vectors in **A** by **B**.