Chapter 4: Introduction to Matrices

Fletcher Dunn
Valve Software

Ian Parberry
University of North Texas

What You’ll See in This Chapter

This chapter introduces matrices. It is divided into three sections.
• Section 4.1 discusses some of the basic properties and operations of matrices strictly from a mathematical perspective.
• Section 4.2 explains how to interpret these properties and operations geometrically.
• Section 4.3 puts the use of matrices in this book in context within the larger field of linear algebra, and is not covered in these notes.

Section 4.1: Matrix: An Algebraic Definition

Definitions

• Algebraic definition of a matrix: a table of scalars in square brackets.
• Matrix dimension is the width and height of the table, \( w \times h \).
• Typically we use dimensions 2 x 2 for 2D work, and 3 x 3 for 3D work.
• We’ll find a use for 4 x 4 matrices also. It’s a kluge. More later.

Matrix Components

• Entries are numbered by row and column, eg. \( m_{ij} \) is the entry in row \( i \), column \( j \).
• Start numbering at 1, not 0.

\[
\begin{bmatrix}
m_{11} & m_{12} & m_{13} \\
m_{21} & m_{22} & m_{23} \\
m_{31} & m_{32} & m_{33}
\end{bmatrix}
\]
Square Matrices

- Same number as rows as columns.
- Entries $m_{ij}$ are called the **diagonal** entries. The others are called **nondiagonal** entries.

![Square Matrix Example]

Diagonal Matrices

A diagonal matrix is a square matrix whose nondiagonal elements are zero.

$$
\begin{bmatrix}
3 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & -5 & 0 \\
0 & 0 & 0 & 2
\end{bmatrix}
$$

The Identity Matrix

The identity matrix of dimension $n$, denoted $I_{n}$, is the $n \times n$ matrix with 1s on the diagonal and 0s elsewhere.

$$I_3 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
$$

Vectors as Matrices

- A row vector is a $1 \times n$ matrix.
- A column vector is an $n \times 1$ matrix.
- They were pretty much interchangeable in the lecture on Vectors.
- They’re not once you start treating them as matrices.

$$
\begin{bmatrix}
1 & 2 & 3
\end{bmatrix}
\quad
\begin{bmatrix}
4 \\
5 \\
6
\end{bmatrix}
$$

Transpose of a Matrix

- The transpose of an $r \times c$ matrix $M$ is a $c \times r$ matrix called $M^T$.
- Take every row and rewrite it as a column.
- Equivalently, flip about the diagonal.

$$
\begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i
\end{bmatrix}^T =
\begin{bmatrix}
a & d & g \\
b & e & h \\
c & f & i
\end{bmatrix}
$$

Facts About Transpose

- Transpose is its own inverse: $(M^T)^T = M$ for all matrices $M$.
- $D^T = D$ for all diagonal matrices $D$ (including the identity matrix $I$).
Transpose of a Vector

If \( \mathbf{v} \) is a row vector, \( \mathbf{v}^T \) is a column vector and vice-versa

\[
\begin{bmatrix}
x \\ y \\ z
\end{bmatrix}^T = \begin{bmatrix} x \\ y \\ z \end{bmatrix}
\]

Multiplying By a Scalar

- Can multiply a matrix by a scalar.
- Result is a matrix of the same dimension.
- To multiply a matrix by a scalar, multiply each component by the scalar.

\[
k \mathbf{M} = k \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \\ m_{41} & m_{42} & m_{43} \end{bmatrix} = \begin{bmatrix} km_{11} & km_{12} & km_{13} \\ km_{21} & km_{22} & km_{23} \\ km_{31} & km_{32} & km_{33} \\ km_{41} & km_{42} & km_{43} \end{bmatrix}
\]

Matrix Multiplication

Multiplying an \( r \times n \) matrix \( \mathbf{A} \) by an \( n \times c \) matrix \( \mathbf{B} \) gives an \( r \times c \) result \( \mathbf{AB} \).

\[
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rn}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} & \cdots & b_{1c} \\ b_{21} & b_{22} & \cdots & b_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nc}
\end{bmatrix} =
\begin{bmatrix}
c_{11} & c_{12} & \cdots & c_{1c} \\ c_{21} & c_{22} & \cdots & c_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ c_{r1} & c_{r2} & \cdots & c_{rc}
\end{bmatrix}
\]

Multiplication: Result

- Multiply an \( r \times n \) matrix \( \mathbf{A} \) by an \( n \times c \) matrix \( \mathbf{B} \) to give an \( r \times c \) result \( \mathbf{C} = \mathbf{AB} \).
- Then \( \mathbf{C} = [c_{ij}] \), where \( c_{ij} \) is the dot product of the \( i \)th row of \( \mathbf{A} \) with the \( j \)th column of \( \mathbf{B} \).
- That is:

\[
c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.
\]

Example

Another Way of Looking at It

\[
\begin{bmatrix}
a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45}
\end{bmatrix} =
\begin{bmatrix}
c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45}
\end{bmatrix}
\]

\[
c_{41} = a_{41} b_{13} + a_{42} b_{23}
\]
2 x 2 Case

\[
AB = \begin{bmatrix}
    a_{11} & a_{12} \\
    a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
    b_{11} & b_{12} \\
    b_{21} & b_{22}
\end{bmatrix}
= \begin{bmatrix}
    a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\
    a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22}
\end{bmatrix}
\]

2 x 2 Example

\[
A = \begin{bmatrix}
    -3 & 0 \\
    5 & 1/2
\end{bmatrix}, \quad B = \begin{bmatrix}
    -7 & 2 \\
    4 & 6
\end{bmatrix}
\]

\[
AB = \begin{bmatrix}
    -3 & 0 \\
    5 & 1/2
\end{bmatrix}
\begin{bmatrix}
    -7 & 2 \\
    4 & 6
\end{bmatrix}
= \begin{bmatrix}
    (-3)(-7) + (0)(4) & (-3)(2) + (0)(6) \\
    (5)(-7) + (1/2)(4) & (5)(2) + (1/2)(6)
\end{bmatrix}
= \begin{bmatrix}
    21 & -6 \\
    -33 & 13
\end{bmatrix}
\]

3 x 3 Case

\[
AB = \begin{bmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & a_{33}
\end{bmatrix}
\begin{bmatrix}
    b_{11} & b_{12} & b_{13} \\
    b_{21} & b_{22} & b_{23} \\
    b_{31} & b_{32} & b_{33}
\end{bmatrix}
= \begin{bmatrix}
    a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\
    a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\
    a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33}
\end{bmatrix}
\]

3 x 3 Example

\[
A = \begin{bmatrix}
    1 & -5 & 3 \\
    0 & 2 & 6 \\
    7 & 2 & -4
\end{bmatrix}, \quad B = \begin{bmatrix}
    -8 & 6 & 1 \\
    7 & 0 & -3 \\
    2 & 4 & 5
\end{bmatrix}
\]

\[
AB = \begin{bmatrix}
    1 & -5 & 3 \\
    0 & 2 & 6 \\
    7 & 2 & -4
\end{bmatrix}
\begin{bmatrix}
    -8 & 6 & 1 \\
    7 & 0 & -3 \\
    2 & 4 & 5
\end{bmatrix}
= \begin{bmatrix}
    1(-8) - (-5)7 + 32 & 16 + (-5)0 + 3-4 & 11 + (-5)(-3) + 3-5 \\
    0(-8) + (-2)7 + 6-2 & 06 + (-2)0 + 64 & 01 + (-2)(-3) + 65 \\
    7(-8) + 27 + (-4)2 & 76 + 20 + (-4)4 & 7-1 + (2)(-3) + (4)5
\end{bmatrix}
= \begin{bmatrix}
    -37 & 18 & 31 \\
    -2 & 24 & 36 \\
    -50 & 26 & -19
\end{bmatrix}
\]

Identity Matrix

• Recall that the identity matrix I (or I_n) is a diagonal matrix whose diagonal entries are all 1.
• Now that we’ve seen the definition of matrix multiplication, we can say that IM = MI = M for all matrices M (dimensions appropriate)

\[
I_3 = \begin{bmatrix}
    1 & 0 & 0 \\
    0 & 1 & 0 \\
    0 & 0 & 1
\end{bmatrix}
\]

Matrix Multiplication Facts

• Not commutative: in general AB ≠ BA.
• Associative:
  \[(AB)C = A(BC)\]
• Associates with scalar multiplication:
  \[k(AB) = (kA)B = A(kB)\]
• \[(AB)^T = B^TA^T\]
• \[(M_1M_2M_3...M_n)^T = M_n^TM_3^TM_2^TM_1^T\]
Row Vector Times Matrix Multiplication

Can multiply a row vector times a matrix

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

Matrix Times Column Vector Multiplication

Can multiply a matrix times a column vector.

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

Row vs. Column Vectors

• Row vs. column vector matters now. Here’s why: Let \( v \) be a row vector, \( M \) a matrix.
  – \( vM \) is legal, \( Mv \) is undefined
  – \( v^T \) is legal, \( v^T M \) is undefined
• DirectX uses row vectors.
• OpenGL uses column vectors.

Common Mistake

\( \mathbf{M}v^T \neq (\mathbf{vM})^T \), but \( \mathbf{M}v^T = (\mathbf{vM})^T \) – compare the following two results:

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

Vector-Matrix Multiplication Facts 1

Associates with vector multiplication.
  • Let \( v \) be a row vector:
    \( v(\mathbf{AB}) = (v\mathbf{A})\mathbf{B} \)
  • Let \( v \) be a column vector:
    \( (\mathbf{AB})v = \mathbf{A}(\mathbf{B}v) \)

Vector-Matrix Multiplication Facts 2

• Vector-matrix multiplication distributes over vector addition:
  \( (\mathbf{v} + \mathbf{w})\mathbf{M} = \mathbf{vM} + \mathbf{wM} \)
• That was for row vectors \( \mathbf{v}, \mathbf{w} \). Similarly for column vectors.
Section 4.2: Matrix – a Geometric Interpretation

Matrices and Geometry
A square matrix can perform any *linear transformation*. What’s that?
- Preserves straight lines
- Preserves parallel lines.
- No translation: the axes do not move.

Linear Transformations
- Rotation
- Scaling
- Orthographic projection
- Reflection
- Shearing
- More about these in the next chapter.

A Movie Quote
- “Unfortunately, no-one can be told what The Matrix is – you have to see it for yourself.”
- Actually, it’s all about *basis vectors*.
- Look back to Chapter 3 if you’ve forgotten about those.

Axial Displacements
Can rewrite any vector \( \mathbf{v} = [x \ y \ z] \) as a sum of *axial displacements*.
\[
\mathbf{V} = [x \ y \ z] = [x \ 0 \ 0] + [0 \ y \ 0] + [0 \ 0 \ z] = x \ [1 \ 0 \ 0] + y \ [0 \ 1 \ 0] + z \ [0 \ 0 \ 1]
\]

Basis Vectors
- Define three unit vectors along the axes:
  \[ \mathbf{p} = [1 \ 0 \ 0] \]
  \[ \mathbf{q} = [0 \ 1 \ 0] \]
  \[ \mathbf{r} = [0 \ 0 \ 1]. \]
- Then we can rewrite the axial displacement equation as
  \[ \mathbf{v} = x \mathbf{p} + y \mathbf{q} + z \mathbf{r} \]
- \( \mathbf{p}, \mathbf{q}, \mathbf{r} \) are known as *basis vectors*.
Arbitrary Basis Vectors

- Can use any three linearly independent vectors
  \[ p = [p_x, p_y, p_z] \]
  \[ q = [q_x, q_y, q_z] \]
  \[ r = [r_x, r_y, r_z] \]
- Linearly independent means that there do not exist scalars \( a, b, c \) such that:
  \[ ap + bq + cr = 0 \]

Orthonormal Basis Vectors

- Best to use an orthonormal basis
- Orthonormal means unit vectors that are pairwise orthogonal:
  \[ p \cdot q = q \cdot r = r \cdot p = 0 \]
- Otherwise things can get weird.

Matrix From Basis Vectors

Construct a matrix \( M \) using \( p, q, r \) as the rows of the matrix:

\[
M = \begin{bmatrix}
p_x & p_y & p_z \\
q_x & q_y & q_z \\
r_x & r_y & r_z \\
\end{bmatrix}
\]

What Does This Matrix Do?

\[
\begin{bmatrix}
x \\ y \\ z \\
\end{bmatrix}
= \begin{bmatrix}
p_x & p_y & p_z \\
q_x & q_y & q_z \\
r_x & r_y & r_z \\
\end{bmatrix}
\begin{bmatrix}
x_p \\ y_q \\ z_r \\
\end{bmatrix}
= \begin{bmatrix}
xp_x + yq_x + zr_x \\ xp_y + yq_y + zr_y \\ xp_z + yq_z + zr_z \\
\end{bmatrix}
= \begin{bmatrix}
xp + yq + zr \\
\end{bmatrix}
\]

Transformation by a Matrix

- If we interpret the rows of a matrix as the basis vectors of a coordinate space, then multiplication by the matrix performs a coordinate space transformation.
- If \( aM = b \), we say that vector \( a \) is transformed by the matrix \( M \) into vector \( b \).

Conversely

See what \( M \) does to the original basis vectors \( [1 \ 0 \ 0], [0 \ 1 \ 0], [0 \ 0 \ 1] \).

\[
\begin{bmatrix}
m_{11} & m_{12} & m_{13} \\
m_{21} & m_{22} & m_{23} \\
m_{31} & m_{32} & m_{33} \\
\end{bmatrix}
\begin{bmatrix}
m_{11} \\
m_{12} \\
m_{13} \\
\end{bmatrix}
= \begin{bmatrix}
m_{11} \\
m_{21} \\
m_{31} \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
m_{11} & m_{12} & m_{13} \\
m_{21} & m_{22} & m_{23} \\
m_{31} & m_{32} & m_{33} \\
\end{bmatrix}
\begin{bmatrix}
m_{21} \\
m_{22} \\
m_{23} \\
\end{bmatrix}
= \begin{bmatrix}
m_{21} \\
m_{22} \\
m_{23} \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
m_{11} & m_{12} & m_{13} \\
m_{21} & m_{22} & m_{23} \\
m_{31} & m_{32} & m_{33} \\
\end{bmatrix}
\begin{bmatrix}
m_{31} \\
m_{32} \\
m_{33} \\
\end{bmatrix}
= \begin{bmatrix}
m_{31} \\
m_{32} \\
m_{33} \\
\end{bmatrix}
\]
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 Matrix Example

- What does the following 2D matrix do?
  \[ M = \begin{bmatrix} 2 & 1 \\ \ -1 & 2 \end{bmatrix} \]
- Extract the basis vectors (the rows of \( M \))
  \[ p = [2 \ 1] \]
  \[ q = [-1,2] \]

What’s the Transformation?

- It moves the unit axes \([1, 0]\) and \([0, 1]\) to the new axes.
- It does the same thing to all vectors.
- Visualize a box being transformed from one coordinate system to the other.
- This is called a skew box.
So What Does It Do?

- Rotates objects counterclockwise by a small amount.
- Scales them up by a factor of two.

3D Transformation Example

Before

After

What’s the Matrix?

Get rows of matrix from new basis vectors.

\[
\begin{bmatrix}
0.707 & -0.707 & 0 \\
1.250 & 1.250 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

So what does it do?
- Rotates objects clockwise by 45°.
- Scales them up along the y axis.

Constructing & Deconstructing Matrices

- By interpreting the rows of a matrix as basis vectors, we have a tool for deconstructing a matrix.
- But we also have a tool for constructing one! Given a desired transformation (e.g. rotation, scale, etc.), we can derive a matrix which represents that transformation.
- All we have to do is figure out what the transformation does to basis vectors, and then place those transformed basis vectors into the rows of a matrix.
- We’ll use this tool repeatedly in Chapter 5 to derive the matrices to perform the linear basic transformations such as rotation, scale, shear, and reflection that we mentioned earlier.

That concludes Chapter 4. Next, Chapter 5: Matrices and Linear Transformations
Chapter 5
Matrices & Linear Transforms

What You’ll See in This Chapter
This chapter is concerned with expressing linear transformations in 3D using 3x3 matrices. It is divided roughly into two parts.
- In the first part, Sections 5.1-5.5, we take the basic tools from previous chapters to derive matrices for primitive linear transformations of rotation, scaling, orthographic projection, reflection, and shearing.
- The second part of this chapter returns to general principles of transformations.
  - Section 5.6 shows how a sequence of primitive transformations may be combined using matrix multiplication to form a more complicated transformation.
  - Section 5.7 discusses various interesting categories of transformations, including linear, affine, invertible, angle-preserving, orthogonal, and rigid-body transforms.

Reminder: Visualize The Matrix (Chapter 4)
- 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!

Construct Matrix from Basis Vectors

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.”
• We’re going to obey the left-hand rule for this (review Chapter 1).

3D Rotate About x-axis

Compare to 2D Case

3D Rotate About y-axis
### 3D Rotate About z-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.
- Game programmers worry less about this because rotation about an arbitrary axis can be expressed as a sum of rotations about cardinal axes (Euler).
- Details are left to the book.

### Scaling Along Cardinal Axes in 2D

- **$R(\theta)$**:
  
  \[
  R(\theta) = \begin{bmatrix}
  \cos \theta & \sin \theta \\
  -\sin \theta & \cos \theta
  \end{bmatrix}
  \]

- **$R_y(\theta)$**:
  
  \[
  R_y(\theta) = \begin{bmatrix}
  \cos \theta & 0 & -\sin \theta \\
  0 & 1 & 0 \\
  \sin \theta & 0 & \cos \theta
  \end{bmatrix}
  \]

- **$R_z(\theta)$**:
  
  \[
  R_z(\theta) = \begin{bmatrix}
  \cos \theta & \sin \theta & 0 \\
  -\sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \]
**Basis Vectors for Scale**

The basis vectors \( \mathbf{p} \) and \( \mathbf{q} \) are independently affected by the corresponding scale factors:

\[
\begin{align*}
\mathbf{p}' &= k_x \mathbf{p} = k_x \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} k_x \\ 0 \end{bmatrix} \\
\mathbf{q}' &= k_y \mathbf{q} = k_y \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ k_y \end{bmatrix}
\end{align*}
\]

**2D Scale Matrix**

Constructing the 2D scale matrix \( S(k_x, k_y) \) from these basis vectors:

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

**3D Scale Matrix**

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

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

**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 again left to the book.

**Orthographic Projection**

Section 5.3: 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.
- For completeness, we present the matrices for these transformations:

\[
\begin{align*}
P_x &= S ([0 \ 1], 0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 \end{bmatrix} \\
P_y &= S ([1 \ 0], 0) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 \end{bmatrix} \\
P_{xy} &= S ([0 \ 0 \ 1], 0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 \end{bmatrix} \\
P_{xz} &= S ([0 \ 1 \ 0], 0) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 \end{bmatrix} \\
P_{yz} &= S ([1 \ 0 \ 0], 0) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 \end{bmatrix}
\end{align*}
\]

Reflection

Section 5.4: Reflection

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 \( \mathbf{n} \):

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

\[
= \begin{bmatrix} 1 - 2n_x^2 & -2n_x n_y & 2n_x n_z \\ -2n_x n_y & 1 - 2n_y^2 & -2n_y n_z \\ 2n_x n_z & -2n_y n_z & 1 - 2n_z^2 \\
\end{bmatrix}
\]
Section 5.5: 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 \( H_x(s) \) be the shear matrix that shears the \( x \) coordinate by the other coordinate, \( y \), by amount \( s \).

\[
H_x(s) = \begin{bmatrix} 1 & 0 & 0 \\ s & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},
\]

\[
H_y(s) = \begin{bmatrix} 1 & s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\]

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, \( z \).
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.
- We wish to render this object given a camera in any position and orientation.
- To do this, we must take the vertices of the object (assuming we are rendering some sort of triangle mesh) and transform them from object space into world space.
- This transform is known as the model transform, which we'll denote $M_{\text{obj}\rightarrow\text{wld}}$.
- From there, we transform world-space vertices using the view transform, denoted $M_{\text{wld}\rightarrow\text{cam}}$ into camera space.
- The math involved is summarized on the next slide.

Geometric Interpretation

- So we see that matrix concatenation works from an algebraic perspective using the associative property of matrix multiplication.
- Let's see if we can get a more geometric interpretation.
- Recall that 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 $B$.
- Let the row vectors $a_1$, $a_2$, and $a_3$ stand for the rows of $A$.
- Then matrix multiplication can alternatively be written like this…

Classes of Transformations

- Linear transformations
- Affine transformations
- Invertible transformations
- Angle preserving transformations
- Orthogonal transformations
- Rigid body transformations

Geometric Interpretation

$$A = \begin{bmatrix} -a_1 & -a_2 & -a_3 \end{bmatrix}$$

$$AB = \begin{bmatrix} -a_1 & -a_2 & -a_3 \end{bmatrix} B = \begin{bmatrix} -a_1B & -a_2B & -a_3B \end{bmatrix}$$

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 $B$. 
Disclaimer

- When we discuss transformations in general, we make use of the synonymous terms mapping or function.
- In the most general sense, a mapping is simply a rule that takes an input and produces an output. We denote that the mapping \( F \) maps \( a \) to \( b \) by writing \( F(a) = b \). (Read “\( F \) of \( a \) equals \( b \).”)
- We are mostly interested in the transformations that can be expressed as matrix multiplication, but others are possible.
- In this section we use the determinant of a matrix. We’re getting a bit ahead of ourselves, since we won’t give a full explanation of determinants until Chapter 6.
- So for now, just know that the determinant of a matrix is a scalar quantity that is very useful for making certain high-level, shall we say, determinations about the matrix. 😊

Section 5.7: Classes of Transformations

Linear Transformations

- A mapping \( F(a) \) is linear if \( F(a + b) = F(a) + F(b) \) and \( F(ka) = kF(a) \).
- The mapping \( F(a) = aM \), where \( M \) is any square matrix, is a linear transformation, because matrix multiplication satisfies the equations in the first bullet point of this slide:
  \[
  F(a + b) = (a + b)M = aM + bM = F(a) + F(b)
  \]
  \[
  F(ka) = (ka)M = k(aM) = kF(a)
  \]

The Zero Vector

- Any linear transformation will transform the zero vector into the zero vector.
- If \( F(0) = a \) and \( a \neq 0 \), then \( F \) cannot be a linear transformation, since \( F(k0) = a \) and therefore \( F(k0) \neq kF(0) \).
- Therefore:
  - Any transformation that can be accomplished with matrix multiplication is a linear transformation.
  - Linear transformations do not include translation.

Caveats

- In some literature, a linear transformation is defined as one in which parallel lines remain parallel after transformation.
- This is almost completely accurate, with two exceptions.
- First, parallel lines remain parallel after translation, but translation is not a linear transformation.
- Second, what about projection? When a line is projected and becomes a single point, can we consider that point parallel to anything?
- Excluding these technicalities, the intuition is correct: a linear transformation may stretch things, but straight lines are not warped and parallel lines remain parallel.

Affine Transformations

- An affine transformation is a linear transformation followed by translation.
- Thus, the set of affine transformations is a superset of the set of linear transformations: any linear transformation is an affine transformation, but not all affine transformations are linear transformations.
- Since all of the transformations we discussed so far in this chapter are linear transformations, they are all also affine transformations. (Though none of them have a translation portion.)
- Any transformation of the form \( v' = vM + b \) is an affine transformation.
Invertible Transformations

- A transformation is invertible if there exists an opposite transformation, known as the inverse of F, that undoes the original transformation.
- In other words, a mapping \( F(a) \) is invertible if there exists an inverse mapping \( F^{-1} \) such that for all \( a \), \( F^{-1}(F(a)) = F(F^{-1}(a)) = a \).
- This implies that \( F^{-1} \) is also invertible.
- There are non-affine invertible transformations, but we will not consider them for the moment.

Are All Affine Transforms Invertible?

- An affine transformation is a linear transformation followed by a translation.
- Obviously, we can always undo the translation portion by simply translating by the opposite amount.
- So the question becomes whether or not the linear transformation is invertible.

Are All Linear Transforms Invertible?

- Intuitively, we know that all of the transformations other than projection can be undone – if we rotate, scale, reflect, or skew, we can always unrotate, unscale, unreflect, or unskew.
- But when an object is projected, we effectively discard one or more dimensions’ worth of information, and this information cannot be recovered.
- Thus all of the primitive transformations other than projection are invertible.

Are All Matrices Invertible? No.

- Since any linear transformation can be expressed as multiplication by a matrix, finding the inverse of a linear transformation is equivalent to finding the inverse of a matrix.
- We will discuss how to do this in Chapter 6. If the matrix has no inverse, we say that it is singular, and the transformation is non-invertible.
- We can use a value called the determinant to determine whether a matrix is invertible.
  - The determinant of an invertible matrix is nonzero.
  - The determinant of a non-invertible matrix is zero.

Singularity and Square Matrices

- When a square matrix is singular, its basis vectors are not linearly independent.
- If the basis vectors are linearly independent, then it they have full rank, and coordinates of any given vector in the span are uniquely determined.
- If the vectors are linearly independent, then there is a portion of the space that is not in the span of the basis.
- This is known as the null space of the matrix.
- If we transform vectors in the null space using the matrix, many vectors will be projected into the same vector in the span of the basis, and we won’t have any way to differentiate them.

Angle Preserving Transformations

- A transformation is angle-preserving if the angle between two vectors is not altered in either magnitude or direction after transformation.
- Only translation, rotation, and uniform scale are angle-preserving transformations.
- An angle-preserving matrix preserves proportions.
- We do not consider reflection an angle-preserving transformation because even though the magnitude of angle between two vectors is the same after transformation, the direction of angle may be inverted.
- All angle-preserving transformations are affine and invertible.
Orthogonal Transformations

- **Orthogonal** is a term that describes a matrix whose rows form an orthonormal basis – the axes are perpendicular to each other and have unit length.
- Orthogonal transformations are interesting because it is easy to compute their inverse, and they arise frequently in practice.
- We will talk more about orthogonal matrices in Chapter 6.
- Translation, rotation, and reflection are the only orthogonal transformations.
- Orthogonal matrices preserve the magnitudes of angles, areas, and volumes, but possibly not the signs.
- The determinant of an orthogonal matrix is 1.
- All orthogonal transformations are affine and invertible.

Rigid Body Transformations

- A **rigid body transformation** is one that changes the location and orientation of an object, but not its shape.
- All angles, lengths, areas, and volumes are preserved.
- Translation and rotation are the only rigid body transformations.
- Reflection is not considered a rigid body transformation.
- Rigid body transformations are also known as *proper transformations*.
- All rigid body transformations are orthogonal, angle-preserving, invertible, and affine.
- Rigid body transforms are the most restrictive class of transforms discussed in this section, but they are also extremely common in practice.
- The determinant of a rigid body transformation matrix is 1.

Transformation Summary

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Linear</th>
<th>Affine</th>
<th>Invertible</th>
<th>Angle-preserving</th>
<th>Orthogonal</th>
<th>Rigid body</th>
<th>Lengthpreserved</th>
<th>Area/volumespreserved</th>
<th>Determinant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear transformations</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Affine transformations</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Invertible</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Angle-preserving transformations</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Orthogonal transformations</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Rigid body transformations</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Translation</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Scale</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Non-uniform scale</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Reflection</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
<tr>
<td>Shearing</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Angle-preserving</td>
<td>Orthogonal</td>
<td>Rigid body</td>
<td>Lengths preserved</td>
<td>Areas/volumes preserved</td>
<td>Determinant</td>
</tr>
</tbody>
</table>

Next

- That concludes Chapter 5: Matrices and Linear Transformations.
- Next will be Chapter 6: More on Matrices.
What You’ll See in This Chapter

This chapter completes our coverage of matrices by discussing a few more interesting and useful matrix operations. It is divided into five sections.

• Section 6.1 covers the determinant of a matrix.
• Section 6.2 covers the inverse of a matrix.
• Section 6.3 discusses orthogonal matrices.
• Section 6.4 introduces homogeneous vectors and 4×4 matrices, and shows how they can be used to perform affine transformations in 3D.
• Section 6.5 discusses perspective projection and shows how to do it with a 4×4 matrix.

Chapter 6

More on Matrices

Fletcher Dunn
Valve Software

Ian Parberry
University of North Texas

3D Math Primer for Graphics & Game Development

Determinant

• Determinant is defined for square matrices.
• Denoted $|M|$ or det $M$.
• Determinant of a 2x2 matrix is

$$|M| = \begin{vmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{vmatrix} = m_{11}m_{22} - m_{12}m_{21}$$

2 x 2 Example

$$\begin{vmatrix} 2 & 1 \\ -1 & 2 \end{vmatrix} = (2)(2) - (1)(-1) = 4 + 1 = 5$$

$$\begin{vmatrix} -3 & 4 \\ 2 & 5 \end{vmatrix} = (-3)(5) - (4)(2) = -15 - 8 = -23$$

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$
3 x 3 Determinant

\[
\begin{vmatrix}
  m_{11} & m_{12} & m_{13} \\
  m_{21} & m_{22} & m_{23} \\
  m_{31} & m_{32} & m_{33}
\end{vmatrix}
\]

\[=
 m_{11}m_{22}m_{33} + m_{12}m_{23}m_{31} + m_{13}m_{21}m_{32} \\
- m_{11}m_{22}m_{31} - m_{12}m_{23}m_{33} - m_{13}m_{21}m_{32} \\
+ m_{11}(m_{22}m_{33} - m_{23}m_{32}) \\
+ m_{12}(m_{23}m_{31} - m_{21}m_{33}) \\
+ m_{13}(m_{21}m_{32} - m_{22}m_{31})
\]

3 x 3 Example

\[
\begin{vmatrix}
  -4 & -3 & 3 \\
  0 & 2 & -2 \\
  1 & 4 & -1
\end{vmatrix}
\]

\[=
 (-4)((-2)(1) - (-2)(-4)) \\
+(-3)((-2)(1) - (0)(-1)) \\
+(3)((0)(4) - (2)(1)) \\
+(-4)((-2)(-8)) \\
+(-3)((-2) - (0)) \\
+(3)(0 - (2)) \\
+(-3)(-2) \\
+(3)(-2) \\
= -24
\]

Triple Product

If we interpret the rows of a 3x3 matrix as three vectors, then the determinant of the matrix is equivalent to the so-called triple product of the three vectors:

\[
\begin{vmatrix}
  a_x & a_y & a_z \\
  b_x & b_y & b_z \\
  c_x & c_y & c_z
\end{vmatrix}
\]

\[=
 (a_yb_z - a_zb_y)c_x \\
+ (a_zb_x - a_xb_z)c_y \\
+ (a_xb_y - a_yb_x)c_z
\]

= \(\mathbf{a} \times \mathbf{b}\) \cdot \mathbf{c}

Minors

- Let \(M\) be an \(r \times c\) matrix.
- Consider the matrix obtained by deleting row \(i\) and column \(j\) from \(M\).
- This matrix will obviously be \((r-1) \times (c-1)\).
- The determinant of this submatrix, denoted \(M_{ij}\), is known as a minor of \(M\).
- For example, the minor \(M_{12}\) is the determinant of the 2 \(\times\) 2 matrix that is the result of deleting the 1st row and 2nd column from \(M\):

\[
M = \begin{bmatrix}
-4 & -3 & 3 \\
0 & 2 & -2 \\
1 & 4 & -1
\end{bmatrix}
\]

\[\Rightarrow M_{12} = \begin{vmatrix}
0 & -2 \\
1 & -1
\end{vmatrix}
= 2
\]

Cofactors

- The cofactor of a square matrix \(M\) at a given row and column is the same as the corresponding minor, only every alternating minor is negated.
- We will use the notation \(C_{ij}\) to denote the cofactor of \(M\) in row \(i\), column \(j\).
- Use \((-1)^{i+j}\) term to negate alternating minors.

\[C_{ij} = (-1)^{i+j}M_{ij}\]

Negating Alternating Minors

The \((-1)^{i+j}\) term negates alternating minors in this pattern:

\[
\begin{vmatrix}
  + & - & + & - & \cdots \\
  - & + & - & + & \cdots \\
  + & - & + & - & \cdots \\
  - & + & - & + & \cdots \\
  \vdots & \vdots & \vdots & \vdots & \ddots
\end{vmatrix}
\]
**n x n Determinant**

- The definition we will now consider expresses a determinant in terms of its cofactors.
- This definition is recursive, since cofactors are themselves signed determinants.
- First, we arbitrarily select a row or column from the matrix.
- Now, for each element in the row or column, we multiply this element by the corresponding cofactor.
- Summing these products yields the determinant of the matrix.

**Example**

For example, arbitrarily selecting row $i$, the determinant can be computed by:

$$|M| = \sum_{j=1}^{n} m_{ij} C^{(ij)} = \sum_{j=1}^{n} m_{ij} (-1)^{i+j} M^{(ij)}$$

**3 x 3 Determinant**

$$\begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{vmatrix} = m_{11} \begin{vmatrix} m_{22} & m_{23} \\ m_{32} & m_{33} \end{vmatrix} - m_{12} \begin{vmatrix} m_{21} & m_{23} \\ m_{31} & m_{33} \end{vmatrix} + m_{13} \begin{vmatrix} m_{21} & m_{22} \\ m_{31} & m_{32} \end{vmatrix}$$

**4 x 4 Determinant**

$$\begin{vmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{vmatrix} = m_{11} \begin{vmatrix} m_{22} & m_{23} & m_{24} \\ m_{32} & m_{33} & m_{34} \\ m_{42} & m_{43} & m_{44} \end{vmatrix} - m_{12} \begin{vmatrix} m_{21} & m_{23} & m_{24} \\ m_{31} & m_{33} & m_{34} \\ m_{41} & m_{43} & m_{44} \end{vmatrix} + m_{13} \begin{vmatrix} m_{21} & m_{22} & m_{24} \\ m_{31} & m_{32} & m_{34} \\ m_{41} & m_{42} & m_{44} \end{vmatrix} - m_{14} \begin{vmatrix} m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \\ m_{41} & m_{42} & m_{43} \end{vmatrix}$$

**Expanding Cofactors**

This equals:

$$m_{11}[(m_{22}m_{33}m_{44} - m_{23}m_{32}m_{44} + m_{24}m_{32}m_{43} - m_{23}m_{34}m_{42})] - m_{12}[(m_{23}m_{34}m_{42} - m_{24}m_{32}m_{43} + m_{22}m_{34}m_{43} - m_{24}m_{32}m_{44})] + m_{13}[(m_{24}m_{32}m_{43} - m_{23}m_{34}m_{42} + m_{22}m_{34}m_{43} - m_{23}m_{32}m_{44})] - m_{14}[(m_{23}m_{32}m_{44} - m_{24}m_{32}m_{43} + m_{22}m_{32}m_{44} - m_{23}m_{32}m_{44})]$$

**Important Determinant Facts**

- The identity matrix has determinant 1: $|I| = 1$.
- The determinant of a matrix product is equal to the product of the determinants:
  $$|AB| = |A||B|.$$  
- This extends to multiple matrices:
  $$|M_1M_2...M_{n-1}M_n| = |M_1||M_2|...|M_{n-1}||M_n|.$$  
- The determinant of the transpose of a matrix is equal to the original.
  $$|M^T| = |M|.$$
Zero Row or Column

- If any row of column in a matrix contains all zeros, then the determinant of that matrix is zero.

\[
\begin{vmatrix}
? & ? & \cdots & ? \\
? & ? & \cdots & ? \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 \\
\end{vmatrix} = 0
\]

Exchanging Rows or Columns

Exchanging any pair of rows or columns negates the determinant.

\[
\begin{vmatrix}
m_{11} & m_{12} & \cdots & m_{1n} \\
m_{21} & m_{22} & \cdots & m_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
m_{n1} & m_{n2} & \cdots & m_{nn} \\
\end{vmatrix} \rightarrow
\begin{vmatrix}
m_{11} & m_{12} & \cdots & m_{1n} \\
m_{21} & m_{22} & \cdots & m_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
m_{n1} & m_{n2} & \cdots & m_{nn} \\
\end{vmatrix}
\]

Adding a Multiple of a Row or Column

- Adding any multiple of a row (or column) to another row (or column) does not change the value of the determinant.
- This explains why shear matrices from Chapter 5 have a determinant of 1.

Geometric Interpretation

- In 2D, the determinant is equal to the signed area of the parallelogram or skew box that has the basis vectors as two sides.
- By signed area, we mean that the area is negative if the skew box is flipped relative to its original orientation.

2 x 2 Determinant as Area

\[
A = \begin{vmatrix}
0.67 & 0.82 \\
0.67 & 2.00 \\
\end{vmatrix}
\]

\[
A = 0.67 \times 2.00 - 0.82 \times 0.67 = 1.34 - 0.54 = 0.80
\]
3 x 3 Determinant as Volume

- In 3D, the determinant is the volume of the parallelepiped that has the transformed basis vectors as three edges.
- It will be negative if the object is reflected (turned inside out) as a result of the transformation.

Uses of the Determinant

- The determinant is related to the change in size that results from transforming by the matrix.
- The absolute value of the determinant is related to the change in area (in 2D) or volume (in 3D) that will occur as a result of transforming an object by the matrix.
- The determinant of the matrix can also be used to help classify the type of transformation represented by a matrix.
- If the determinant of a matrix is zero, then the matrix contains a projection.
- If the determinant of a matrix is negative, then the matrix contains a reflection.

Section 6.2: Inverse of a Matrix

Inverse of a Matrix

- The inverse of a square matrix \( \mathbf{M} \), denoted \( \mathbf{M}^{-1} \) is the matrix such that when we multiply by \( \mathbf{M}^{-1} \), the result is the identity matrix.
  \[
  \mathbf{M} \mathbf{M}^{-1} = \mathbf{M}^{-1} \mathbf{M} = \mathbf{I}.
  \]
- Not all matrices have an inverse.
- An obvious example is a matrix with a row or column of zeros: no matter what you multiply this matrix by, you will end up with the zero matrix.
- If a matrix has an inverse, it is said to be invertible or non-singular. A matrix that does not have an inverse is said to be non-invertible or singular.

Invertibility and Linear Independence

- For any invertible matrix \( \mathbf{M} \), the vector equality \( \mathbf{vM} = \mathbf{0} \) is true only when \( \mathbf{v} = \mathbf{0} \).
- Furthermore, the rows of an invertible matrix are linearly independent, as are the columns.
- The rows and columns of a singular matrix are linearly dependent.

Determinant and Invertibility

- The determinant of a singular matrix is zero and the determinant of a non-singular matrix is non-zero.
- Checking the magnitude of the determinant is the most commonly used test for invertibility, because it's the easiest and quickest.
The Classical Adjoint

• Our method for computing the inverse of a matrix is based on the classical adjoint.
• The classical adjoint of a matrix $M$, denoted $\text{adj } M$, is defined to be the transpose of the matrix of cofactors of $M$.
• For example, let:

$$M = \begin{bmatrix} 2 & -3 & 3 \\ 0 & 2 & -2 \\ 1 & 4 & -1 \end{bmatrix}$$

Computing the Cofactors

Compute the cofactors of $M$:

$$C^{(11)} = \begin{vmatrix} 2 & -2 \\ 1 & -1 \end{vmatrix} = 6$$
$$C^{(12)} = \begin{vmatrix} 4 & -2 \\ 1 & -1 \end{vmatrix} = -2$$
$$C^{(13)} = \begin{vmatrix} 4 & 2 \\ 1 & 4 \end{vmatrix} = -2$$
$$C^{(21)} = \begin{vmatrix} -3 & 3 \\ 1 & -1 \end{vmatrix} = 9$$
$$C^{(22)} = \begin{vmatrix} -3 & 3 \\ 1 & 4 \end{vmatrix} = 13$$
$$C^{(23)} = \begin{vmatrix} 2 & -2 \\ 1 & 4 \end{vmatrix} = -8$$
$$C^{(31)} = \begin{vmatrix} 2 & -2 \\ -3 & 3 \end{vmatrix} = -8$$
$$C^{(32)} = \begin{vmatrix} -3 & 3 \\ -3 & 3 \end{vmatrix} = -8$$

Classical Adjoint of $M$

The classical adjoint of $M$ is the transpose of the matrix of cofactors:

$$\text{adj } M = \begin{bmatrix} C^{(11)} & C^{(12)} & C^{(13)} \\ C^{(21)} & C^{(22)} & C^{(23)} \\ C^{(31)} & C^{(32)} & C^{(33)} \end{bmatrix}^T$$

$$= \begin{bmatrix} 6 & -2 & -2 \\ 9 & 1 & 13 \\ 0 & -8 & -8 \end{bmatrix}^T = \begin{bmatrix} 6 & 9 & 0 \\ -2 & 1 & -8 \\ -2 & 13 & -8 \end{bmatrix}$$

Back to the Inverse

• The inverse of a matrix is its classical adjoint divided by its determinant:

$$M^{-1} = \frac{\text{adj } M}{|M|}.$$  
• If the determinant is zero, the division is undefined, which jives with our earlier statement that matrices with a zero determinant are non-invertible.

Example of Matrix Inverse

If:

$$M = \begin{bmatrix} -4 & -3 & 3 \\ 0 & 2 & -2 \\ 1 & 4 & -1 \end{bmatrix}$$

$$M^{-1} = \frac{\text{adj } M}{|M|} = \frac{1}{-24} \begin{bmatrix} 6 & 9 & 0 \\ -2 & 1 & -8 \\ -2 & 13 & -8 \end{bmatrix} = \begin{bmatrix} -1/4 & -3/8 & 0 \\ 1/12 & -1/24 & 1/3 \\ 1/12 & -13/24 & 1/3 \end{bmatrix}$$

Gaussian Elimination

• There are other techniques that can be used to compute the inverse of a matrix, such as Gaussian elimination.
• Many linear algebra textbooks incorrectly assert that such techniques are better suited for implementation on a computer because they require fewer arithmetic operations.
• This is true for large matrices, or for matrices with a structure that can be exploited.
• However, for arbitrary matrices of smaller order like the 2 x 2, 3 x 3, and 4 x 4 used most often in geometric applications, the classical adjoint method is faster.
• The reason is that the classical adjoint method provides for a branchless implementation, meaning there are no if statements or loops that cannot be unrolled statically.
• This is a big win on today’s superscalar architectures and vector processors.
Facts About Matrix Inverse

- The inverse of the inverse of a matrix is the original matrix. If \( M \) is nonsingular, \( (M^{-1})^{-1} = M \).
- The identity matrix is its own inverse: \( I^{-1} = I \).
- Note that there are other matrices that are their own inverse, such as any reflection matrix, or a matrix that rotates 180° about any axis.
- The inverse of the transpose of a matrix is the transpose of the inverse: \( (M^T)^{-1} = (M^{-1})^T \)

More Facts About Matrix Inverse

- The inverse of a product is equal to the product of the inverses in reverse order.
  \[ (AB)^{-1} = B^{-1}A^{-1} \]
- This extends to more than two matrices:
  \[ (M_1M_2...M_{n-1}M_n)^{-1} = M_n^{-1}M_{n-1}^{-1}...M_2^{-1}M_1^{-1} \]
- The determinant of the inverse is the inverse of the determinant:
  \[ |M^{-1}| = 1/|M| \]

Geometric Interpretation of Inverse

- The inverse of a matrix is useful geometrically because it allows us to compute the reverse or opposite of a transformation – a transformation that undoes another transformation if they are performed in sequence.
- So, if we take a vector \( v \) and transform it by a matrix \( M \), and then transform it by the inverse \( M^{-1} \) of \( M \), then we will get \( v \) back.
- We can easily verify this algebraically:
  \[ (vM)M^{-1} = v(MM^{-1}) = vl = v \]

Orthogonal Matrices

- A square matrix \( M \) is orthogonal if and only if the product of the matrix and its transpose is the identity matrix: \( MM^T = I \).
- If a matrix is orthogonal, its transpose and the inverse are equal: \( M^T = M^{-1} \).
- If we know that our matrix is orthogonal, we can essentially avoid computing the inverse, which is a relatively costly computation.
- For example, rotation and reflection matrices are orthogonal.

Section 6.3: Orthogonal Matrices

Testing Orthogonality

Let \( M \) be a 3 x 3 matrix. Let's see exactly what it means when \( MM^T = I \).

\[
\begin{bmatrix}
m_{11} & m_{12} & m_{13} \\
m_{21} & m_{22} & m_{23} \\
m_{31} & m_{32} & m_{33}
\end{bmatrix}
\begin{bmatrix}
m_{11} & m_{21} & m_{31} \\
m_{12} & m_{22} & m_{32} \\
m_{13} & m_{23} & m_{33}
\end{bmatrix}
= 
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]
9 Equations

This gives us 9 equations, all of which must be true in order for $M$ to be orthogonal:

\[
\begin{align*}
&m_{11}m_{11} + m_{12}m_{12} + m_{13}m_{13} = 1 \\
&m_{11}m_{21} + m_{12}m_{22} + m_{13}m_{23} = 0 \\
&m_{11}m_{31} + m_{12}m_{32} + m_{13}m_{33} = 0 \\
&m_{21}m_{11} + m_{22}m_{12} + m_{23}m_{13} = 0 \\
&m_{21}m_{21} + m_{22}m_{22} + m_{23}m_{23} = 1 \\
&m_{21}m_{31} + m_{22}m_{32} + m_{23}m_{33} = 0 \\
&m_{31}m_{11} + m_{32}m_{12} + m_{33}m_{13} = 0 \\
&m_{31}m_{21} + m_{32}m_{22} + m_{33}m_{23} = 0 \\
&m_{31}m_{31} + m_{32}m_{32} + m_{33}m_{33} = 1
\end{align*}
\]

Consider the Rows

Let the vectors $r_1$, $r_2$, $r_3$ stand for the rows of $M$:

\[
\begin{align*}
&\begin{bmatrix} m_{11} & m_{12} & m_{13} \end{bmatrix} \\
&\begin{bmatrix} m_{21} & m_{22} & m_{23} \end{bmatrix} \\
&\begin{bmatrix} m_{31} & m_{32} & m_{33} \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
&\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} -r_1 \\ -r_2 \\ -r_3 \end{bmatrix}
\end{align*}
\]

9 Equations Using Dot Product

Now we can re-write the 9 equations more compactly:

\[
\begin{align*}
&r_1 \cdot r_1 = 1 \\
&r_1 \cdot r_2 = 0 \\
&r_1 \cdot r_3 = 0 \\
&r_2 \cdot r_1 = 0 \\
&r_2 \cdot r_2 = 1 \\
&r_2 \cdot r_3 = 0 \\
&r_3 \cdot r_1 = 0 \\
&r_3 \cdot r_2 = 0 \\
&r_3 \cdot r_3 = 1
\end{align*}
\]

Two Observations

- First, the dot product of a vector with itself is 1 if and only if the vector is a unit vector.
- Therefore, the equations with a 1 on the right hand side of the equals sign will only be true when $r_1$, $r_2$, and $r_3$ are unit vectors.
- Second, the dot product of two vectors is 0 if and only if they are perpendicular.
- Therefore, the other six equations (with 0 on the right hand side of the equals sign) are true when $r_1$, $r_2$, and $r_3$ are mutually perpendicular.

Conclusion

- So, for a matrix to be orthogonal, the following must be true:
  1. Each row of the matrix must be a unit vector.
  2. The rows of the matrix must be mutually perpendicular.
- Similar statements can be made regarding the columns of the matrix, since if $M$ is orthogonal, then $M^T$ must be orthogonal.

Orthonormal Bases Revisited

- Notice that these criteria are precisely those that we said in Chapter 3 were satisfied by an orthonormal set of basis vectors.
- There we also noted that an orthonormal basis was particularly useful because we could perform the “opposite” coordinate transform from the one that is always available, using the dot product.
- When we say that the transpose of an orthogonal matrix is its inverse, we are just restating this fact in the formal language of linear algebra.
9 is Actually 6

- Also notice that 3 of the orthogonality equations are duplicates (since dot product is commutative), and between these 9 equations, we actually have 6 constraints, leaving 3 degrees of freedom.
- This is interesting, since 3 is the number of degrees of freedom inherent in a rotation matrix.
- But again note that rotation matrices cannot compute a reflection, so there is slightly more freedom in the set of orthogonal matrices than in the set of orientations in 3D.

A Note on Terminology

- In linear algebra, we described a set of basis vectors as **orthogonal** if they are mutually perpendicular.
- It is not required that they have unit length. If they do have unit length, they are an **orthonormal** basis.
- Thus the rows and columns of an orthogonal matrix are orthonormal basis vectors.
- However, constructing a matrix from a set of orthogonal basis vectors does not necessarily result in an orthogonal matrix (unless the basis vectors are also orthonormal).

Caveats

- When computing a matrix inverse we will usually only take advantage of orthogonality if we know *a priori* that a matrix is orthogonal.
- If we don’t know in advance, it’s probably a waste of time checking.
- Finally, even matrices which are orthogonal in the abstract may not be exactly orthogonal when represented in floating point, and so we must use tolerances, which have to be tuned.

Scary Monsters (Matrix Creep)

- Recall that that rotation matrices (and products of them) are orthogonal.
- Recall that the rows of an orthogonal matrix form an orthonormal basis.
- Or at least, that’s the way we’d like them to be.
- But the world is not perfect. Floating point numbers are subject to numerical instability.
- Aka “matrix creep” (apologies to David Bowie)
- We need to orthogonalize the matrix, resulting in a matrix that has mutually perpendicular unit vector axes and is (hopefully) as close to the original matrix as possible.

Gramm-Schmidt Orthogonalization

- Here’s how to control matrix creep.
- Go through the rows of the matrix in order.
- For each, subtract off the component that is parallel to the other rows.
- More details: let \( r_1, r_2, r_3 \) be the rows of a 3 x 3 matrix \( M \).
- Remember, you can also think of these as the \( x \)-, \( y \)-, and \( z \)-axes of a coordinate space.
- Then an orthogonal set of row vectors, \( r_1', r_2', r_3' \) can be computed as follows:

Steps 1 and 2

- Step 1: Normalize \( r_1 \) to get a new vector \( r_1' \) (meaning make its magnitude 1)
- Step 2: Replace \( r_2 \) by
  \[
  r_2' = r_2 - \langle r_2', r_1 \rangle r_1'
  \]
- \( r_2' \) is now perpendicular to \( r_1' \) because
  \[
  r_1', r_2' = r_1' - \langle r_1', r_2 \rangle r_1' = 0
  \]
Steps 3, 4, and 5

- Step 3: Normalize \( r_2' \)
- Step 4: Replace \( r_3 \) by
  \[ r_3' = r_3 - ( r_1'.r_3 ) r_1' - ( r_2'.r_3 ) r_2' \]
- Step 5: Normalize \( r_3' \)

Checking \( r_3' \) and \( r_1' \)

- \( r_3' \) is now perpendicular to \( r_1' \) because
  \[
  r_1'.r_3' = \left( r_1' - ( r_1'.r_3 ) r_1' - ( r_2'.r_3 ) r_2' \right) \cdot r_3
  = r_1'.r_3 - ( r_1'.r_3 ) ( r_1'.r_1' ) - ( r_2'.r_3 ) ( r_2'.r_2' )
  = r_1'.r_3 - r_1'.r_3 - 0
  = 0
  \]

Checking \( r_3' \) and \( r_2' \)

- \( r_3' \) is now perpendicular to \( r_2' \) because
  \[
  r_2'.r_3' = \left( r_2' - ( r_1'.r_3 ) r_1' - ( r_2'.r_3 ) r_2' \right) \cdot r_3
  = r_2'.r_3 - ( r_1'.r_3 ) ( r_2'.r_1' ) - ( r_2'.r_3 ) ( r_2'.r_2' )
  = r_2'.r_3 - 0 - r_2'.r_3
  = 0
  \]

Bias

- This is biased towards \( r_3 \), meaning that \( r_1 \) doesn’t change but the other basis vectors do change.
- Option: instead of subtracting off the whole amount, subtract off a fraction of the original axis.
- Let \( k \) be a fraction – say 1/4

Gramm-Schmidt in Practice

- Step 1: normalize \( r_1', r_2', r_3 \)
- Step 2: repeat a dozen or so times:
  \[
  r_1' = r_1 - k ( r_1.r_2 ) r_2 - k ( r_1.r_3 ) r_3
  r_2' = r_2 - k ( r_1.r_2 ) r_1 - k ( r_2.r_3 ) r_3
  r_3' = r_3 - k ( r_1.r_3 ) r_1 - k ( r_2.r_3 ) r_2
  \]
- Step 3: Do a vanilla Gramm-Schmidt to catch any remaining “abnormality”

Section 6.3: 4×4 Homogenous Matrices
Homogenous Coordinates

- Extend 3D into 4D.
- The 4th dimension is not “time”.
- The 4th dimension is really just a kluge to help the math work out (later in this lecture).
- The 4th dimension is called \( w \).

Extending 1D into Homogenous Space

- Start with 1D, its easier to visualize than 3D.
- Homogenous 1D coords are of the form \((x, w)\).
- Imagine the vanilla 1D line lying at \( w = 1 \).
- So the 1D point \( x \) has homogenous coords \((x, 1)\).
- Given a homogenous point \((x, w)\), the corresponding 1D point is its projection onto the line \( w = 1 \) along a line to the origin, which turns out to be \((x/w, 1)\).

Projecting Onto 1D Space

- Each point \( x \) in 1D space corresponds to an infinite number of points in homogenous space, those on the line from the origin through the point \((x, 1)\).
- The homogenous points on this line project onto its intersection with the line \( w = 1 \).

What are the 2D Coords of Homogenous Point \((p, q)\)?

- So equation of line is \( w = qx/p \).
- Therefore, when \( w = 1 \), \( x = p/q \).
- This means that \( r = p/q \).
- So the homogenous point \((p, q)\) projects onto the 1D point \((p/q, 1)\).
- That is, the 1D equivalent of the homogenous point \((p, q)\) is \( p/q \).

Simultaneous Equations

- Equation of line is \( w = ax + b \).
- \((p, q)\) and \((0, 0)\) are on the line.
- Therefore:
  \[
  q = ap + b \\
  0 = a0 + b, 
  \]
- That is, \( b = 0 \) and \( a = q/p \).
Extending 2D into Homogenous Space

• 2D next, it’s still easier to visualize than 3D.
• Homogenous 2D coordinates are of the form \((x, y, w)\).
• Imagine the vanilla 2D plane lying at \(w = 1\).
• So the 2D point \((x, y)\) has homogenous coordinates \((x, y, 1)\).

Projecting Onto 2D Space

• Each point \((x, y)\) in 2D space corresponds to an infinite number of points in homogenous space.
• Those on the line from the origin thru \((x, y, 1)\).
• The homogenous points on this line project onto its intersection with the plane \(w = 1\).

2D Homogenous Coordinates

• Just like before (argument omitted), the homogenous point \((x, y, w)\) corresponds to the 2D point \((x/w, y/w, 1)\).
• That is, the 2D equivalent of the homogenous point \((p, q, r)\) is \((p/r, q/r)\).

3D Homogenous Coordinates

• This extends to 3D in the obvious way.
• The homogenous point \((x, y, z, w)\) corresponds to the 3D point \((x/w, y/w, z/w, 1)\).
• That is, the 3D equivalent of the homogenous point \((p, q, r, s)\) is \((p/s, q/s, r/s)\).

Point at Infinity

• \(w\) can be any value except 0 (divide by zero error).
• The point \((x, y, z, 0)\) can be viewed as a “point at infinity”
Why Use Homogenous Space?

- It will let us handle translation with a matrix transformation.
- Embed 3D space into homogenous space by basically ignoring the w component.
- Vector \((x, y, z)\) gets replaced by \((x, y, z, 1)\).
- Does that “1” at the end sound familiar?

Homogenous Matrices

Embed 3D transformation matrix into 4D matrix by using the identity in the \(w\) row and column.

\[
\begin{bmatrix}
  m_{11} & m_{12} & m_{13} \\
  m_{21} & m_{22} & m_{23} \\
  m_{31} & m_{32} & m_{33}
\end{bmatrix} \Rightarrow \begin{bmatrix}
  m_{11} & m_{12} & m_{13} & 0 \\
  m_{21} & m_{22} & m_{23} & 0 \\
  m_{31} & m_{32} & m_{33} & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]

3D Matrix Multiplication

\[
\begin{bmatrix}
  x & y & z
\end{bmatrix}
\begin{bmatrix}
  m_{11} & m_{12} & m_{13} \\
  m_{21} & m_{22} & m_{23} \\
  m_{31} & m_{32} & m_{33}
\end{bmatrix}
= \begin{bmatrix}
  x m_{11} + y m_{21} + z m_{31} \\
  x m_{12} + y m_{22} + z m_{32} \\
  x m_{13} + y m_{23} + z m_{33}
\end{bmatrix}_T
\]

4D Matrix Multiplication

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

Translation Matrices

Kluge 3D translation matrix by shearing 4D space.

\[
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  \Delta x & \Delta y & \Delta z & 1
\end{bmatrix}
= \begin{bmatrix}
  x + \Delta x \\
  y + \Delta y \\
  z + \Delta z \\
  1
\end{bmatrix}
\]

Translation vs. Orientation

- Just like in 3D, compose 4D operations by multiplying the corresponding matrices.
- The translation and orientation parts of a composite matrix are independent.
- For example, let \(R\) be a rotation matrix and \(T\) be a translation matrix.
- What does \(M = RT\) look like?
Rotate then Translate

- Then we could rotate and then translate a point \( \mathbf{v} \) to a new point \( \mathbf{v}' \) using \( \mathbf{v}' = \mathbf{vRT} \).
- We are rotating first and then translating.
- The order of transformations is important, and since we use row vectors, the order of transformations coincides with the order that the matrices are multiplied, from left to right.

\[ \mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

\[ \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix} \]

\[ \mathbf{M} = \mathbf{RT} \]

• Just as with 3 x 3 matrices, we can concatenate the two matrices into a single transformation matrix, which we'll call \( \mathbf{M} \).

  Let \( \mathbf{M} = \mathbf{RT} \), so

  \[ \mathbf{v}' = \mathbf{vRT} = \mathbf{v}(\mathbf{RT}) = \mathbf{vM} \]

\[ \mathbf{M} = \begin{bmatrix} 11 & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

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

In Reverse

- Applying this information in reverse, we can take a 4 x 4 matrix \( \mathbf{M} \) and separate it into a linear transformation portion, and a translation portion.
- We can express this succinctly by letting the translation vector \( \mathbf{t} = [\Delta x, \Delta y, \Delta z] \).

\[ \mathbf{M} = \begin{bmatrix} \mathbf{R} & 0 \\ \mathbf{t} & 1 \end{bmatrix} \]

Points at Infinity Again

- Points at infinity are actually useful.
- They orient just like points with \( w \neq 0 \): multiply by the orientation matrix.
- But they don’t translate: translation matrices have no effect on them.
Matrix Without Translation

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

\[
= \begin{bmatrix}
  x'r_{11} + yr_{21} + zr_{31} \\
  x'r_{12} + yr_{22} + zr_{32} \\
  x'r_{13} + yr_{23} + zr_{33} \\
  0
  \end{bmatrix}
  \]

Matrix With Translation

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

\[
= \begin{bmatrix}
  x'r_{11} + yr_{21} + zr_{31} \\
  x'r_{12} + yr_{22} + zr_{32} \\
  x'r_{13} + yr_{23} + zr_{33} \\
  0
  \end{bmatrix}
  \]

\{ \text{Same} \}

So What?

• The translation part of 4D homogenous transformation matrices has no effect on points at infinity.
• Use points at infinity for things that don’t need translating (eg. Surface normals).
• Use regular points (with w = 1) for things that do need translating (eg. Points that make up game objects).

4x3 Matrices

• The last column of 4D homogenous transformation matrices is always [0, 0, 0, 1]^T.
• Technically it always needs to be there for the algebra to work out.
• But we know what it’s going to do, so there’s no reason to implement it in code.

General Affine Transformations

Armed with 4 x 4 transform matrices, we can now create more general affine transformations that contain translation. For example:
• Rotation about an axis that does not pass through the origin
• Scale about a plane that does not pass through the origin
• Reflection about a plane that does not pass through the origin
• Orthographic projection onto a plane that does not pass through the origin

General Affine Transformations

• The basic idea is to translate the center of the transformation to the origin, perform the linear transformation using the techniques developed in Chapter 5, and then transform the center back to its original location.
• We will start with a translation matrix T that translates the point p to the origin, and a linear transform matrix R from Chapter 5 that performs the linear transformation.
• The final transformation matrix A will be the equal to the matrix product \(TRT^{-1}\).
• \(T^{-1}\) is of course the translation matrix with the opposite translation amount as T.
Observation

- Thus, the extra translation in an affine transformation only changes the last row of the 4 x 4 matrix.
- The upper 3 x 3 portion, which contains the linear transformation, is not affected.
- Our use of homogenous coordinates so far has really been nothing more than a mathematical kludge to allow us to include translation in our transformations.

Section 6.4: 4x4 Matrices and Perspective Projection

Projections

- We’ve only used \( w = 1 \) and \( w = 0 \) so far.
- There’s a use for the other values of \( w \) too.
- We’ve seen how to do orthographic projection before.
- Now we’ll see how to do perspective projection too.

Orthographic Projection

- Orthographic projection has parallel projectors.
- The projected image is the same size no matter how far the object is from the projection plane.
- We want objects to get smaller with distance.
- This is known as perspective foreshortening.
The Pinhole Camera

- The math is based on a pinhole camera.
- Take a closed box that's very dark inside.
- Make a pinhole.
- If you point the pinhole at something bright, an image of the object will be projected onto the back of the box.
- That's kind of how the human eye works too.

Projection Geometry

- Let's project on a plane parallel to the x-y plane.
- Choose a distance \(d\) from the pinhole to the projection plane, called the focal distance.
- The pinhole is called the focal point.
- Put the focal point at the origin and the projection plane at \(z = -d\).
- (Remember the concept of camera space?)
Do the Math

- View it from the side.
- Consider where a point $p$ gets projected onto the plane – at a point $p'$.
- Start with the $y$ coordinate for now.

![Projection plane diagram](image)

Finding $p'$

From the previous slide, by similar triangles:

$$\frac{-p'y}{d} = \frac{py}{z} \quad \Rightarrow \quad p'y = -\frac{dp'y}{z}$$

Same for the $x$-axis:

$$p'_x = -\frac{dp_x}{z}$$

Result

- All projected points have a $z$ value of $-d$.
- Therefore $p$ is projected onto $p'$ like this:

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \Rightarrow \quad p' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \frac{-dx}{z} \\ \frac{-dy}{z} \\ -d \end{bmatrix}$$

In Practice

In practice we move the projection plane in front of the focal point.

Accentuate the Positive

Doing so removes the annoying minus signs. This:

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \Rightarrow \quad p' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \frac{-dx}{z} \\ \frac{-dy}{z} \\ -d \end{bmatrix}$$

Becomes this:

$$p' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \frac{dx}{z} \\ \frac{dy}{z} \\ d \end{bmatrix}$$
Projection Using 4D Matrix

- We can actually do this with a 4D homogenous matrix.
- First manipulate $p'$ to have a common denominator:
  \[
  p' = \begin{bmatrix}
  dx/z \\
  dy/z \\
  d/z \\
  
  \end{bmatrix}
  = \begin{bmatrix}
  dx/z \\
  dy/z \\
  dz/z \\
  
  \end{bmatrix}
  = (d/z) \begin{bmatrix}
  x \\
  y \\
  z \\
  
  \end{bmatrix}
  \]

Entering the 4th Dimension

- To multiply by $d/z$, divide by $z/d$.
- Instead of dividing by $z/d$, make that our $w$ coordinate:
  \[
  \begin{bmatrix}
  x \\
  y \\
  z \\
  z/d \\
  
  \end{bmatrix}
  \]
- We need a 4x4 matrix that transforms an “ordinary” point $[x \ y \ z \ 1]$ into this.

The Projection Matrix

\[
\begin{bmatrix}
  x & y & z & 1 \\
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 1/d \\
  0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
  x \\
  y \\
  z \\
  z/d \\
\end{bmatrix}
\]

Notes

- Multiplication by this matrix doesn't actually perform the perspective transform, it just computes the proper denominator into $w$. The perspective division actually occurs when we convert from 4D to 3D by dividing by $w$.
- There are many variations. For example, we can place the plane of projection at $z = 0$, and the center of projection at $[0, 0, -d]$. This results in a slightly different equation.

This Seems Overly Complicated.

- It seems like it would be simpler to just divide by $z$, rather than bothering with matrices.
- So why is homogenous space interesting?
  1. $4 \times 4$ matrices provide a way to express projection as a transformation that can be concatenated with other transformations.
  2. Projection onto non-axially aligned planes is possible.
- Basically, we don't need homogenous coordinates, but $4 \times 4$ matrices provide a compact way to represent and manipulate projection transformations.

Real Projection Matrices

- The projection matrix in a real graphics geometry pipeline (perhaps more accurately known as the clip matrix) does more than just copy $z$ into $w$. It will differ from the one we derived in two important respects:
  1. Most graphics systems apply a normalizing scale factor such that $w = 1$ at the far clip plane. This ensures that the values used for depth buffering are distributed appropriately for the scene being rendered, in order to maximize precision of depth buffering.
  2. The projection matrix in most graphics systems also scales the $x$ and $y$ values according to the field of view of the camera.
- We'll get into these details in Chapter 10, when we show what a projection matrix looks like in practice in DirectX and OpenGL.
That concludes Chapter 6. Next, Chapter 7: Polar Coordinate Systems