3 – 3D Math Foundations

3D COORDINATE SYSTEMS

Points can be represented in homogeneous form:
\[ P = [x \ y \ z \ 1] \]

GLSL – vec4
GraphicsLib3D – Point3D

Matrices

Column-major form:
\[
\begin{bmatrix}
A00 & A10 & A20 & A30 \\
A01 & A11 & A21 & A31 \\
A02 & A12 & A22 & A32 \\
A03 & A13 & A23 & A33
\end{bmatrix}
\]

GraphicsLib3D – Matrix3D
GLSL – mat4

Identity matrix

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

GraphicsLib3D – Matrix3D.setToIdentity()

3D TRANSFORMATIONS

Translation:
\[
\begin{bmatrix}
x+T_x \\
y+T_y \\
z+T_z \\
1
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & T_x \\
0 & 1 & 0 & T_y \\
0 & 0 & 1 & T_z \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

GraphicsLib3D – Matrix3D.translate(x,y,z)

Scaling:
\[
\begin{bmatrix}
x*S_x \\
y*S_y \\
z*S_z \\
1
\end{bmatrix} = \begin{bmatrix}
S_x & 0 & 0 & 0 \\
0 & S_y & 0 & 0 \\
0 & 0 & S_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

GraphicsLib3D – Matrix3D.scale(double sx, double sy, double sz)
Scaling To Change Systems

Consider a point in a Right-Handed system. What are its coordinates in a LH system aligned at the same origin?

\[
P_{\text{RH}} = (6, 4, 2) \quad \Rightarrow \quad P_{\text{LH}} = (?, ?, ?)
\]

Scale (RHS, LHS):

\[
\begin{pmatrix}
6 \\
4 \\
-2 \\
1
\end{pmatrix}
= 
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
6 \\
4 \\
2 \\
1
\end{pmatrix}
\]

3D ROTATION TRANSFORMS

Rotation about X by \( \theta \):

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \theta & -\sin \theta & 0 \\
0 & \sin \theta & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

GraphicsLib3D – Matrix3D.rotateX(degrees)

Recall 2D rotations can be about any point. Similarly, 3D rotations can be about any line. (axis of rotation) :

Rotation about Y by \( \theta \):

\[
\begin{pmatrix}
\cos \theta & 0 & \sin \theta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \theta & 0 & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

GraphicsLib3D – Matrix3D.rotateY(degrees)

Rotation about Z by \( \theta \):

\[
\begin{pmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

GraphicsLib3D – Matrix3D.rotateZ(degrees)
**EULER’S THEOREM**

“Any sequence of rotations about a point is equivalent to a single rotation about some axis through that point.”

[Leonard Euler, 1707–1783]

Thus, rotation about a line (through the origin) can be done with a set of rotations about the X, Y, and Z axes:

1. Translate the axis so it goes through the origin
2. Rotate by appropriate “Euler” angles about X, Y, Z
3. “Undo” the translation

**Visualizing Euler’s Theorem**

**3D VECTORS**

Like 2D vectors, 3D vectors can be

- Represented as the difference of two (3D) points
- Translated without changing their value
- Represented as a single point (a vector from the origin)

\[ V = (P_2 - P_1) = (P_4 - P_3) = (P_4 - (0,0,0)) = [x, y, z] \]

**Vector operations**

**Addition:**

\[ A = [u, v, w] \]
\[ B = [x, y, z] \]
\[ C = A + B = [u+x, v+y, w+z] \]

**GraphicsLib3D – Vector3D.add(Vector3D)**

**Length:**

\[ A = [u, v, w] \]
\[ B = \text{length}(A) = \sqrt{u^2 + v^2 + w^2} \]

**GraphicsLib3D – Vector3D.magnitude()**

**Reflection, Refraction – described in textbook.**

(not in GraphicsLib3D)

**DOT PRODUCT**

Given vectors \( V [x,y,z] \) and \( W [x,y,z] \), their “dot product” is

\[ \hat{V} \cdot \hat{W} = \sum_{i=1}^{3} V_i \cdot W_i = (V_x \cdot W_x) + (V_y \cdot W_y) + (V_z \cdot W_z) \]

**Uses For The Dot Product**

**Angle between vectors:**

\[ \hat{V} \cdot \hat{W} = \hat{V} \cdot \hat{W} \cdot \cos(\theta) \]

**Which means:**

\[ \cos(\theta) = \frac{\hat{V} \cdot \hat{W}}{\|\hat{V}\| \cdot \|\hat{W}\|} \]

and if \( V \) and \( W \) are normalized (unit length):

\[ \cos(\theta) = \hat{V} \cdot \hat{W} \]
Uses For The Dot Product (cont.)

Vector magnitude: squared magnitude = dot product of a vector with itself
\[ \vec{a} \cdot \vec{a} = |\vec{a}| \cdot |\vec{a}| \cdot \cos(\theta) \]
\[ = |\vec{a}| \cdot |\vec{a}| \cdot \cos(0) = |\vec{a}| \cdot |\vec{a}| + 1 \]
\[ = |\vec{a}| \]
\[ |\vec{a}| = \sqrt{\vec{a} \cdot \vec{a}} \]

Uses For The Dot Product (cont.)

Relationship of a point to a line:
Which side of line \( P_0 - P_1 \) is \( P_2 \) on?
\[ \vec{a} = P_1 - P_0 \]
\[ \vec{b} = P_2 - P_1 \]
signOf( \( \vec{a} \cdot \vec{b} \) )

Uses For The Dot Product (cont.)

- **Collinear:**
  \[ \vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| = ab \quad (because \ \cos(0) = 1) \]

- **Collinear but opposite directions:**
  \[ \vec{a} \cdot \vec{b} = -ab \quad (because \ \cos(180) = -1) \]

- **Perpendicular:**
  \[ \vec{a} \cdot \vec{b} = 0 \quad (because \ \cos(90) = 0) \]

- **“Same direction”:**
  \[ \vec{a} \cdot \vec{b} > 0 \quad (because \ \cos(-90..90) > 0) \]

Cross Products

Another operation on 3D vectors is:
\[ \vec{V} \times \vec{W} = \vec{R} \]
\[ \text{R is a vector } [x \ y \ z], \text{ where} \]
\[ \vec{R}_i = (\vec{V}_{i\oplus 1} \times \vec{W}_{i\oplus 1}) - (\vec{V}_{i\oplus 0} \times \vec{W}_{i\oplus 0}) \]

\[ i \oplus 0 = \{i \text{ mod } 3\} + 1, \]
\[ i \oplus 1 = \{i - 1\} - 0.7 \cdot \{i - 1\} \]

Cross Products (cont.)

Said another way:
\[ R_x = [V_y \cdot W_z] - [V_z \cdot W_y] \]
\[ R_y = [V_z \cdot W_x] - [V_x \cdot W_z] \]
\[ R_z = [V_x \cdot W_y] - [V_y \cdot W_x] \]

Visually:
\[ \vec{V} = [x \ y \ z] \]
\[ \vec{W} = [x \ y \ z] \]
\[ R_i = [V_{i\oplus 0} \cdot W_{i\oplus 0}] - [V_{i\oplus 1} \cdot W_{i\oplus 1}] \]

Uses For The Cross Product

- **Note that any two 3D vectors define a plane:**

![Diagram of cross product](image)
Cross Product Uses (cont.)

- An important property of \( \vec{V} \times \vec{W} \): perpendicular to the plane:

\[ \vec{R} = \vec{V} \times \vec{W} \]

\[ \vec{V} \quad \vec{W} \]

Cross Product Uses (cont.)

- Cross product vector direction: “right-hand rule”

\[ \vec{R} = \vec{V} \times \vec{W} \]

Cross Product Uses (cont.)

We sometimes need to find “outward normals”

\[ \vec{N} = \vec{P} \times \vec{Q} \]

Vertices must be properly ordered
- what happens if not CCW?

\[ \vec{P} = \vec{V}_2 - \vec{V}_1 \]
\[ \vec{Q} = \vec{V}_3 - \vec{V}_1 \]