Transformations & Matrices

Introduction to Computer Graphics
CS 351-50

Quiz #2 Review

• Dot Product
  \[ a \cdot b = |a| |b| \cos \theta = a_x b_x + a_y b_y + a_z b_z \]
  \[ a \cdot b = a^T b \]
  thus
  \[ [a_x \ a_y \ a_z] \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = a_x b_x + a_y b_y + a_z b_z \]

Quiz #2 Review

• Cross Product & Normals
  • Points to Vectors:
    \[ \overrightarrow{P_{\text{tip}}} - \overrightarrow{P_{\text{tail}}} \]
    – Point ordering for triangles
    \[ \mathbf{N} = (P_2 - P_1) \times (P_3 - P_2) \]
    \[ = (P_1 - P_2) \times (P_3 - P_1) \]
    \[ = (P_1 - P_3) \times (P_2 - P_1) \]
    – OpenGL assumes Counter-Clockwise Order

Example:  Change the time on the clock

Example:  Move clock hands
Clock Transformations

- Translate to Origin
- Move hand with rotation
- Move hand back to clock
- Do other hand

Clock Transformations

\[ M = T(a, b) \cdot R(t) \cdot T(-a, -b) \]

Translations

\[ T = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \end{bmatrix}, \quad T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \end{bmatrix} \]

Scale

\[ S = \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \end{bmatrix}, \quad S^{-1} = \begin{bmatrix} 1/x & 0 & 0 & 0 \\ 0 & 1/y & 0 & 0 \\ 0 & 0 & 1/z & 0 \end{bmatrix} \]
## Rotations

Rotate(q, 1, 0, 0) =
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \theta & -\sin \theta & 0 \\
0 & \sin \theta & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Rotate(q, 0, 1, 0) =
\[
\begin{bmatrix}
\cos \theta & 0 & \sin \theta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \theta & 0 & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Rotate(q, 0, 0, 1) =
\[
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

## 3D Positive Rotations

![3D Positive Rotations Diagram](image)

## More on 2D Transformations

Think of a bunny (one we are drawing to a screen)

- Where is she?
- Where is she facing?
- How big is she?
- Think of drawing this bunny’s movement

## Encode state in Matrix

Bunny at Initial State
- Position = 0,0
- Facing = up (Y)

Temporary matrix:
- TempM1, TempM2 = identity

SaveViewMat = identity
CurrentViewMatrix = identity

## Example: Move the bunny

Want to move bunny to:
- (50,100)
- facing 30 degrees to left

- Two good ways to think about how to get there
  - User’s point of view
    - (world coordinates)
  - From the bunny’s point of view
    - (object coordinates)
  - Bunny follows directions we give her
Example: Move the bunny
- User’s point of view
  - (world coordinates)
    - Rotate bunny around (0,0) until she is facing 30 degrees (θ = π/3) to left, then Translate (0,0) to (50,100)

\[
\begin{bmatrix}
1 & 0 & 50 \\
0 & 1 & 100 \\
0 & 0 & 1 \\
\end{bmatrix} \begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta & 50 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} = T_{(50,100)}R_{\theta}
\]

From the point of view of bunny
- Give her directions
  - Walk from (0,0) to (50,100)
  - Rotate 30 degrees to left (θ = π/3)

\[
\begin{bmatrix}
1 & 0 & 50 \\
0 & 1 & 100 \\
0 & 0 & 1 \\
\end{bmatrix} \begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta & 50 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} = T_{(50,100)}R_{\theta}
\]

Example Code: Movement 1
```c
MyVec4f new_origin(0,0,0,1);
float init_rot = 0;
new_origin.w(1); new_origin.z(0);
new_origin.y(30);
new_origin.x(20);
load4DMatrix(0);

if (DEBUG_PRINT) {
    tempMat2.print("tempMat2 = current Matrix");
    tempMat2 = tempMat2.rotate_z(init_rot);
    tempMat2.print("tempMat1  = Translate (new_origin) ");
    tempMat1 = Translate (new_origin);
    tempMat1.print("tempMat1 Identity ");
    tempMat1.identity();
    tempMat1.print("current = tempMat1 * tempMat2
    ");
}
```

Works same for any geometry
- User’s point of view
  - (world coordinates)
    - Rotate around (0,0) facing 30 degrees (θ = π/3) to left, then Translate (0,0) to (50,100)

\[
\begin{bmatrix}
1 & 0 & 50 \\
0 & 1 & 100 \\
0 & 0 & 1 \\
\end{bmatrix} \begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta & 50 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} = T_{(50,100)}R_{\theta}
\]

Note: write down matrix in opposite order in which they are applied from the user’s point of view

Example Code: Movement 2
```c
MyVec4f new_origin.x(50); new_origin.y(100);
new_origin.z(0);
new_origin.w(1);
load4DMatrix(new_origin);
myMat1 = new_origin.w(1); myMat2 = new_origin.z(0);
myMat3 = new_origin.y(30); myMat4 = new_origin.x(20);
load4DMatrix(myMat1);
load4DMatrix(myMat2);
load4DMatrix(myMat3);
load4DMatrix(myMat4);
load4DMatrix(currentMat);

tempMat1 = Translate (new_origin);
tempMat2 = currentMat;
tempMat2.print("tempMat2 Rotate by 0\(^\circ\) ");
tempMat2 = tempMat2.rotate_z(30);
tempMat2.print("tempMat2 Rotate by 30\(^\circ\)");
tempMat2.print("tempMat2 Rotate by 0\(^\circ\) ");

current = tempMat1 * tempMat2;

drawBunnyPoly();
```

From the point of view of bunny
- Move her another hop:
  - Note: Post multiply M by another transformation matrix

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & 50 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta & 50 & 100 \\
\sin \theta & \cos \theta & 30 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} = T_{(50,100)}R_{\theta}
\]

Example: bunny’s point of view
- Give her directions
  - Walk from (0,0) to (50,100)
  - Rotate 30 degrees to left (θ = π/3)
Example Code: Movement 3

```c
float new_rot = M_PI/6;
tempMat1.identity();
tempMat1.rot_z(new_rot);
tempMat1.print("tempMat1 = rotate by PI/6");
tempMat2 = currentMat;
tempMat2.print("tempMat2 = current Matrix");
currentMat = tempMat2*tempMat1;
currentMat.print("current = temp2*temp1");
load4DMatrix(currentMat);
drawBunnyPoly();
```

Example Code: Movement 4

```c
new_origin.x(30);
tempMat1.identity();
tempMat1.setTranslation(new_origin);
tempMat1.print("tempMat1 = Translate by 30,0,0,1");
tempMat2 = currentMat;
tempMat2.print("tempMat2 = currentMat");
currentMat = tempMat2*tempMat1;
if (DEBUG_PRINT) {printf("current =temp1 * temp2
");
currentMat.print();}
load4DMatrix(currentMat);
drawBunnyPoly();
```

Matrix & 2D Transforms

```c
gMatrixMode(GL_MODELVIEW);
gLoadIdentity();
gMultMatrixf(N); /* apply transformation N */
gMultMatrixf(M); /* apply transformation M */
gMultMatrixf(L); /* apply transformation L */
gBegin(GL_POINTS);
gVertex3fv(v); /* draw transformed vertex v */
gEnd();
```

Result: \( I N M L = NML \)

Summary

- Either think in terms of grand, fixed coordinate system
  - Order of operations is the opposite of order of how they appear in the code
  - Hard to do…

Summary

- Instead, think of a local coordinate system tied to the object you are drawing
  - Operations occur relative to this changing coordinate system
  - Matrix multiplications happen in natural order in the code
- In the end, the code is the same, you just have to think about it differently
- Read Chapter 3 of OpenGL book (especially the "Viewing and Modeling transformations" section)
Orthographic Viewing Matrix

// create a viewing volume, see pg 124 of
// OGL Programming book (Version 1.1)
// note: numbers should be proportional to window size

```c
void glOrtho(float left, float right, float bottom, float top, float near, float far);
```

thus the center in screen x is 0, bottom in y is 0, top is 80;

---

L-systems

Handouts

http://ai.toastbrot.ch/life/intro.php