Interactive Game Loop
Interactive Game Loop

Game Input (Keyboard, Mouse) → Physics Strategy, AI → Render Triangles (one frame)

30 to 60 frames per second

Exit?

Cleanup States
3D Graphics Rendering Pipeline

- **Geometry Pipeline**
  - Processing Vertices
  - Mainly *floating-point* operations

- **Rasterization Pipeline**
  - Processing Pixels
  - Mainly dealing with *integer* operations
3D Graphics Rendering Pipeline

- Geometry Pipeline
  - Processing Vertices
  - Mainly floating-point operations

- Rasterization Pipeline
  - Processing Pixels
  - Mainly dealing with Integer operations
  - MMX was originally designed to accelerate this type of functionality
3D Graphics Rendering Pipeline

- Geometry Pipeline
  - Processing Vertices
  - Mainly floating-point operations
  - SSE/SSE2 were designed for this part
- Rasterization Pipeline
  - Processing Pixels
  - Mainly dealing with Integer operations
  - MMX was originally designed to accelerate this type of functionality
Fixed Function 3D Graphics Pipeline

- **Geometry Pipeline**
  - Processing Vertices
  - Mainly floating-point operations
  - SSE/SSE2 were designed for this part

- **Rasterization Pipeline**
  - Processing Pixels
  - Mainly dealing with Integer operations
  - MMX was originally designed to accelerate this functionality
3D Coordinates

- LHS was more common for graphics viewing space
- XNA uses RHS, however
- Camera or viewer position can be set up using 3D API
Geometry Format — Vertex Coordinates

(X1, Y1, Z1)
(X2, Y2, Z2)
(X3, Y3, Z3)
Geometry Format — Vertex Normals

+Y

+X

+Z

(NX1, NY1, NZ1)

(NX2, NY2, NZ2)

(NX3, NY3, NZ3)
Geometry Format — Vertex Colors

(R1, G1, B1, A1)

(R2, G2, B2, A2)

(R3, B3, B3, G3)

+X

+Y

+Z
Triangle-based Geometry Representation

Triangle List
(note the vertex order)

Triangle Strip

Triangle Fan
Specifying a 3D object

- Vertex ordering is critical when culling mode enabled
- Direct3D and XNA “default”s are counter-clockwise culling
  - Weird default setting for an RHS
Specifying a 3D object

- Vertex ordering is critical when culling mode enabled
- We will discuss normal computation later in this lecture

Triangle list
\{v1, v2, v7\},
\{v2, v8, v7\},
\{v2, v3, v4\},
\{v2, v4, v8\},
\{v4, v7, v8\},
\{v4, v6, v7\}

Triangle strip
\{v1, v2, v7, v8\},
\{v3, v4, v2, v8\},
\{v6, v7, v4, v8\}
3D Rendering Pipeline

- World Transform
- View Transform
- Lighting
- Projection Transform
- Backface Culling
- Clipping
- Perspective Divide
- Viewport Transform
- Rasterization
Transformation Pipeline

• World Transformation
  – Model coordinates $\rightarrow$ World coordinates

• View Transformation
  – World coordinates $\rightarrow$ Camera space

• Projection Transformation
  – Camera space $\rightarrow$ View Plane

• These are a series of matrix multiplications
World Transformation

World Coordinates

World origin

+y

Local model coordinates

+Z

Local model coordinates

+X

- Translation
- Rotation
- Scaling
View Transformation

World Coordinates

+Y

+Z

+X

World origin

- Camera position
- Look vector
Projection Transformation

• Set up camera internals

• Set up
  – Field of View (FOV)
  – View frustum
  – View planes

• Will discuss later
Homogeneous Coordinates

• Enable all transformations to be done by “multiplication”
  – Primarily for translation (See next few slides)

• Add one coordinate (w) to a 3D vector

• Each vertex has [x, y, z, w]
  – W will be useful for perspective projection
  – W should be 1 in a Cartesian Coordinate System
Transformation 1: Translation (Offset)
Translation Matrix

\[
[x_t, y_t, z_t, 1] = [x, y, z, 1] \cdot \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
T_x & T_y & T_z & 1
\end{bmatrix}
\]
Transformation 2: Scaling
Scaling Matrix

\[
\begin{bmatrix}
S_x & 0 & 0 & 0 \\
0 & S_y & 0 & 0 \\
0 & 0 & S_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[\begin{bmatrix}
x_s, y_s, z_s, 1\end{bmatrix} = \begin{bmatrix}x, y, z, 1\end{bmatrix} \cdot \]

\[
\begin{bmatrix}
S_x & 0 & 0 & 0 \\
0 & S_y & 0 & 0 \\
0 & 0 & S_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Transformation 3: Rotation
2D Rotation

\[ [x', y', 1] = [x, y, 1] \cdot \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

Rotate along which axis?
3D Rotation Matrix

Rotation along $\mathbf{Z}$ axis: $[x', y', z', 1] = [x, y, z, 1] \cdot \begin{bmatrix}
\cos \theta & \sin \theta & 0 & 0 \\
-\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$

Rotation along $\mathbf{Y}$ axis: $[x', y', z', 1] = [x, y, z, 1] \cdot \begin{bmatrix}
\cos \theta & 0 & -\sin \theta & 0 \\
0 & 1 & 0 & 0 \\
\sin \theta & 0 & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$

Rotation along $\mathbf{X}$ axis: $[x', y', z', 1] = [x, y, z, 1] \cdot \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \theta & \sin \theta & 0 \\
0 & -\sin \theta & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$
Non-Commutative Property (1)

1. Counter-clockwise 90° along y
2. Clockwise 90° along x

1. Clockwise 90° along x
2. Counter-clockwise 90° along y
Non-Commutative Property (1)

\[
\begin{vmatrix}
\begin{array}{ccc}
\cos(\frac{-\pi}{2}) & 0 & -\sin(\frac{-\pi}{2}) \\
0 & 1 & 0 \\
\sin(\frac{-\pi}{2}) & 0 & \cos(\frac{-\pi}{2})
\end{array}
\end{vmatrix}
\times
\begin{vmatrix}
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
0 & \cos(\frac{\pi}{2}) & \sin(\frac{\pi}{2}) & 0 \\
0 & -\sin(\frac{\pi}{2}) & \cos(\frac{\pi}{2}) & 0 \\
0 & 0 & 0 & 1
\end{array}
\end{vmatrix}
= \begin{vmatrix}
0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
-1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{vmatrix}
\]

\[(x'', y'', z'') = (-z, -x, y)\]

\[
\begin{vmatrix}
\begin{array}{ccc}
1 & 0 & 0 \\
0 & \cos(\frac{\pi}{2}) & \sin(\frac{\pi}{2}) \\
0 & -\sin(\frac{\pi}{2}) & \cos(\frac{\pi}{2})
\end{array}
\end{vmatrix}
\times
\begin{vmatrix}
\begin{array}{cccc}
\cos(\frac{-\pi}{2}) & 0 & -\sin(\frac{-\pi}{2}) \\
0 & 1 & 0 \\
\sin(\frac{-\pi}{2}) & 0 & \cos(\frac{-\pi}{2})
\end{array}
\end{vmatrix}
= \begin{vmatrix}
0 & 0 & 1 & 0 \\
-1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{vmatrix}
\]

\[(x'', y'', z'') = (-y, -z, x)\]
Non-Commutative Property (2)

1. Translation by \((x, y, z)\)
2. Scale by 2 times

1. Scale by 2 times
2. Translation by \((x, y, z)\)
Non-Commutative Property (2)

\[(x'', y'', z'') = (x'Sx+Sx'Tx, y'Sy+Sy'Ty, z'Sz+Sz'Tz)\]

Offsets were scaled as well

\[(x'', y'', z'') = (x'Sx+Tx, y'Sy+Ty, z'Sz+Tz)\]
Non-Commutative Property

• Ordering matters!

• Be careful when performing matrix multiplication