Interactive Game Loop

- Game Input (Keyboard, Mouse)
- Physics Strategy, AI
- Render Triangles (one frame)
- Exit?

Cleanup States

30 to 60 frames per second

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

### 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 type of 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

Geometry Format — Vertex Normals

Geometry Format — Vertex Colors

Triangle-based Geometry Representation
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

Triangle list
{v1, v3, v2},
{v1, v5, v3},
{v5, v6, v3},
{v4, v3, v6},
{v1, v7, v6},
{v1, v6, v5}

Triangle strip
{v5, v3, v1, v2},
{v5, v6, v3, v4},
{v7, v6, v1, v5}

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 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

Transformation Pipeline
World Transformation

- World Coordinates
- Local model coordinates
- World origin
- +x, +y, +z
- Translation
- Rotation
- Scaling

View Transformation

- World Coordinates
- World origin
- +x, +y, +z
- 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)

\[(x, y, z) \rightarrow (x_t, y_t, z_t)\]

Transformation Matrix

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
T_x & T_y & T_z & 1
\end{bmatrix}
\]

Transformation 2: Scaling

Scaling Matrix

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

2D Rotation

3D Rotation Matrix

Non-Commutative Property (1)
Non-Commutative Property (1)

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

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

Non-Commutative Property (2)

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

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

Offsets were scaled as well

Non-Commutative Property (2)

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

Non-Commutative Property

- Ordering matters!
- Be careful when performing matrix multiplication