Start Up

- Any questions from last time?
- Go over acceleration techniques?
- Or intersection algorithms?

Slide Credits

- Leonard McMillan, Seth Teller, Fredo Durand, Barb Cutler - MIT
- G. Drew Kessler, Larry Hodges - Georgia Institute of Technology
- John Hart - University of Illinois
- Rick Parent - Ohio State University

Homogeneous Coordinates

- Add an extra dimension
  - in 2D, we use 3 x 3 matrices
  - in 3D, we use 4 x 4 matrices
- Each point has an extra value, w

\[
\begin{pmatrix}
    x' \\
    y' \\
    z' \\
    w'
\end{pmatrix} =
\begin{pmatrix}
    a & b & c & d \\
    e & f & g & h \\
    i & j & k & l \\
    m & n & o & p
\end{pmatrix}
\begin{pmatrix}
    x \\
    y \\
    z \\
    w
\end{pmatrix}
\]

\[ p' = Mp \]
Translate \((tx, ty, tz)\)

- Why bother with the extra dimension? Because now translations can be encoded in the matrix!

\[
\begin{pmatrix}
  x' \\
  y' \\
  z'
\end{pmatrix} =
\begin{pmatrix}
  1 & 0 & tx \\
  0 & 1 & ty \\
  0 & 0 & tz
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\]

Scale \((sx, sy, sz)\)

- Isotropic (uniform) scaling: \(s_x = s_y = s_z\)

\[
\begin{pmatrix}
  x'^* \\
  y'^* \\
  z'^*
\end{pmatrix} =
\begin{pmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & s_z
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\]

- You only have to implement uniform scaling

Rotation

- About z axis

\[
\begin{pmatrix}
  x' \\
  y' \\
  z'
\end{pmatrix} =
\begin{pmatrix}
  \cos \phi & -\sin \phi & 0 \\
  \sin \phi & \cos \phi & 0 \\
  0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\]

- About x axis:

\[
\begin{pmatrix}
  x' \\
  y' \\
  z'
\end{pmatrix} =
\begin{pmatrix}
  1 & 0 & 0 \\
  0 & \cos \alpha & \sin \alpha \\
  0 & \sin \alpha & \cos \alpha
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\]

- About y axis:

\[
\begin{pmatrix}
  x' \\
  y' \\
  z'
\end{pmatrix} =
\begin{pmatrix}
  \cos \theta & 0 & \sin \theta \\
  0 & 1 & 0 \\
  -\sin \theta & 0 & \cos \theta
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\]

Rotation

- About \((k_x, k_y, k_z)\), an arbitrary unit vector

(Rodrigues Formula)

\[
\begin{pmatrix}
  x' \\
  y' \\
  z'
\end{pmatrix} =
\begin{pmatrix}
  k_x(k(1-c)+c) & k_y(k(1-c)+c) & k_z(k(1-c)+c) \\
  k_y(k(1-c)+c) & k_z(k(1-c)+c) & k_x(k(1-c)+c) \\
  k_z(k(1-c)+c) & k_x(k(1-c)+c) & k_y(k(1-c)+c)
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\]

where \(c = \cos \theta\) & \(s = \sin \theta\)

How are transforms combined?

Scale then Translate

Use matrix multiplication:

\[
T = \begin{pmatrix}
1 & 0 & 3 \\
0 & 1 & 2 \\
0 & 0 & 1
\end{pmatrix}
\]

Caution: matrix multiplication is NOT commutative!
**Non-commutative Composition**

Scale then Translate: 
\[
p' = T(S\ p) = TS\ p
\]

Translate then Scale: 
\[
p' = S(T\ p) = ST\ p
\]

\[
\begin{pmatrix}
0 & 0 & 3 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
2 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
2 & 0 & 3 \\
0 & 2 & 1 \\
0 & 0 & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 & 0 & 2 \\
0 & 2 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

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

**Transformations in Modeling**

- Position objects in a scene
- Change the shape of objects
- Create multiple copies of objects
- Projection for virtual cameras
- Animations

**Scene Description**

**Simple Scene Description File**

```plaintext
Camera {
    center 0 0 10
    direction 0 0 -1
    up 0 1 0
}

Lights {
    numLights 1
    DirectionalLight {
        direction -0.5 -0.5 -1
        color 1 1 1
    }
}

Background { color 0.2 0 0.6 }

Materials {
    numMaterials <=0
    <MATERIALS>
}

Group {
    numObjects <=0
    <OBJECTS>
}
```
Hierarchical Models

- Logical organization of scene

Simple Example with Groups

- Adding Materials

- Adding Transformations

Using Transformations

- Position the logical groupings of objects within the scene

- Transformation in group

Directed Acyclic Graph is more efficient and useful
Processing Model Transformations

- Goal
  - Get everything into world coordinates
  - Traverse graph/tree in depth-first order
  - Concatenate transformations
  - Can store intermediate transformations
  - Apply/associate final transformation to primitive at leaf node
- What about cylinders, superquadrics, etc.?
  - Transform ray!

Transform the Ray

- Map the ray from \textit{World Space} to \textit{Object Space}

\[ p_{WS} = M \quad p_{OS} \]
\[ p_{OS} = M^{-1} \quad p_{WS} \]

Transform Ray

- New origin:
  \[ \text{origin}_{OS} = M^{-1} \quad \text{origin}_{WS} \]

- New direction:
  \[ \text{direction}_{OS} = M^{-1} \quad \text{direction}_{WS} \]

Transforming Points & Directions

- Transform point
  \[ \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \]

- Transform direction
  \[ \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \]

- Map intersection point and normal back to world coordinates

Transform Normals

- Why? They’re used for shading

- Object color only
- Diffuse Shading

Transforming Normals

- A surface normal is a \textit{property}, not a \textit{geometric entity}

- Correct normal transformation matrix:
  \[ A = (M^{-1})^T \]
Constructive Solid Geometry (CSG)

Given overlapping shapes A and B:

- **Union**
- **Intersection**
- **Subtraction**

Collect all the intersections

Implementing CSG

1. Test "inside" intersections:
   - Find intersections with A, test if they are inside/outside B
   - Find intersections with B, test if they are inside/outside A

2. Overlapping intervals:
   - Find the intervals of "inside" along the ray for A and B
   - Compute union/intersection/subtraction of the intervals

How can we implement CSG?

Points on A, Outside of B
Points on A, Inside of B
Points on B, Outside of A
Points on B, Inside of B

Seminal paper

Ray tracing CSG models

CSG
- Form object as booleans of primitive objects
  - Primitives: sphere, cube, cylinder, cone
  - Boolean operators: union, intersection, difference
- Tree structure used to manage operations
  - Leaf nodes are primitive objects
  - Intermediate nodes specify combination operator

Ray tracing CSG models
- Intersect ray with primitives
- Produces “spans” along ray
- Perform Boolean operations on spans
- Determines intersection of evaluated model
- Calculate normal at intersection

Union
- Min (t_C min, t_B min)
  - Ray intersects union: at first intersection

Intersection
- ++
  - If ((t_C min < t_B min) and (t_C max > t_B min)): t_B min
  - Else If ((t_B min < t_C min) and (t_B max > t_C min)): t_C min
  - Else: none

Difference
- -
  - If ((t_B min < t_C min) and (t_B max > t_C max)): t_B min
  - Else if (t_B max < t_C min): t_C max
  - Else: none

Possible ways for 2 spans to overlap
Difference

First time in C not in B
If \( t_{C_{\text{min}}} < t_{B_{\text{min}}} \): \( t_{C_{\text{min}}} \)
Else if \( t_{B_{\text{max}}} < t_{C_{\text{max}}} \): \( t_{B_{\text{max}}} \)
Else: none

Primitives

Anything that can be intersected (easily) with a ray

Conics: solve analytically using \( R(t) \)
Convex polyhedra
A plane (a cutting plane is useful)

can be used as a modeling tool (boolean operations)
surface model (e.g., polyhedron) computed from CGS
or
Can be used as a model representation
keep tree structure and ray trace directly

Controlling the Combinations

Tree Structure

Tree Structure #1
Tree Structure

- Intersect ray with leaf nodes (primitive objects)
- Combine intersection spans according to intermediate nodes
  - union
  - intersection
  - difference
- Might create multiple spans

Union of Spans

Intersection of Spans

Difference of Spans

Normals of CSG intersections

Normal of some surface (or its negation)

Union or intersection:
positive normal of intersected surface
Difference normals

- Intersection is one of:
  - $t_{\text{min}}$ of positive object – normal of surface
  - $t_{\text{max}}$ of negative object – negated normal

Add transformations to tree

Add transformations to tree

Bounding Volumes

- Construction:
  - Use bounding volumes at leaf nodes
  - Union bounding volumes at interior nodes

- Traversal:
  - Top-down
  - Test bounding volume at interior

Example

Example

Example

Example
For example:

- Discuss status/problems/issues with this week’s programming assignment