CS 586/480
Computer Graphics II

Dr. David Breen
Matheson 408
Thursday 6PM - 8:50PM

Presentation 2
10/7/04

Start Up
- Any questions from last time?
- Presentation assignments
  - Shailaja
  - Ilya
  - Dmitry
- Programming assignment submissions
- Go over sampling image plane?

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

More Geometry & Intersections
Ray/Plane Intersection

Ray is defined by \( R(t) = R_o + R_d \cdot t \) where \( t \geq 0 \)
- \( R_o \) = Origin of ray at \((x_o, y_o, z_o)\)
- \( R_d \) = Direction of ray \([x_d, y_d, z_d]\) (unit vector)

Plane is defined by \([A, B, C, D]\)
- \( Ax + By + Cz + D = 0 \) for a point in the plane
- Normal Vector, \( N = [A, B, C] \) (unit vector)
- \( A^2 + B^2 + C^2 = 1 \)
- \( D = -N \cdot P_0 \) (\( P_0 \) - point in plane)

Ray/Plane (cont.)

Substitute the ray equation into the plane equation:
\[
A(x_o + x_d t) + B(y_o + y_d t) + C(z_o + z_d t) + D = 0
\]

Solve for \( t \):
\[
t = \frac{-A(x_o + \frac{D}{N}) + B(y_o + \frac{D}{N}) + C(z_o + \frac{D}{N})}{A x_d + B y_d + C z_d}
\]

What Can Happen?

1. Calculate \( N \cdot R_d \) and compare it to zero.
2. Calculate \( t_i \) and compare it to zero.
3. Compute intersection point.
4. Flip normal if \( N \cdot R_d \) is positive

Ray/Plane Summary

Intersection point:
\[
(x, y, z) = (x_o + x_d t_i, y_o + y_d t_i, z_o + z_d t_i)
\]
Ray-Parallelepiped Intersection
- Axis-aligned
- From $(X_1, Y_1, Z_1)$ to $(X_2, Y_2, Z_2)$
- Ray $P(t)=R_o+R_d t$

Naïve ray-box Intersection
- Use 6 plane equations
- Compute all 6 intersection
- Check that points are inside box $Ax+By+Cz+D \leq 0$

Factoring out computation
- Pairs of planes have the same normal
- Normals have only one non-0 component
- Do computations one dimension at a time
- Maintain $t_{near}$ and $t_{far}$ (closest and farthest so far)

Test if parallel
- If $R_o x = 0$, then ray is parallel
- If $R_o x < X_1$ or $R_o x > x_2$ return false
If not parallel

- Calculate intersection distance $t_1$ and $t_2$
  - $t_1 = (X_1 - R_{ox})/R_{dx}$
  - $t_2 = (X_2 - R_{ox})/R_{dx}$

Test 1

- Maintain $t_{near}$ and $t_{far}$
  - If $t_1 > t_2$, swap
  - If $t_1 > t_{near}$, $t_{near} = t_1$
  - If $t_2 < t_{far}$, $t_{far} = t_2$
- If $t_{near} > t_{far}$, box is missed

Test 2

- If $t_{far} < 0$, box is behind

Algorithm recap

- Do for all 3 axes
  - Calculate intersection distance $t_1$ and $t_2$
  - Maintain $t_{near}$ and $t_{far}$
  - If $t_{near} > t_2$, box is missed
  - If $t_{far} < 0$, box is behind
- If box survived tests, report intersection at $t_{near}$
Ray/Ellipsoid Intersection

Ellipsoid's surface is defined by the set of points \( \{(x_s, y_s, z_s)\} \) satisfying the equation:
\[
\frac{(x_s/a)^2}{2} + \frac{(y_s/b)^2}{2} + \frac{(z_s/c)^2}{2} - 1 = 0
\]

Centers at origin

Ray/Cylinder Intersection

Cylinder's surface is defined by the set of points \( \{(x_s, y_s, z_s)\} \) satisfying the equation:
\[
(x_s)^2 + (y_s)^2 - r^2 = 0
- Z_0 \leq z_s \leq Z_0
\]

Ray/Ellipsoid Intersection

- Substitute ray equation into surface equations
- This is a quadratic equation in \( t \):
  - \( At^2 + Bt + C = 0 \)
  - Analyze as before
  - Solve for \( t \) with quadratic equation
  - Plug \( t \) back into ray equation - Done
  - Well,... not exactly

Ray/Cylinder Intersection

- Is intersection point \( P_i \) between \(-Z_0\) and \( Z_0\)?
- If not, \( P_i \) is not valid
- Also need to do intersection test with \( z = -Z_0 \) \( | Z_0 \) plane
- If \( (P_x)^2 + (P_y)^2 \leq r^2 \), you've intersected a "cap"
- Which valid intersection is closer?

Torus

- Product of two implicit circles
  \[
  (x - R)^2 + (y - R)^2 = 0
  (x + R)^2 + (y + R)^2 = 0
  \]

- Surface of rotation: replace \( z \) with \( z^2 + y^2 \)
  \[
  f(x, y, z) = (x^2 + y^2 + z^2 - R^2)^2 - 4R^2x^2 - 4R^2y^2
  \]

- Quartic!!!
- Up to four ray torus intersections
Superquadrics

\[\begin{align*}
\mathbf{x}(\rho, \phi, \eta) &= \left[ a_1 \cos^\rho \eta \cos^\theta \phi, \\
&\quad a_2 \cos^\rho \eta \sin^\theta \phi, \\
&\quad a_3 \sin^\rho \eta \right] \\
&\quad -\pi \leq \phi \leq \pi, \\
&\quad -\rho \leq \rho \leq \rho
\end{align*}\]

SMF Triangle Meshes

- \(v\) vertices
- \(f\) triangles

<table>
<thead>
<tr>
<th>vertices</th>
<th>(v)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>1 2 3</td>
</tr>
<tr>
<td>1 -1 -1</td>
<td>1 -1 1</td>
</tr>
<tr>
<td>1 1 -1</td>
<td>1 -1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>triangles</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td>1 3 4</td>
</tr>
<tr>
<td>2 4 6</td>
<td>1 6 2</td>
</tr>
<tr>
<td>3 4 6</td>
<td>2 4 6</td>
</tr>
</tbody>
</table>

Draw data structure

Triangle Meshes (.iv)

Transformations & Hierarchical Models
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, $w$

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

$$p' = Mp$$

Homogeneous Coordinates

- Most of the time $w = 1$, and we can ignore it

$$\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix} =
\begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}$$

Translate $(tx, ty, tz)$

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

$$\begin{bmatrix}
  x' \\
  y' \\
  z'
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & 0 & t_x \\
  0 & 1 & 0 & t_y \\
  0 & 0 & 1 & t_z \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}$$

Scale $(sx, sy, sz)$

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

$$\begin{bmatrix}
  x' \\
  y' \\
  z'
\end{bmatrix} =
\begin{bmatrix}
  s_x & 0 & 0 & 0 \\
  0 & s_y & 0 & 0 \\
  0 & 0 & s_z & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}$$

- You only have to implement uniform scaling
Rotation

About z axis

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

Rotation

About x axis:

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

Rotation

About y axis:

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

Rotation

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

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} = \begin{bmatrix}
kk_x(1-c) + c & kk_x(1-c) - ks & kk_x(1-c) + ks & 0 \\
kk_y(1-c) + ks & kk_y(1-c) + c & kk_y(1-c) - ks & 0 \\
kk_z(1-c) - ks & kk_z(1-c) + c & kk_z(1-c) - ks & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

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

How are transforms combined?

Scale then Translate

Translate \((0,0)\) to \((5,3)\)

Scale \((2,2)\)

Translate \((3,1)\)

Use matrix multiplication:

\[p' = T(Sp) = (TS)p\]

\[
TS = \begin{bmatrix}
1 & 0 & 3 & 2 & 0 & 0 & 2 & 0 & 3 \\
0 & 1 & 1 & 0 & 2 & 0 & 2 & 4 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

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

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

Translate then Scale: \[ p' = S(Tp) = ST p \]

Transformations in Ray Tracing

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

Transformations in Modeling

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

\[
\begin{pmatrix}
2 & 0 & 0 & 1 & 0 & 3 & 2 & 0 & 6 \\
0 & 2 & 0 & 0 & 1 & 1 & 0 & 2 & 2 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}
\]
**Scene Description**

- **Scene**
  - Camera
  - Lights
  - Background
  - Materials
  - Objects

**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 <n>
  <MATERIALS>
}

Group {
  numObjects <n>
  <OBJECTS>
}
```

**Hierarchical Models**

- Logical organization of scene

**Simple Example with Groups**

```plaintext
Group {
  numObjects 3
  Group {
    numObjects 3
    Box { <BOX PARAMS> }
    Box { <BOX PARAMS> }
    Box { <BOX PARAMS> }
  }
  Group {
    numObjects 2
    Box { <BOX PARAMS> }
    Box { <BOX PARAMS> }
    Box { <BOX PARAMS> }
  }
  Box { <BOX PARAMS> }
  Sphere { <SPHERE PARAMS> }
  Sphere { <SPHERE PARAMS> }
}
Plane { <PLANE PARAMS> }
```
Adding Materials

Group :
  numObjects 3
  Group :
    numObjects 3
    Box { <BOX PARAMS> }
    Box { <BOX PARAM> }
    Box { <BOX PARAM> }
  Group :
    numObjects 2
    Group :
      Box { <BOX PARAM> }
      Box { <BOX PARAM> }
    Group :
      Box { <BOX PARAM> }
      Sphere { <SPHERE PARAM> }
      Sphere { <SPHERE PARAM> }
  Plane { <PLANE PARAM> }

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 **World Space** to **Object Space**

\[
P_{WS} = M \cdot P_{OS}
\]

\[
P_{OS} = M^{-1} \cdot P_{WS}
\]

Transforming Points & Directions

- Transform point

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix} =
\begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]

- Transform direction

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix} =
\begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
\]

- Map intersection point and normal back to world coordinates

Why is the Normal important?

- It’s used for shading
  - makes things look 3D!

object color only
( Assignment 1)

Diffuse Shading
( Assignment 3)
Given overlapping shapes A and B:

<table>
<thead>
<tr>
<th>Union</th>
<th>Intersection</th>
<th>Subtraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>A \ B</td>
<td>A \ B</td>
<td>A − B</td>
</tr>
</tbody>
</table>

For example:

How can we implement CSG?

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

Wrap Up

- Discuss next programming assignment
  - Due 10/24/04
- I am away next week. No class!
- Discuss status/problems/issues with this week’s programming assignment