Any questions from last time?
Go over sampling image plane?
Or intersection algorithms?

Ray/Plane Intersection

Ray is defined by \( R(t) = R_0 + R_d \cdot t \) where \( t \geq 0 \)
- \( R_0 \) = 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)

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 \( t \):
\[ t = \frac{-(-A x_o - B y_o - C z_o - D)}{(A x_d + B y_d + C z_d)} \]
\[ t = \frac{-N \cdot R_0 - N \cdot P_0}{N \cdot R_d} \]
What Can Happen?

Ray/Plane Summary

Intersection point:

\[(x_i, y_i, z_i) = (x_o + x_o d_i, y_o + y_o d_i, z_o + z_o d_i)\]

1. Calculate N • R_o and compare it to zero.
2. Calculate t_i and compare it to zero.
3. Compute intersection point.
4. Flip normal if N • R_d is positive

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
**Ray/Ellipsoid Intersection**

Ellipsoid's surface is defined by the set of points \( \{(x_s, y_s, z_s)\} \) satisfying the equation:

\[
\frac{(x_s/a)^2}{a^2} + \frac{(y_s/b)^2}{b^2} + \frac{(z_s/c)^2}{c^2} - 1 = 0
\]

Cylinder's surface is defined by the set of points \( \{(x_s, y_s, z_s)\} \) satisfying the equation:

\[
(x_s)^2 + (y_s)^2 - r^2 = 0 \quad -z_0 \leq z_s \leq z_0
\]

Centers at origin

**Ray/Cylinder Intersection**

Ellipsoid's surface is defined by the set of points \( \{(x_s, y_s, z_s)\} \) satisfying the equation:

\[
\frac{(x_s/a)^2}{a^2} + \frac{(y_s/b)^2}{b^2} + \frac{(z_s/c)^2}{c^2} - 1 = 0
\]

Cylinder's surface is defined by the set of points \( \{(x_s, y_s, z_s)\} \) satisfying the equation:

\[
(x_s)^2 + (y_s)^2 - r^2 = 0 \quad -z_0 \leq z_s \leq z_0
\]

Centers at origin
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 $(P_i x)^2 + (P_i y)^2 \leq r^2$, you've intersected a "cap"
- Which valid intersection is closer?

Superquadrics

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_x \cos \eta \cos \phi \\ a_y \cos \eta \sin \phi \\ a_z \sin \eta \end{bmatrix}$$

$$p(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) P_{i,j+1} \quad 0 \leq u \leq 1 \quad 0 \leq v \leq 1$$

Tesselate Patches and Superquadrics

Tessellate Patches and Superquadrics

Torus

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

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

- Quarto!!!
- Up to four torus intersections

Bezier Patch

- Patch of order $(n, m)$ can be defined in terms of a set of $(n+1)(m+1)$ control points $P_{i,j+1}$ for integer indices $i = 0$ to $n$, $j = 0$ to $m$.

$$p(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) P_{i,j+1} \quad 0 \leq u \leq 1 \quad 0 \leq v \leq 1$$

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$
Utah Teapot

- Modeled by 32 Bézier Patches
- Control points available at http://www.holmes3d.net/graphics/teapot

SMF Triangle Meshes

```
vertices
v -1 -1 -1
v -1 -1 1
v -1 1 -1
v -1 1 1
v 1 -1 -1
v 1 -1 1
v 1 1 -1
v 1 1 1
```

```
triangles
f 1 3 4
f 1 4 2
f 5 6 8
f 5 8 7
f 1 2 6
f 1 6 5
f 3 7 8
f 3 8 4
f 1 5 7
f 1 7 3
f 2 4 8
f 2 8 6
```

Triangle Meshes (.iv)

```
// [vertices] (.iv)
\v v1 x y z
\v v2 x y z
\v v3 x y z
\v v4 x y z
\v v5 x y z
\v v6 x y z
\v v7 x y z
\v v8 x y z
\v...
```

Transformations & Hierarchical Models

- 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
  \[ \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} w \\ z \\ y \\ x \end{bmatrix} \]

Homogeneous Coordinates

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

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

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

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

\[
\begin{bmatrix}
  x' \\
  y' \\
  z'
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & 0 & L \\
  0 & 1 & 0 & L \\
  0 & 0 & 1 & L \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]

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

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

You only have to implement uniform scaling!

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

Rotation

About z axis

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

About x axis:

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

About y axis:

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

Rotation

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

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

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

How are transforms combined?

Use matrix multiplication:

\[
T(p) = T(S) p = (TS) p
\]

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

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

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

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

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

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

Transformations in Ray Tracing

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

```
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 0.2 0.4 }

Materials { 
numMaterials <> 
<MATERIALS> 

Group { 
numObjects <> 
<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

Directed Acyclic Graph

is more efficient and useful

- Leaf Node (superquadric)
- Non-Leaf Node (boolean operation)
- Transformation
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

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

Transform Ray

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

Transforming Points & Directions

- Transform point
- Transform direction
- 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 property, not a geometric entity
- Correct normal transformation matrix:
  \[A = (M^{-1})^T\]

See http://www.cgafaq.info/wiki/Transforming_Normals
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>

How can we implement 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

Collect all the intersections

Implementing CSG

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**
Ray intersects union: at first intersection
\[
\text{Min}(t_C^\text{min}, t_B^\text{min})
\]

**Intersection**
First time in B and in C
If \((t_C^\text{max} < t_B^\text{min})\) and \((t_C^\text{min} > t_B^\text{max})\):
- \(t_B^\text{min}\)
Else if \((t_C^\text{max} < t_B^\text{min})\) and \((t_C^\text{min} > t_B^\text{max})\):
- \(t_C^\text{min}\)
Else: none

**Difference**
First time in B not in C
If \((t_B^\text{min} < t_C^\text{min})\): \(t_B^\text{min}\)
Else if \((t_C^\text{min} < t_B^\text{max})\): \(t_B^\text{max}\)
Else: none

Possible ways for 2 spans to overlap
Difference

First time in C not in B

If \( \min(t_C) < \min(t_B) \):
\( t_C \)
Else if \( \max(t_B) < \max(t_C) \):
\( t_B \)
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
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:
  - $t_{\min}$ of positive object – normal of surface
  - $t_{\max}$ of negative object – negated normal

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 (Simon Chorley)
Example (Simon Chorley)

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