Advanced Rendering Techniques

Dr. David Breen
Online
Wednesday 6PM → 8:50PM
Presentation 4
4/29/20
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
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, $w$

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

\[
p' = M p
\]
Most of the time $w = 1$, and we can ignore it.

$$\begin{pmatrix}
x' \\
y' \\
z' \\
1
\end{pmatrix} = \begin{pmatrix}
a & b & c & d \\
e & f & g & h \\
i & j & k & l \\
0 & 0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}$$
Translate \((tx, ty, tz)\)

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

\[
\begin{pmatrix}
    x' \\ y' \\ z' \\ 1
\end{pmatrix} =
\begin{pmatrix}
    1 & 0 & 0 & t_x \\
    0 & 1 & 0 & t_y \\
    0 & 0 & 1 & t_z \\
    0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    x \\ y \\ z \\ 1
\end{pmatrix}
\]
Scale \((s_x, s_y, s_z)\)

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

\[
\begin{pmatrix}
  x' \\
  y' \\
  z' \\
  1
\end{pmatrix} =
\begin{pmatrix}
  S_x & 0 & 0 & 0 \\
  0 & S_y & 0 & 0 \\
  0 & 0 & S_z & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]

- **You only have to implement uniform scaling**
Rotation

About z axis

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

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

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

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

\[
\begin{pmatrix}
x' \\
y' \\
z' \\
1
\end{pmatrix} =
\begin{pmatrix}
k_xk_x(1-c)+c & k_yk_x(1-c)-k_zs & k_xk_z(1-c)+k_yS & 0 \\
k_yk_x(1-c)+k_zs & k_yk_y(1-c)+c & k_yk_z(1-c)-k_xs & 0 \\
k_zk_x(1-c)-k_yS & k_zk_y(1-c)+k_xs & k_zk_z(1-c)+c & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]

where \( c = \cos \theta \) & \( s = \sin \theta \)
How are transforms combined?

Scale then Translate

Use matrix multiplication: \( p' = T( S( p ) ) = ((TS)p) \)

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

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

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

Translate then Scale: \[ p' = S ( T p ) = ST p \]
Non-commutative Composition

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

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

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

\[
ST = \begin{pmatrix}
2 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 3 \\
0 & 1 & 1 \\
0 & 0 & 1 \\
\end{pmatrix}
= \begin{pmatrix}
2 & 0 & 6 \\
0 & 2 & 2 \\
0 & 0 & 1 \\
\end{pmatrix}
\]
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

Scene

- Camera
- Lights
- Background
- Materials
- Objects
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
        Group {
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> }
        }
        Group {
            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 PARAMS> }
        Box { <BOX PARAMS> }
    }
    Group {
        numObjects 2
        Group {
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> }
        }
        Group {
            Box { <BOX PARAMS> }
            Sphere { <SPHERE PARAMS> }
            Sphere { <SPHERE PARAMS> }
        }
    }
    Plane { <PLANE PARAMS> }
}
Adding Transformations
Using Transformations

- Position the logical groupings of objects within the scene

- Transformation in group
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**

\[ p_{WS} = M \quad p_{OS} \]

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

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

- New direction:
  \[ \text{direction}_{OS} = M^{-1} (\text{origin}_{WS} + 1 \times \text{direction}_{WS}) - M^{-1} \text{origin}_{WS} \]
  \[ \text{direction}_{OS} = M^{-1} \text{direction}_{WS} \]
Transforming Points & Directions

- **Transform point**
  \[
  \begin{pmatrix}
  x' \\
  y' \\
  z'
  \end{pmatrix} =
  \begin{pmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  0 & 0 & 0 & 1
  \end{pmatrix}
  \begin{pmatrix}
  x \\
  y \\
  z \\
  1
  \end{pmatrix} =
  \begin{pmatrix}
  ax + by + cz + d \\
  ex + fy + gz + h \\
  ix + jy + kz + l \\
  1
  \end{pmatrix}
  \]

- **Transform direction**
  \[
  \begin{pmatrix}
  x' \\
  y' \\
  z'
  \end{pmatrix} =
  \begin{pmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  0 & 0 & 0 & 1
  \end{pmatrix}
  \begin{pmatrix}
  x \\
  y \\
  z \\
  0
  \end{pmatrix} =
  \begin{pmatrix}
  ax + by + cz \\
  ex + fy + gz \\
  ix + jy + kz \\
  0
  \end{pmatrix}
  \]

- **Map intersection point and normal back to world coordinates**

Homogeneous Coordinates: \((x,y,z,w)\)

\(W = 0\) is a point at infinity (direction)
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:

\[ A = \left( M^{-1} \right)^T \]

Constructive Solid Geometry (CSG)

Given overlapping shapes A and B:

**Union**

**Intersection**

**Subtraction**
How can we implement CSG?

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

Union

Intersection

Subtraction
Collect all the intersections

Union                Intersection        Subtraction
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
Constructive Solid Geometry

Ray Tracing CSG Models

Rick Parent
Ohio State U.
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
Ray intersects union: at first intersection

$\text{Min } (t^C_{\min}, t^B_{\min})$
Possible ways for 2 spans to overlap
If \((t^C_{\text{min}} < t^B_{\text{min}}) \text{ and } (t^C_{\text{max}} > t^B_{\text{min}})\): \(t^B_{\text{min}}\)
Else If \((t^B_{\text{min}} < t^C_{\text{min}}) \text{ and } (t^B_{\text{max}} > t^C_{\text{min}})\): \(t^C_{\text{min}}\)
Else: none

First time in B and in C
First time in B not in C

If \((t^B_{\text{min}} < t^C_{\text{min}})\): \(t^B_{\text{min}}\)
Else if \((t^C_{\text{max}} < t^B_{\text{max}})\): \(t^C_{\text{max}}\)
Else: none
Difference

First time in C not in B

\[
\text{If } (t^C_{\text{min}} < t^B_{\text{min}}): \quad t^C_{\text{min}} \\
\text{Else if } (t^B_{\text{max}} < t^C_{\text{max}}): \quad t^B_{\text{max}} \\
\text{Else: none}
\]
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

- T1
- T2
- T3
- T4
- T5

T2 (rectangle)
T1 (circle)
T3 (rectangle)
T4
T5
Tree Structure #2
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

http://www.cs.mtue.edu/~shene/COURSES/cs3621/NOTES/model/csg.html
Bounding Volumes

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

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

Diagram:

- **T5**
  - **T4**
  - **T2** rectangle
  - **T1** circle
  - **T3** rectangle
Example
Example
Example
Example  (Simon Chorley)
Example (Simon Chorley)
For example:
Wrap Up

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