Lecture Outline

- Reading for Last Class: §5.1 – 5.2, Eberly 2e
- Reading for Today: §4.4 – 4.7, Eberly 2e
- Reading for Next Class: §5.3 – 5.5, Eberly 2e, CGA handout
- Last Time: Introduction to Animation
  - Definition, overview, brief history
  - Principles of traditional animation
  - Keyframe animation, inbetweening (interpolation)
  - Articulated figures (preliminaries of character modeling)
  - Dynamics vs. kinematics, forward vs. inverse
- Today: Scene Graph Rendering
  - State: transforms, bounding volumes, render state, animation state
  - Managing renderer and animation state
  - Rendering: object-oriented message passing overview
- Next Class: Special Effects (SFX), Skinning, Morphing
- Coming Up: More Videos (Lectures 19 & 20)
### Where We Are

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Topic</th>
<th>Primary Source(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Course Overview</td>
<td>Chapter 1, Eberly 2</td>
</tr>
<tr>
<td>1</td>
<td>CG Basics: Transformation Matrices; Lab 0</td>
<td>Sections (§) 2.1-2.2</td>
</tr>
<tr>
<td>2</td>
<td>Viewing 1: Overview, Projections</td>
<td>§ 2.3, esp. 2.3.4; PVFII slides</td>
</tr>
<tr>
<td>3</td>
<td>Viewing 2: Viewing Transformation</td>
<td>§ 2.3, esp. 2.3.4; PVFII slides</td>
</tr>
<tr>
<td>4</td>
<td>Lab 1a: Flash &amp; OpenGL Basics</td>
<td>Ch. 2, 10; Angel Primer</td>
</tr>
<tr>
<td>5</td>
<td>Viewing 3: Graphics Pipeline</td>
<td>§ 2.3, esp. 2.3.7; 2.6, 2.7</td>
</tr>
<tr>
<td>6</td>
<td>Scan Conversion 3: Lines, Midpoint Algorithm</td>
<td>§ 2.5.1, 3.1; PVFII slides</td>
</tr>
<tr>
<td>7</td>
<td>Viewing 4: Clipping &amp; Culling; Lab 1b</td>
<td>§ 2.3.5, 2.4, 2.3.3, 3.1.3</td>
</tr>
<tr>
<td>8</td>
<td>Scan Conversion 4: Polygons, Clipping Intro</td>
<td>§ 2.4, 2.5, esp. 2.5.4, 3.16</td>
</tr>
<tr>
<td>9</td>
<td>Surface Detail 1: Illumination &amp; Shading</td>
<td>§ 2.5, 2.6, 1-2, 2.6.2, 4.3.2, 20.2</td>
</tr>
<tr>
<td>10</td>
<td>Lab 2a: Direct3D / DirectX Intro</td>
<td>§ 2.7, Direct3D handout</td>
</tr>
<tr>
<td>11</td>
<td>Surface Detail 2: Textures, OpenGL Shading</td>
<td>§ 2.6.3, 20.2-30.4; Primer</td>
</tr>
<tr>
<td>12</td>
<td>Surface Detail 3: Mappings; OpenGL Textures</td>
<td>§ 20.5-20.13</td>
</tr>
<tr>
<td>13</td>
<td>Surface Detail 4: Pixel/Vertex Shading; Lab 2b</td>
<td>§ 3.1</td>
</tr>
<tr>
<td>14</td>
<td>Surface Detail 5: DIRECTX Shading; CGLSL</td>
<td>§ 3.2-3.4; Direct3D handout</td>
</tr>
<tr>
<td>15</td>
<td>Demos 1: CGA, Fun, Scene Graphs; State</td>
<td>§ 4.1-4.3; CGA handout</td>
</tr>
<tr>
<td>16</td>
<td>Lab 3a: Shading &amp; Transparency</td>
<td>§ 2.6, 20.1; Primer</td>
</tr>
<tr>
<td>17</td>
<td>Animation 1: Basics, Keyframes; HW/Exam</td>
<td>§ 5.1-5.2</td>
</tr>
<tr>
<td>18</td>
<td>Scene Graphs: Rendering; Lab 3b: Shader</td>
<td>§ 4.4-4.7</td>
</tr>
<tr>
<td>19</td>
<td>Demos 2: SPX, Skinning, Morphing</td>
<td>§ 4.4-4.5; CGA handout</td>
</tr>
<tr>
<td>20</td>
<td>Demos 3: Surfaces, B-Rep/Volume Graphics</td>
<td>§ 10.4, 12.7; Mesh handout</td>
</tr>
</tbody>
</table>

Lightly-shaded entries denote the due date of a written problem set. Heavily-shaded entries, that of a machine problem (programming assignment); blue-shaded entries, that of a paper review, and the green-shaded entry, that of the term project.

Green, blue and red letters denote exam review, exam, and exam solution review dates.

---

### Acknowledgements:

**Computer Animation Intro**

- **Jason Lawrence**
  - Assistant Professor
  - Department of Computer Science
  - University of Virginia

- **Thomas A. Funkhouser**
  - Professor
  - Department of Computer Science
  - Computer Graphics Group
  - Princeton University

Acknowledgment: slides by Misha Kazhdan, Allison Klein, Tom Funkhouser, Adam Finkelstein and David Dobkin

Review [1]: 19th Century Animation Before Motion Pictures


Tarzan © 2000 Disney http://youtu.be/ezMnoSS5Hw


Review [2]: Animation, Simulation & Visualization

What is animation?
- Make objects change over time according to scripted actions

What is simulation?
- Predict how objects change over time according to physical laws

Adapted from slides © 2010 J. Lawrence, University of Virginia


© 2007 Wikipedia, Zoetrope http://www.cs.berkeley.edu/People/...
Review [3]:
Principles of Traditional Animation

- Squash and Stretch
- Timing
- Anticipation
- Staging
- Follow Through and Overlapping Action
- Straight Ahead Action and Pose-to-Pose Action
- Slow In and Out
- Arcs
- Exaggeration
- Secondary action
- Appeal

SIGGRAPH: http://bit.ly/1DsO44

© 2010 J. Lawrence, University of Virginia

Review [4]:
Traditional Animation – Anticipation

- The preparation for an action.
  - Muscle contraction prior to extension
  - Bending over to lift a heavy object
  - Luxo’s dad responds to Luxo Jr. off screen before Luxo Jr. appears.

Luxo Jr. © 1986 Pixar
http://www.pixar.com/shorts/ljr/
http://youtu.be/qGxoui3IFS0

© 2010 J. Lawrence, University of Virginia
Review [5]:
Keyframe Animation & Inbetweening

- Interpolate variables describing keyframes to determine poses for character “in-between”

Lasseter '87

Review [6]:
Linear Interpolation aka Lerping

- Inbetweening:
  - Linear interpolation - usually not enough continuity
Review [7]:
Articulated Figures

- Character poses described by set of rigid bodies connected by “joints”

```
Base
  Arm
    Hand
```

Scene Graph

Angel Figures 8.8 & 8.9

Review [8]:
Character Modeling

- Well-suited for humanoid characters

```
Root
  Chest
    Neck
      Head
    LCollar
      LShld
        LElbow
          LWrist
    RCollar
      RShld
        RElbow
          RWrist
  LHip
    LKnee
    LAnkle
  RHip
    RKnee
    RAnkle
```

Rose et al. ’96
Review [9]: Bones & Joints

- Articulated figure:

  ![Articulated figure diagram](image)

  Hip → Upper leg → Knee → Lower leg → Ankle → Foot

  Upper leg (hip rot) → Hip rotate
  Lower leg (knee rot) → Hip rotate + knee rot
  Foot (ankle rot)

© 2010 J. Lawrence, University of Virginia

Scene Graph Traversal

- A 3D scene...
- ...and its scene graph

- Object: Tons A
  - Mesh: Tons
  - Material: Brick
  - Shader: Water Fx

- Object: Tons B
  - Mesh: Chamber Box
  - Material: Water

- Object: Cube
  - Mesh: Chamber Box
  - Material: Wooden

© 2002 – 2005 Virtools
http://bit.ly/eM1gz8
Scene Graph Rendering

Acknowledgements:
Scene Graphs – Eberly 1e

David H. Eberly
Chief Technology Officer
Geometric Tools, LLC
http://www.geometrictools.com

3D Game Engine Design © 2000 D. H. Eberly
Review: What Information is in Scene Graphs?

- Transforms
- Bounding Volumes
- Render State
- Animation State

![Diagram of a simple tree with one grouping node.]

Figure 4.1 A simple tree with one grouping node.

Kinds of Transforms

- Local
  - Translation, rotation, scaling, shearing
  - All within parent’s coordinate system

\[
\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{pmatrix}.
\]

Using this compacted notation, the product of two homogeneous matrices is

\[
\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}.
\]

and the product of a homogeneous matrix with a homogeneous vector \( \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} \) is

\[
\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.
\]

- World: Position Child \( C \) With Respect to Parent \( P \) (Depends on Local)

\[
\begin{pmatrix} x_{WC} \\ y_{WC} \\ z_{WC} \\ 1 \end{pmatrix} = \begin{pmatrix} M_{WC} \\ P_{WC} \end{pmatrix} = \begin{pmatrix} M_{WC} & P_{WC} \end{pmatrix} = \begin{pmatrix} M_{WC} & P_{WC} \end{pmatrix}.
\]

- Both Together Part of Modelview Transformation

Adapted from 3D Game Engine Design © 2000 D. H. Eberly
Traversing Scene Graph: World Transform of Node

The world transform of the root node in the scene graph is just its local transform. The world position of a node $N_k$ in a path $N_0 \cdots N_k$, where $N_0$ is the root node, is generated recursively by the above definition as

$$M_{\text{world}}^{(N_k)} \cdot T_{\text{world}}^{(N_k)} = \left( M_{\text{local}}^{(N_0)} \cdot T_{\text{local}}^{(N_0)} \right) \cdots \left( M_{\text{local}}^{(N_k)} \cdot T_{\text{local}}^{(N_k)} \right).$$

Bounding Volumes [1]: Definition

- **Bounding Volume Hierarchies (BVHs)**
  - Root: entire scene
  - Interior node: rectangle (volume in general) enclosing other nodes
  - Leaves: primitive objects
  - Often axis-aligned (e.g., axis-aligned bounding box aka AABB)
- **Used**
  - Visible surface determination (VSD) – especially occlusion culling
  - Other intersection testing: collisions, ray tracing

Bounding Volume Hierarchy (BVH) © 2009 Wikipedia
http://en.wikipedia.org/wiki/Bounding_volume_hierarchy
Bounding Volumes [2]:
Types Covered in Eberly

- Spheres
- Oriented Boxes aka Oriented Bounding Boxes (OBBs)
- Capsules
- Lozenges
- Cylinders
- Ellipsoids

Renderer State

- Can Capture Render Information Hierarchically
- Example
  - Suppose subtree has all leaf nodes that want textures alpha blended
  - Can tag root of subtree with “alpha blend all”
  - Alternatively: tag every leaf
- How Traversal Works: State Accumulation
  - Root-to-leaf traversal accumulates state to draw geometry
  - Renderer checks whether state change is needed before leaf drawn
- Efficiency Considerations
  - Minimize state changes
  - Reason: memory copy (e.g., system to video memory) takes time
Animation State

- Can capture animation information hierarchically
- Example
  - Consider articulated figure from last lecture
  - Let each node represent joint of character model
    - Neck
    - Shoulder
    - Elbow
    - Wrist
    - Knee
- Procedural transformation
- How it works: Controllers
  - Each node has controller function/method
  - Manages quantity that changes over time (e.g., angle)

Updating Scene Graphs

- Need to merge bounding volumes (boxes, lozenges, capsules)
- Update geometric state: `UpdateGS`
  ```cpp
def Spatial::UpdateGS(float time, bool initiator) {
    UpdateWorldData(time);
    UpdateWorldBound();
    if (initiator)
        PropagateBoundToRoot();
}
```
- `UpdateWorldData`: Virtual function, controls downward pass
- `UpdateWorldBound`: Also virtual, controls upward pass
- `PropagateBoundToRoot`: Not virtual, simple recursive call
  - `parent.UpdateWorldBound()`
  - `parent.PropagateBoundToRoot()`
By Spheres vs. By Oriented Boxes

Pseudocode

```
bool CullSpherePlane (Sphere sphere, Plane plane)
{
    return Dot(plane.N, sphere.C) - plane.d < -sphere.r;
}
```

Can Also Cull by: Lozenges, Cylinders, Ellipsoids

Main Draw Method

```
void Renderer::Draw (Spatial scene)
{
    scene.onDraw(thisRenderer);
}
```

Calls virtual function Draw(renderer)
Passed down
Geometry::Draw(Renderer renderer)
Node::Draw(Renderer renderer) Calls child.onDraw(renderer)
Derived Classes of Geometry
TriMesh::Draw(Renderer renderer)
Similarly for other derived classes
Summary

- Reading for Last Class: §5.1 – 5.2, Eberly 2e
- Reading for Today: §4.4 – 4.7, Eberly 2e
- Reading for Next Class:

- Last Time: Introduction to Animation
  - Definition, overview, brief history, principles
  - Keyframes, interpolation, articulated figures for character modeling
  - Dynamics vs. kinematics, forward vs. inverse

- Today: Scene Graph Rendering
  - State: transforms, bounding volumes, render state, animation state
  - Updating: merging bounding volumes
  - View frustum culling
  - Rendering: object-oriented message passing overview

- Next Class: Special Effects (SFX), Skinning, Morphing; More Videos

Terminology

- Shading and Transparency in OpenGL: Alpha, Painter’s, z-buffering
- Animation – Modeling Change Over Time According to Known Actions
- Keyframe Animation
  - Keyframe
  - Interpolation
  - Character model
- State in Scene Graphs
  - Transforms – local & global TRS to orient parts of model
  - Bounding volumes – spheres, boxes, capsules, lozenges, ellipsoids
  - Renderer state – lighting, shading/textures/alpha
  - Animation state – TRS transformations (especially R), controllers
- Traversal: Moving through Data Structure, Calling Methods