Lecture 18 of 41

Scene Graphs: Rendering
Lab 3b: Shader

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course pages: http://bit.ly/hGvXlH / http://bit.ly/eVizrE
Public mirror web site: http://www.kddresearch.org/Courses/CIS636
Instructor home page: http://www.cis.ksu.edu/~bhsu

Readings:

Today: §4.4 – 4.7, Eberly 2e – see http://bit.ly/ieUq45
Next class: §5.3 – 5.5, Eberly 2e, CGA handout

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

Lecture	Topic	Primary Source(s)
0	Course Overview	Chapter 1, Eberly 2e
1	**CG Basics: Transformation Matrices; Lab 0**	**Sections (§) 2.1, 2.2**
2	Viewing 1: Overview, Projections	§ 2.2.3 – 2.2.4, 2.8
3	Viewing 2: Viewing Transformation	§ 2.3 esp. 2.3.4; FVFH slides
4	**Lab 1a: Flash & OpenGL Basics**	**Ch. 2, 16[1], Angel *Primer***
5	Viewing 3: Graphics Pipeline	§ 2.3 esp. 2.3.7; 2.6, 2.7
6	Scan Conversion 1: Lines, Midpoint Algorithm	§ 2.5.1, 3.1; FVFH slides
7	**Viewing 4: Clipping & Culling; Lab 1b**	**§ 2.3.5, 2.4, 3.1.3**
8	Scan Conversion 2: Polygons, Clipping Intro	§ 2.4, 2.5 esp. 2.5.4, 3.1.6
9	Surface Detail 1: Illumination & Shading	§ 2.5, 2.6.1 – 2.6.2, 4.3.2, 20.2
10	**Lab 2a: Direct3D / DirectX Intro**	**§ 2.7, Direct3D handout**
11	Surface Detail 2: Textures; OpenGL Shading	§ 2.6.3, 20.3 – 20.4, *Primer*
12	Surface Detail 3: Mappings; OpenGL Textures	§ 20.5 – 20.13
13	**Surface Detail 4: Pixel/Vertex Shad.; Lab 2b**	**§ 3.1**
14	Surface Detail 5: Direct3D Shading; OGLSL	§ 3.2 – 3.4, Direct3D handout
15	Demos 1: CGA, Fun; Scene Graphs: State	§ 4.1 – 4.3, **CGA handout**
16	**Lab 3a: Shading & Transparency**	**§ 2.6, 20.1, *Primer***
17	**Animation 1: Basics, Keyframes; HW/Exam**	**§ 5.1 – 5.2**
	Exam 1 review; Hour Exam 1 (evening)	**Chapters 1 – 4, 20**
18	**Scene Graphs: Rendering; Lab 3b: Shader**	**§ 4.4 – 4.7**
19	**Demos 2: SFX; Skinning, Morphing**	**§ 5.3 – 5.5, CGA handout**
20	Demos 3: Surfaces; B-reps/Volume Graphics	§ 10.4, 12.7, **Mesh handout**

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
http://www.cs.virginia.edu/~jdl/

Computer Science at the University of Virginia

Acknowledgment: slides by Misha Kazhdan, Allison Klein, Tom Funkhouser, Adam Finkelstein and David Dobkin
http://bit.ly/eB1Oj4

Thomas A. Funkhouser
Professor
Department of Computer Science
Computer Graphics Group
Princeton University
http://www.cs.princeton.edu/~funk/

Princeton University

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

© 2007 Wikipedia, *Phenakistoscope*
http://bit.ly/eAnURG

© 2008 Wikipedia, *Thaumatrope*
http://bit.ly/fFl6xH

Zoetrope (Praxinoscope)

Tarzan © 2000 Disney
http://youtu.be/zc3MnoSS5Hw

Adapted from slides © 2010 J. Lawrence, University of Virginia
CS 4810: Introduction to Computer Graphics – http://bit.ly/hPIXdi

Review [2]: Animation, Simulation & Visualization

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

Pixar

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

Wilhelmson *et al.* (2004)
http://youtu.be/EgumU0Ns1YI
http://avl.ncsa.illinois.edu
http://bit.ly/eA8PXN

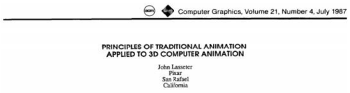
University of Illinois

Adapted from slides © 2010 J. Lawrence, University of Virginia
CS 4810: Introduction to Computer Graphics – http://bit.ly/hPIXdi

7

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


Computer Graphics, Volume 21, Number 4, July 1987

PRINCIPLES OF TRADITIONAL ANIMATION APPLIED TO 3D COMPUTER ANIMATION

John Lasseter
Pixar
San Rafael
California

Lasseter, J. (1987). Principles of traditional animation applied to 3D computer animation. *Computer Graphics*, 21(4), pp. 35-44.
SIGGRAPH: http://bit.ly/1DsO44
ACM Portal: http://bit.ly/eyx2PN

8

Review [4]: Traditional Animation – Anticipation

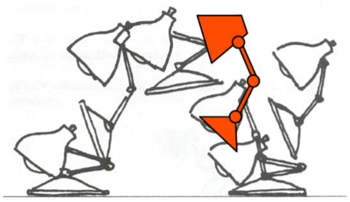
- The preparation for an action.
  - Muscle contraction prior to extension
  - Bending over to lift a heavy object
  - Luxo's dad responds to Luxo Jr. off screen before Luxo Jr. appears.

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

9

Review [5]: Keyframe Animation & Inbetweening

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

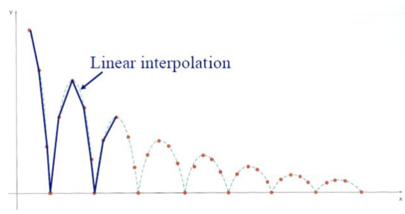


Lasseter `87

10

Review [6]: Linear Interpolation *aka* Lerping

- Inbetweening:
  - Linear interpolation - usually not enough continuity

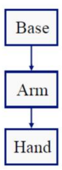


H&B Figure 16.16

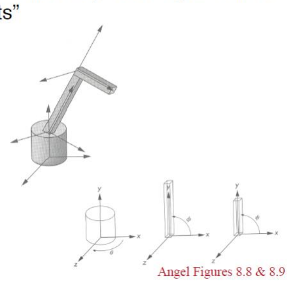
11

Review [7]: Articulated Figures

- Character poses described by set of rigid bodies connected by "joints"



Scene Graph

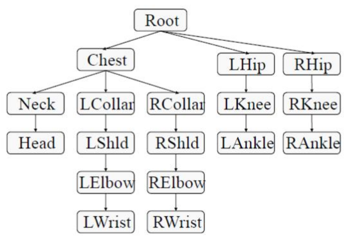


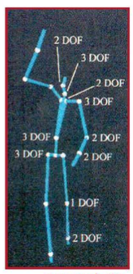
Angel Figures 8.8 & 8.9

12

Review [8]: Character Modeling

- Well-suited for humanoid characters

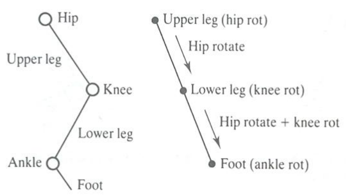




Rose et al. `96

Review [9]: Bones & Joints

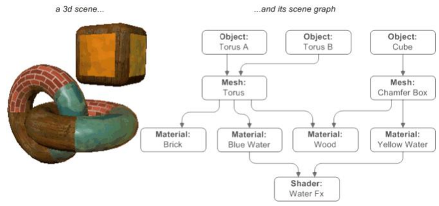
- Articulated figure:



Watt & Watt

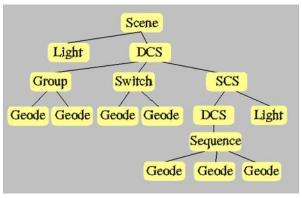
CS 4810: Introduction to Computer Graphics – http://bit.ly/hPlXdi

Scene Graph Traversal



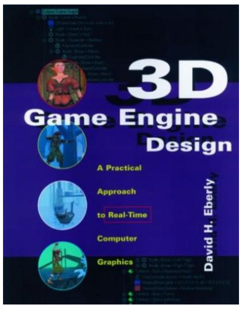
http://bit.ly/eM1gz8

Scene Graph Rendering



http://www.evl.uic.edu/pape/talks/VSI97/pf/

Acknowledgements: Scene Graphs – Eberly 1e



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

See http://bit.ly/ieUq45 for second edition

Review: What Information is in Scene Graphs?

- Transforms
- Bounding Volumes
- Render State
- Animation State

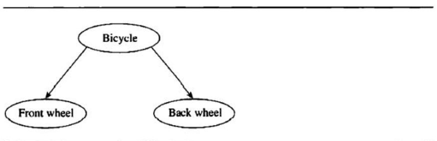


Figure 4.1 A simple tree with one grouping node.

See http://bit.ly/ieUq45 for second edition

Review: Kinds of Transforms

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

$$\langle M \mid \vec{T} \rangle := \left[\begin{array}{c|c} M & \vec{T} \\ \hline \vec{0}^{\mathrm{T}} & 1 \end{array} \right]. \quad (4.1)$$

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

$$\langle M_1 \mid \vec{T}_1 \rangle \langle M_2 \mid \vec{T}_2 \rangle = \langle M_1 M_2 \mid M_1 \vec{T}_2 + \vec{T}_1 \rangle \quad (4.2)$$

and the product of a homogeneous matrix with a homogeneous vector $[\vec{V} \mid 1]^{\mathrm{T}}$ is

$$\langle M \mid \vec{T} \rangle \, \vec{V} = M\vec{V} + \vec{T}. \quad (4.3)$$

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

$$\langle M^{(C)}_{\text{world}} \mid \vec{T}^{(C)}_{\text{world}} \rangle = \langle M^{(P)}_{\text{world}} \mid \vec{T}^{(P)}_{\text{world}} \rangle \langle M^{(C)}_{\text{local}} \mid \vec{T}^{(C)}_{\text{local}} \rangle$$
$$= \langle M^{(P)}_{\text{world}} M^{(C)}_{\text{local}} \mid M^{(P)}_{\text{world}} \vec{T}^{(C)}_{\text{local}} + \vec{T}^{(P)}_{\text{world}} \rangle.$$

- Both Together Part of Modelview Transformation

See http://bit.ly/ieUq45 for second edition

19

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

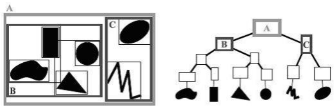
$$\left\langle M_{\text{world}}^{(N_k)} \,\middle|\, \vec{T}_{\text{world}}^{(N_k)} \right\rangle = \left\langle M_{\text{local}}^{(N_0)} \,\middle|\, \vec{T}_{\text{local}}^{(N_0)} \right\rangle \cdots \left\langle M_{\text{local}}^{(N_k)} \,\middle|\, \vec{T}_{\text{local}}^{(N_k)} \right\rangle.$$

Adapted from *3D Game Engine Design* © 2000 D. H. Eberly
See http://bit.ly/ieUq45 for second edition

20

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

21

Bounding Volumes [2]: Types Covered in Eberly

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

Adapted from *3D Game Engine Design* © 2000 D. H. Eberly
See http://bit.ly/ieUq45 for second edition

22

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

Adapted from *3D Game Engine Design* © 2000 D. H. Eberly
See http://bit.ly/ieUq45 for second edition

23

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)

© 2002 D. M. Murillo
http://bit.ly/eZ9MA8

Adapted from *3D Game Engine Design* © 2000 D. H. Eberly
See http://bit.ly/ieUq45 for second edition

24

Updating Scene Graphs

- Need to Merge Bounding Volumes (Boxes, Lozenges, Capsules)
- Update Geometric State: `UpdateGS`

```
void 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 ()`

Adapted from *3D Game Engine Design* © 2000 D. H. Eberly
See http://bit.ly/ieUq45 for second edition

Rendering Scene Graphs [1]: View Frustum Culling

- By Spheres *vs.* By Oriented Boxes

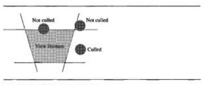


Figure 4.2 Examples of culled and unculled objects.

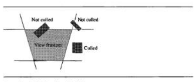


Figure 4.3 Examples of culled and unculled objects.

- Pseudocode

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

```
bool CullBoxPlane (Box box, Plane plane)
{
    r = box.a0*|Dot(plane.N,box.A0)| +
        box.a1*|Dot(plane.N,box.A1)| +
        box.a2*|Dot(plane.N,box.A2)|;

    return Dot(plane.N,box.C) - plane.d < -r;
}
```

- Can Also Cull by: Lozenges, Cylinders, Ellipsoids

Adapted from *3D Game Engine Design* © 2000 D. H. Eberly
See http://bit.ly/ieUq45 for second edition

Rendering Scene Graphs [2]: Message Passing

- Main `Draw` Method

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

`Spatial::OnDraw(Renderer renderer)`
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

Adapted from *3D Game Engine Design* © 2000 D. H. Eberly
See http://bit.ly/ieUq45 for second edition

Summary

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