Introduction to CIS 536/636


CGA and Realism

Example: think of flecks in traditional film reels

Where We Are

Review [1]: Scene Graphs

- Scene Graph: General Data Structure used in CG
- Used to: compute visibility, set up rendering pipeline
- Actual graph: general graph, forest, or rooted tree
- Scene Graph Traversal: Initial Step – Drives Rendering
- Features of Scene Graphs
  - Spatial partitioning: 4. Using bounding volume hierarchies
  - Leaves: primitive components
  - Interior nodes: assembly operations, model/object transformations
  - Root(s): scene or major objects

Review [2]: Aesthetic Considerations

- Non-Photorealistic Rendering: Aimed at achieving Natural Aesthetic
  - Cartoon shaders: use sharp gradient (thresholded)
  - Pencil shaders: blurring, stippling
- CGA and Realism
  - Term from signal processing
  - Two sampled signals indistinguishable from (aliases of) one another
  - Examples: jaggies, Moiré vibration (Moiré pattern)
  - Anti-aliasing: operations to prevent such effects
- Temporal Aliasing
  - Similar effect in animation
  - Small artifact can be much more jarring!
  - Example: think of flecks in traditional film reels

Review [3]: CG Feature Films & Shorts

- Where We Are

- Lecture Outline

- Reading for Last Class: §4.1 – 4.3, Eberly 2
- Reading for Today: §3.6, 20.1, Eberly 2
- OpenGL primer material
- Reading for Next Class: §5.1 – 5.2, Eberly 2

- Last Time: Scene Graphs: CGA Demos, Videos
  - Scene graphs and state – main topic
  - State of CGA: videos and discussion
  - Demos to download
    - Adobe Maya: http://students.autodesk.com
  - Today: Shading and Transparency in OpenGL
  - Transparency revisited
    - OpenGL how-to: http://bit.ly/hRaQgk
    - Alpha blending (15.025, 15.040)
    - Screen-door transparency (15.030)
    - Painter’s algorithm & depth buffering (z-buffering)
Review [4]: OpenGL Shading (Overview)

- Set Up Point Light Sources
- Set Up Materials, Turn Lights On
- Start Drawing (glBegin ... glEnd)

See also: OpenGL: A Primer, 3rd edition (Angel)

Adapted from slides © 2003 F. Pfenning, Carnegie Mellon University

Transparency in OpenGL [1]: Transparent vs. Translucent, Blended

© 1997 – 2011 Khronos Group
http://www.khronos.org/opengl

Transparency in OpenGL [2]: Blending vs. Screen Door

16.396 How can I create screen door transparency?

This is accomplished by defining a single pixel primitive with gldrawpixelf() and by modifying the fragment program. The fragment program must define a new light setting parameter, commonly called the screen door parameter. The screen door parameter determines the amount of transparency, and is used to blend into the background color. For example, setting the screen door parameter to 0.5 results in a more opaque object, and setting it to 1.0 makes the object completely transparent. Screen door transparency is sometimes preferred over the “painter’s algorithm” for objects that need to be transparent.

See also: Viola et al. (2004), http://www.cs.columbia.edu/~viola/

Transparency in OpenGL [3]: Screen Door

© 1997 – 2011 Khronos Group
http://www.khronos.org/opengl

Transparency in OpenGL [4]: Glass

16.396 How can I render glass with OpenGL?

This property is difficult to render, because glass is made up of layers, not a single material. To achieve this effect, we will use the glass primitive with a single-layer effect. The stains are small, randomly distributed, and are added in a specific order. The glass primitive is ideal for this purpose because it is highly reflective and can be used to simulate glass surfaces.

See also: Lim (2010), http://www.cs.columbia.edu/~viola/

Transparency in OpenGL [5]: Alpha & Painter's Algorithm

© 1997 – 2011 Khronos Group
http://www.khronos.org/opengl

See also: Wikipedia, Painter's Algorithm
http://www.wikipedia.org
Transparency in OpenGL [6]: Painter’s algorithm & Z-buffering

10:30 If you were to choose primitive and order another primitive behind it, expect the second problem to surface through the first, but its not there!

Transparency in OpenGL [6]: Partial Transparency

10:30 How can I make part of my texture maps transparent or transparent?

If depends on the effect you’re trying to achieve.

If you want blending to occur after the texture has been applied, then use the OpenGL blending function. Try this:

glEnable( GL_BLEND );
glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA );

You might want to use the alpha here that resulted from texture mapping in the base function. If so, (GL_SRC_ALPHA, GL_ONE), etc.

However, if you want blending to occur before the primitive is rendered (i.e., you want parts of the texture map to allow the containing objects of the primitive to show through), then you can use OpenGL, blending (instead, you use glBlendFunc), and set the base environment to GL_ZERO. In this case, you’d want to leave the base environment to its default value of (0, 0, 0).

Summary

- Reading for Last Class: §4.1 – 4.3, Eberly 2e; CGA handout
- Reading for Today: §2.6, 20.1, Eberly 2e; OpenGL primer material
- Reading for Next Class: §5.1 – 5.2, Eberly 2e
- Last Time: Scene Graphs
  - Maintaining state
  - Coming up: traversal
- CGA Demos, Videos
  - State of CGA: videos
  - Issues: photorealism, hardware, traditional (non-CG) animation
  - Techniques showcased: multiasp, texturing, alpha, portals
- Shading and Transparency in OpenGL
  - Alpha blending
  - Painter’s algorithm — less efficient, can handle non-opaque objects
  - Depth buffering (Z-buffering) — in hardware, fast, opaque only

Terminology

- Non-photorealistic Rendering
  - Cartoon shaders
  - Pencil shaders
- CGA and Realism
  - Aliasing — reconstructed image differs from original
  - Alias — artifact in reconstructed image (jaggy, Moiré pattern, etc.)
  - Anti-aliasing — techniques (e.g., area sampling) for avoiding aliasing
  - Temporal aliasing — aliasing over time (e.g., in animation)
  - Temporal anti-aliasing — smoothing out aliasing over time
- Shading and Transparency in OpenGL
  - Alpha blending — using A channel of R, G, B, A to combine colors
  - Painter’s algorithm aka priority fill — back-to-front rendering
  - Depth buffering (Z-buffering) — checking Z values