EECS 487
September 25, 2006

news
• visit from EA on Wednesday
• no special assignment talk today (unless I do one…)
• proj2 out tomorrow (due in 2 weeks, 6 days)

preliminary issues
• coordinate systems
  – eye space, world space
• transformations: 4x4 matrices
  – combination of rotate, scale, translate, and more
• homogeneous coordinates: points & vectors
  – 4th coordinate added to 3D points and vectors
  – for points it’s 1, for vectors it’s 0
  – thus: P – P = V, P + V = P, V + V = V, P + P = ?
• more on these issues soon

OpenGL lighting
• based on simplifying assumptions:
  – several lights (e.g. 8)
  – types:
    • directional
    • positional
    • spot light
  – reflected light is a combination of 3 terms:
    • ambient (general background level of brightness)
    • diffuse (like latex paint – not shiny)
    • specular (mirror-like; shiny)

diffuse vs. specular
• diffuse:
  – light reflects equally in all directions
• specular:
  – light reflects in one direction (like a mirror)
• which is more realistic?

Plate 16. Twelve spheres, each with different material parameters. The row properties are as follows: row 1 - No ambient reflection; row 2 - Grey ambient reflection; row 3 - Blue ambient reflection. The first column uses a blue diffuse material color with no specular properties. The second column adds white specular reflection with a low shininess exponent. The third column uses a high shininess exponent and thus has a more concentrated highlight. The fourth column uses the blue diffuse color and, instead of specular reflection, adds an emissive component.
multiple lights
- OpenGL has a notion of global ambient light, plus 8 (e.g.) individual light sources
- each light source has colors for:
  - ambient
  - diffuse
  - specular
- is this physically-based?

material properties
- each surface is assigned “material” properties
- 4 colors:
  - ambient
  - diffuse
  - specular
  - emissive
- plus:
  - shininess (specular exponent)

computing final color
- color at a vertex comes from:
  - global ambient light
  - individual light contributions
  - material properties
- in OGL fixed pipeline, lighting is computed per vertex during vertex processing
- resulting colors interpolated across Δ’s

blackboard...
- details for light computations:
  red book, chapter 6

application set up
- see red book, chapter 6
- online version:
  http://glprogramming.com/red/chapter05.html

flow control in jot
GL_VIEW class renders the scene
geom/gl_view.H
1. clear buffer,
2. initialize OGL state (default values)
3. setup lights (see code example)
4. draw objects
drawing objects

loop over list of GELs (disp/gel.H)
generic scene object, includes 2D objects like text
in window corner, also 3D objects (GEOM: geom/geom.H) that contain meshes

for each GEOM:
  send material properties to OGL
  send xform to OGL
  draw BMESH (mesh/bmesh.H)

next

• project 2 out tomorrow
  – OpenGL lighting model in software
  – modified OpenGL lighting in GLSL
    with added “abstraction”
  – environment and bump mapping using GLSL

drawing a mesh

draw BMESH:
  for each Patch (mesh/patch.H)
    draw triangle strips using StripCB
      (mesh/stripcb.H)
  it sends to OGL:
    vertex normals,
    positions,
    colors, etc., depending on type of StripCB
  lets different shaders share same triangle strips
  used in p2 for software lighting (send vert colors)