Point Lighting Using Shaders

Game Design Experience

Professor Jim Whitehead

March 13, 2009

UC SANTA CRUZ

Announcements

- Days until Final Project Due: 3
  - Due Monday, March 16
  - Can turn in game until 5pm Monday
  - Few students have been attending help sessions
  - We will not be able to help you as well at the last minute
  - Help sessions
    - Friday
      - 5:00PM - 8:45PM, Oakes 101
    - Sunday
      - 8pm – 1am(ish), BE 105 (Unix lab)
    - Post question to forum
    - Don’t let yourself stay stuck for too long. 1-2 hours max!

Pop Quiz

- Ungraded – test your knowledge of key concepts
  - Similar to questions that will appear on final
- What does the world matrix represent?
- What does multiplying world * view * projection do?
- What are the two main types of shaders? What do they do?
- What is a normal vector? What is a normalized vector?
- To make a scene brighter, perform what operation on color values?
- What are texture coordinates?

Pop Quiz Answers

- **What does the world matrix represent?**
  - The transformation of a model's coordinates into world coordinates
- **What does multiplying world * view * projection do?**
  - Transforms model coords into world coords, then applies camera
- **What are the two main types of shaders? What do they primarily do?**
  - Vertex shader, pixel shader
  - Vertex shader: mostly changes vertex locations
  - Pixel shader: mostly changes pixel color values
- **What is a normal vector? What is a normalized vector?**
  - Normal vector: A vector pointing in the direction perpendicular to a surface
  - Normalized vector: one where all values lie between 0 and 1
- **To make a scene brighter, perform what operation on color values?**
  - Increase color values
- **What are texture coordinates?**
  - Also known as u,v coordinates, they are fractions of the distance between upper left and lower right corners of a bitmap image

Lighting

- In games, often want to have parts of a scene that are more lit than other parts
  - Helps create the mood of a scene
    - Dark and mysterious, bright and cheerful
  - Increase realism
    - Streetlights are brighter under the light
- Lighting is a complex subject
  - Many ways to create lights, shadows
  - Physical materials interact with light in different ways
  - Dull surface, shiny surface, skin: all different

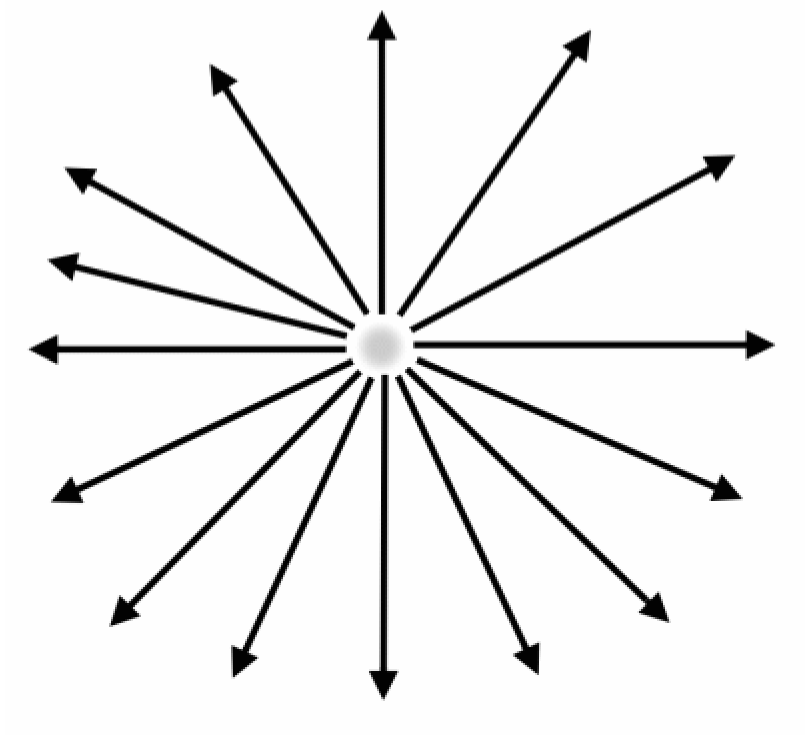
Ambient Light

- Ambient light
  - When a scene has a uniform level of lighting
  - All surfaces of all objects have the same amount of light
- In code
  - Brighter lighting
    - RGB values that are closer to 1
    - As lights get brighter, everything seems more and more white
  - Dimmer lighting
    - RGB values that are closer to 0
    - As lights get dimmer, everything seems more dark
- Ambient light is not very realistic

Point Light

- Represents lights that are similar to a bare light bulb
- Light radiates uniformly in all directions
- Light modeled with a location (lightPos) and an intensity (xPower, values between 0 and 3 work well)

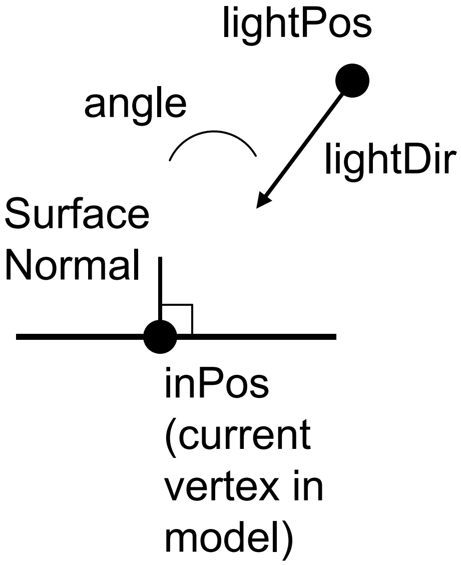
http://exploreankit.files.wordpress.com/2007/05/lightbulb1.jpg



http://www.gamasutra.com/features/20030418/pointlight.gif

Point lighting on a model

- To determine point lighting on a model
  - Determine lightDir vector
    - Direction from point light to location on surface of model
      - lightDir = inPos – lightPos
      - Normalize to make next step easier
  - Compute angle between lightDir and surface normal
    - This gives the percentage of the light's value to apply to surface
    - Determine using dot product
      - a dot b = |a||b| cos (angle)
      - If a & b are normalized, a dot b is cos(angle)
      - Cos(0) = 1, Cos(pi/2) = 0
      - If light overhead (angle = 0), get full intensity
      - If light parallel to surface, get no lighting



Point lighting on a model (cont’d)

- Compute final color as follows
  - ▸ Calculate a base color
    - Grab a color value from a texture by applying texture coordinates
    - Or, apply a uniform base color
  - ▸ Compute the fraction of the light’s intensity that reaches model
    - Model intensity = light intensity (xPower) * cos(angle)
  - ▸ Add the ambient light and the light from the point light to the base color
    - Final color = base color * (model intensity + ambient)

Some important details

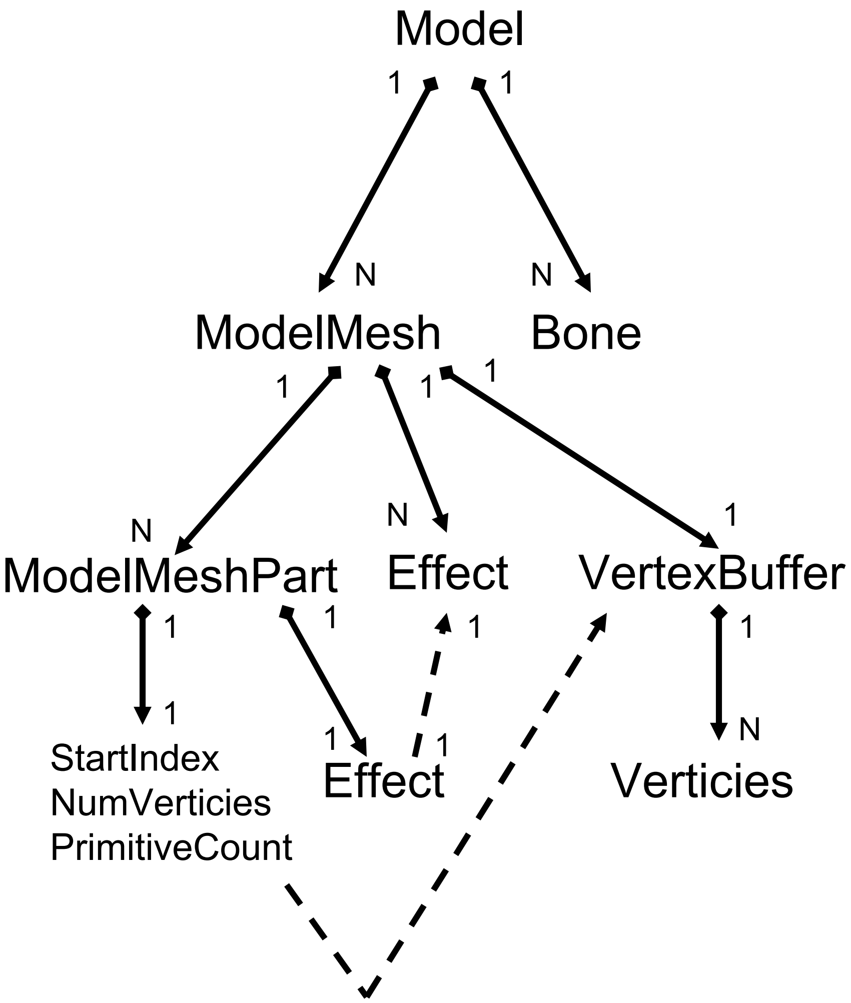
- To compute lighting, Vertex shader needs normal vectors as input
  - Normals come into the Shader via the NORMAL0 semantic
  - These need to be supplied from C#/XNA, since they are part of the model
  - This occurs by default if you draw meshes
    - mesh.Draw sends normal information
  - If drawing triangles, need to tell XNA to send normal information
    - Do this by using the VertexPositionNormalTexture class to define vertices of triangles
      - Each point has (x,y,z) position, (x,y,z) normal, and (u,v) texture coordinate
    - Then, must
      - GraphicsDevice.VertexDeclaration = new VertexDeclaration(GraphicsDevice, VertexPositionNormalTexture.VertexElements);
      - This determines the kind of input data that is passed to the vertex shader

Using your own shader with a mesh

- By default, each part of a mesh has a shader associated with it
  - ▸ Each ModelMeshPart has an associated Effect
  - ▸ An Effect is a shader
- To use your own shader, need to replace model effects with your own

```
for (int i = 0; i < mesh.MeshParts.Count; i++)
    {
        // Set this MeshParts effect to
        // our pixel lighting effect
        mesh.MeshParts[i].Effect = effect;
     }
```

- Overrides effects present in model originally



Example point shader in XNA

- *Example of a point shader C#/XNA*
- *Demonstrated shader from*
  - *http://www.riemers.net/eng/Tutorials/XNA/Csharp/Series3/Per-pixel_lighting.php*