Fixed Function Pipeline

- OpenGL Fixed Function Pipeline
  - OpenGL states can be changed
  - States out of a fixed number of alternatives
  - Same principal operations always performed on stream elements
    - Can only discard certain operations
  - Action on stream elements (vertices/fragments) can not be programmed

In the following:
- Non-OpenGL effects using fixed function pipeline
- Multipass
  - Combine pixel results of multiple rendering passes
  - Pre-computed texture maps
  - Pre-compute results of certain non-standard operations and store in texture maps

Advanced shading

- The Phong lighting model
  - $I = I_a + I_s[ksN_o N_L + ks N_H]$<br>  $I_a$: surface normal<br>  $I_s$: light source direction<br>  $N$: reflection vector<br>  $N_L$: direction to the viewpoint<br>  $n$: materials specular reflection exponent<br>  $k_s$: surface reflection exponent<br>  Specular reflections show the color of the light source<br>  Not valid for all materials

- The Phong/Blinn lighting model
  - $I = I_a + I_s[ksN_o N_L + ks N_H]$<br>  $I_a$: surface normal<br>  $I_s$: light source direction<br>  $N$: reflection vector<br>  $N_L$: direction to the viewpoint<br>  $n$: materials specular reflection exponent<br>  $k_s$: surface reflection exponent<br>  Specular reflections show the color of the light source<br>  Not valid for all materials
Advanced shading

- In OpenGL, per-vertex Phong lighting is employed to simulate specular reflections
- But, during scan-conversion vertex color is interpolated
  - OpenGL uses Gouraud shading instead of Phong shading
  - Texture color affects highlight color
  - Sharp highlights might be missed if the surface is tessellated coarsely

Advanced shading

- To get separate highlight color highlights have to be added to the diffusely lit textured surface
  - OpenGL computes $C(C_r + C_s)$
  - Instead
    - Compute $C_r$ and $C_s$ separately
    - Modulate $C_r$ with $C_t$
    - Add result to $C_s$
  - Solution can be obtained by $\alpha$-blending

Advanced shading

- Multipass technique to get separate highlight color
  - Render the textured object without specular light $\rightarrow C_{ds}$
  - Render the untextured object with specular light only $\rightarrow C_{ss}$
  - Combine the results in the frame buffer by additive blend
    - `glBlendFunc(GL_ONE, GL_ONE)`
    - `glBlendEquation(GL_ADD)`

Advanced shading

- Using OpenGL 1.2
  - `glLightModeli(GL_LIGHT_MODEL_COLOR_CONTROL, GL_SEPARATE_SPECULAR_COLOR);`

Advanced shading

- How to get sharp highlights without adaptive refinement (tesselation)
  - Using the Blinn/Phong model specular highlights depend on $L, N$ and $V$
  - By fixing $L$ and $V$ highlights depend only on $N$
  - $N$ can be used to index into a texture map in which the reflection for $R = f(N,V)$ are stored
  - Texture map looks like reflection from chrome hemisphere
Advanced shading
- Spherical texture maps

Texture coordinates
map into sphere
map texture

Texture coordinates
computed from the
normal vector

Text contain pre-
computed Phong lighting

Advanced shading
- Phong shading with highlight texture
  - Render the object with diffuse light \( \rightarrow C_d \)
  - Render the unlit, untextured white object and modulate it
    with the highlight texture \( \rightarrow C_h \)
  - Combine the results in the frame buffer by additive blend
    \(-\) or use multi-textures in one rendering pass

Texture Mapping
- Enable Texture mapping
  - `glEnable(GL_TEXTURE_2D)`
- Texture definition
  - `glTexImage2D(target, level, int_format, w, h, border, format, type, *texels)`
- Texture filtering: Nearest Neighbor, (B)Linear
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)`
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)`

Texture Mapping
- More on textures
  - Texture coords can be modified by a texture matrix
    \(-\) 4x4 proj. matrix applied to coords bevor mapping
    \(-\) `glMatrixMode(GL_TEXTURE)`
    \(-\) Slide-projector, spot-lights

  - Textures can be named and bind to identifiers
    \(-\) Active texture is part of OpenGL state
    \(-\) Texture can be cached in texture memory
    \(-\) Calling the identifier activates the bind texture
    \(-\) Avoids reloading entire texture map

Texture Mapping
- Construct and define MipMaps
  - `gluBuild2DMipmaps()`
- Specify Minification-Filter
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST_MIPMAP_NEAREST)`
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR)`
Texture Mapping

- (New) OpenGL extensions offer extended functionality
  - The multi-texture extension (in 1.3)
  - Multiple simultaneous textures, each with own set of texture coordinates and texture environment
  - Cascaded texture lookups and blends

Texture Mapping

- Implementation of Multi-Textures
  - `glActiveTextureARB(GL_TEXTURE1_ARB)`
    - Specifies texture that is modified by subsequent calls
      - Texture environment, texture matrix ...
  - `glMultiTexCoord2fARB(GL_TEXTURE1_ARB, u, v);`
    - Specifies per-vertex texture coordinate associated with a multitexture

Texture Mapping

- Multi-Textures

Texture Mapping

- Light maps
  - Combine structure-texture with light texture
  - Re-use material with different illumination
  - Light maps for diffuse light
    - Only luminance channel
    - Low resolution (diffuse)
  - Reflectance, Irradiance, Radiosity

Texture Mapping

- Light maps
  - Varying spotlight using texture coordinates

OpenGL Extensions

- The concept of extensions
  - Naming conventions
    - ARB vs. cross vendor vs. vendor specific extensions
  - Querying:
    - compile time vs. run time
OpenGL Extensions

- Mechanism for providing access to non-standard features of the hardware
- Every vendor may define own extensions without having to ask anybody else
- Multiple vendors may cooperate to define more common extensions to ensure compatibility across platforms

Naming Conventions for Extensions

- ARB: Architecture Review Board
  - Ext. that is likely to become part of core OpenGL in the future
- EXT: Multi-vendor extensions
  - Ext. that multiple vendors have agreed upon
- NV, SGI, SUN...: vendor-specific ext.
  - Ext. supported only by one vendor, sometimes experimental and not available on all systems of that vendor (e.g. SGIS, SGIX)

Dealing with Extensions

- Since all extensions are optional, applications cannot rely on their presence
- Have to check both at compile time and at run time!
- Compile time check:
  ```
  #ifdef GL_ARB_multitexture
  // use multitexture ext. in here
  ...
  #endif
  ```

Dealing with Extensions (2)

- Run time check:
  ```
  const Glubyte *extstring=
  glutGetString(GL_EXTENSIONS);
  strcpy(exts, extstring);
  extsupported= 0;
  next= strtok(exts, " ");
  while(next) {
    if(!strcmp(next, "GL_ARB_multitexture"))
      extsupported= 1; break;
    next= strtok(exts, NULL);
  }
  ```