University of British Columbia

CPSC 414 Computer Graphics

Picking and Lighting

Week 4, Fri 26 Sep 2003

- recap: viewports and picking
- picking 2
- lighting

News

- signup for project 1 demo slots
- extra TA hours in labs
  - Fri (today) 12-1:30
  - Monday 10-2
  - Tuesday 11-1
  - Wednesday 10-1
  - Thursday 11-1
- normal lab hours
  - Fri 10-11
  - Wed 1-3

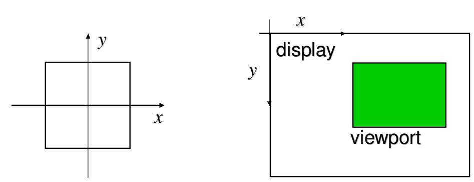
University of British Columbia

CPSC 414 Computer Graphics

Viewports and picking recap

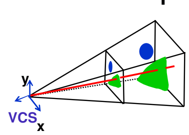
Viewport recap

- coord sys: onscreen pixels
  - determined by display/window system
  - origin often upper left

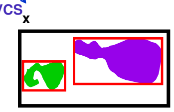


3 Picking Approaches recap

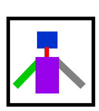
- manual ray intersection



- bounding extents



- backbuffer coloring



University of British Columbia

CPSC 414 Computer Graphics

Picking 2

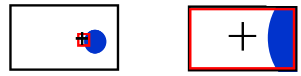
Select/Hit

- assign per-object integer keys (names)
- use small region around cursor for viewport
- redraw in special mode
- store hit list of objects in region
- examine hit list

- OpenGL support

Viewport

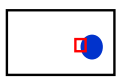
- small rectangle around cursor
  - change coord sys so fills viewport



- why rectangle instead of point?
  - people aren't great at positioning mouse
    - Fitts's Law: time to acquire a target is function of the distance to and size of the target
  - allow several pixels of slop

Viewport

- tricky to compute
  - invert viewport matrix, set up new orthogonal projection
- simple utility command
  - gluPickMatrix(x,y,w,h,viewport)
    - x,y: cursor point
    - w,h: sensitivity/slop (in pixels)
  - push old setup first, so can pop it later



Render Modes

- glRenderMode(mode)
  - GL_RENDER: normal color buffer
    - default
  - GL_SELECT: selection mode for picking
  - (GL_FEEDBACK: report objects drawn)

Name Stack

- "names" are just integers

  glInitNames()
- flat list

  glLoadName(name)
- or hierarchy supported by stack

  glPushName(name), glPopName
  - can have multiple names per object

Hierarchical Names Example

```
for(int i = 0; i < 2; i++) {
  glPushName(i);
  for(int j = 0; j < 2; j++) {
    glPushMatrix();
    glPushName(j);
    glTranslatef(i*10.0,0,j * 10.0);
      glPushName(HEAD);
      glCallList(snowManHeadDL);
      glLoadName(BODY);
      glCallList(snowManBodyDL);
      glPopName();
    glPopName();
    glPopMatrix();
  }
  glPopName();
}
```

http://www.lighthouse3d.com/opengl/picking/

Hit List

- glSelectBuffer(buffersize, *buffer)
  - where to store hit list data
- on hit, copy entire contents of name stack to output buffer.
- hit record
  - number of names on stack
  - minimum and minimum depth of object vertices
    - depth lies in the z-buffer range [0,1]
    - multiplied by 2^32 -1 then rounded to nearest int

Separate Pick Function?

- use same function to draw and pick
  - simpler to code
  - name stack commands ignored in render mode
- customize functions for each
  - potentially more efficient
  - can avoid drawing unpickable objects

Select/Hit

- advantages
  - faster
    - OpenGL support means hardware accel
    - only do clipping work, no shading or rasterization
  - flexible precision
    - size of region controllable
  - flexible architecture
    - custom code possible, e.g. guaranteed frame rate
- disadvantages
  - more complex

University of British Columbia

CPSC 414 Computer Graphics

Lighting: Illumination

Goal

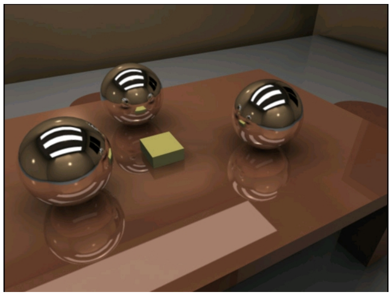
model interaction of light with matter in a way that appears realistic and is fast

- phenomenological reflection models
  - ignore real physics, approximate the look
  - simple, non-physical
  - Phong, Blinn-Phong
- physically based reflection models
  - simulate physics
  - BRDFs: Bidirectional Reflection Distribution Functions

Photorealistic Illumination

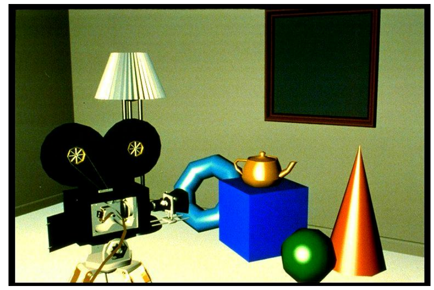
[electricimage.com]

Photorealistic Illumination



[electricimage.com]

Fast Local Illumination



Light Sources and Materials

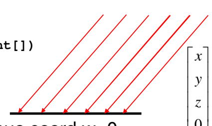
- appearance depends on
  - light sources, locations, properties
  - material (surface) properties
  - viewer position
- local illumination
  - compute at material, from light to viewer
- global illumination (later in course)
  - ray tracing: from viewer into scene
  - radiosity: between surface patches

Illumination in the Pipeline

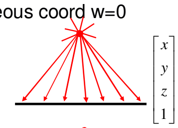
- local illumination
  - only models light arriving directly from light source
  - no interreflections and shadows
    - can be added through tricks, multiple rendering passes
- light sources
  - simple shapes
- materials
  - simple, non-physical reflection models

Light Sources

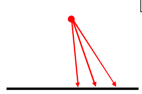
- types of light sources
  - `glLightfv(GL_LIGHT0,GL_POSITION,light[])`
  - directional/parallel lights
    - real-life example: sun
    - infinitely far source: homogeneous coord w=0



  - point lights
    - same intensity in all directions

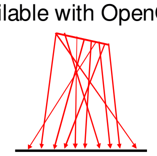


  - spot lights
    - limited set of directions:
      - point+direction+cutoff angle



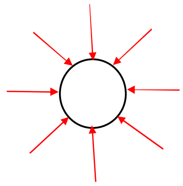
Light Sources

- area lights
  - light sources with a finite area
  - more realistic model of many light sources
  - not available with projective rendering pipeline, (i.e., not available with OpenGL)



Light Sources

- ambient lights
  - no identifiable source or direction
  - hack for replacing true global illumination
    - (light bouncing off from other objects)

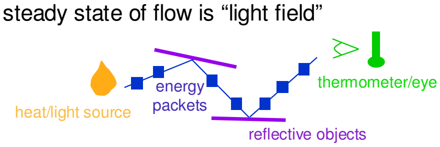


Light Sources

- geometry: positions and directions
  - standard: world coordinate system
    - effect: lights fixed wrt world geometry
    - demo: http://www.xmission.com/~nate/tutors.html
  - alternative: camera coordinate system
    - effect: lights attached to camera (car headlights)
  - points and directions undergo normal model/view transformation
- illumination calculations: camera coords

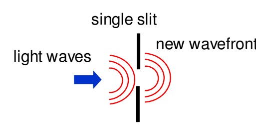
Illumination as Radiative Transfer

- radiative heat transfer approximation
  - substitute light for heat
  - treat light as packets of energy (photons)
    - ignore wavelength-dependent effects
  - model transport as flow (light transport)
  - steady state of flow is “light field”



Light Transport Assumptions

- geometrical optics (light is photons not waves)
  - no diffraction
    - diffraction example:



  - no polarization (some sunglasses)
    - light of all orientations gets through
  - no interference (packets don’t interact)
    - interference demo: http://www.falstad.com/ripple

Light Transport Assumptions II

- color approximated by discrete wavelengths
  - quantized approx of dispersion (rainbows)
  - quantized approx of fluorescence (cycling vests)
- no propagation media (surfaces in vacuum)
  - no atmospheric scattering (fog, clouds)
    - some tricks to simulate explicitly
  - no refraction (mirages)

Light Transport Assumptions III

- light travels in straight line
  - no gravity lenses
- superposition (lights can be added)
  - no nonlinear reflection models
    - nonlinearity handled separately