Picking

Week 10, Mon Mar 14

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2005
News

- some people still haven’t demoed P2
- if you don’t demo you get a 0!
  - come see me after class
- sign up with cyang@cs ASAP
  - 40112005
  - 84323013
  - 79325999
  - 81163990
Project 3

- proposal due 6pm Wed Mar 23
  - short, < 1 page of text is fine
  - need at least 1 image
    - annotated screenshot mockup
- final project due 6pm Thu Apr 7
  - face to face demos again
  - I will be grading
Project 3

- required functionality
  - 3D, interactive, lighting/shading
  - texturing, picking
- advanced functionality pieces
  - two for 1-person team
  - four for 2-person team
  - six for 3-person team
P3: Advanced Functionality

- (new) navigation
- procedural modelling/textures
  - particle systems
- collision detection
- simulated dynamics
- level of detail control
- advanced rendering effects
- on-screen control panel (HUD)
- using motion capture data
- whatever else you want to do – check with us!
Review: Radiosity

- conserve light energy in a volume
  - model light transport until convergence
  - solution captures diffuse-diffuse bouncing of light
- view independent technique
  - calculate solution for entire scene offline
  - browse from any viewpoint in realtime
Review: Radiosity

- divide surfaces into small patches
- loop: check for light exchange between all pairs
  - form factor: orientation of one patch wrt other patch (n x n matrix)
Review: Image-Based Rendering

- store and access only pixels
  - no geometry, no light simulation, ...
  - input: set of images
  - output: image from new viewpoint
    - surprisingly large set of possible new viewpoints
Review: Image As Signal

- 1D slice of raster image
  - discrete sampling of 1D spatial signal
- theorem
  - any signal can be represented as an (infinite) sum of sine waves at different frequencies

Examples from Foley, van Dam, Feiner, and Hughes
Review: Summing Waves
Review: Summing Waves II

represent spatial signal as sum of sine waves (varying frequency and phase shift) very commonly used to represent sound “spectrum”
Review: 1D Sampling and Reconstruction

- problems
  - jaggies – abrupt changes
  - lose data
Review: Sampling Theorem and Nyquist Rate

- Shannon Sampling Theorem
  - continuous signal can be completely recovered from its samples iff sampling rate greater than twice maximum frequency present in signal
- sample past Nyquist Rate to avoid aliasing
  - twice the highest frequency component in the image’s spectrum

![Figure 14.17 Sampling below the Nyquist rate. (Courtesy of George Wolberg, Columbia University.)](image)
Review: Aliasing

- incorrect appearance of high frequencies as low frequencies
- to avoid: antialiasing
  - supersample
    - sample at higher frequency
  - low pass filtering
    - remove high frequency function parts
    - aka prefiltering, band-limiting
Correction & Review: Supersampling
Review: Low-Pass Filtering
Picking
Reading

- Red Book
  - Selection and Feedback Chapter
    - all
  - Now That You Know Chapter
    - only Object Selection Using the Back Buffer
Interactive Object Selection

- move cursor over object, click
  - how to decide what is below?
- ambiguity
  - many 3D world objects map to same 2D point
- four common approaches
  - manual ray intersection
  - bounding extents
  - backbuffer color coding
  - selection region with hit list
Manual Ray Intersection

- do all computation at application level
  - map selection point to a ray
  - intersect ray with all objects in scene.
- advantages
  - no library dependence
Manual Ray Intersection

- do all computation at application level
  - map selection point to a ray
  - intersect ray with all objects in scene.

- advantages
  - no library dependence

- disadvantages
  - difficult to program
  - slow: work to do depends on total number and complexity of objects in scene
Bounding Extents

- keep track of axis-aligned bounding rectangles

- advantages
  - conceptually simple
  - easy to keep track of boxes in world space
Bounding Extents

- disadvantages
  - low precision
  - must keep track of object-rectangle relationship

- extensions
  - do more sophisticated bound bookkeeping
Backbuffer Color Coding

- use backbuffer for picking
  - create image as computational entity
  - never displayed to user
- redraw all objects in backbuffer
  - turn off shading calculations
  - set unique color for each pickable object
    - store in table
  - read back pixel at cursor location
    - check against table
Backbuffer Color Coding

- **advantages**
  - conceptually simple
  - variable precision

- **disadvantages**
  - number of color bits must be adequate
  - introduce 2x redraw delay
for(int i = 0; i < 2; i++)
    for(int j = 0; j < 2; j++) {
        glPushMatrix();
        switch (i*2+j) {
            case 0: glColor3ub(255,0,0);break;
            case 1: glColor3ub(0,255,0);break;
            case 2: glColor3ub(0,0,255);break;
            case 3: glColor3ub(250,0,250);break;
        }
        glTranslatef(i*3.0,0,-j * 3.0)
        glCallList(snowman_display_list);
        glPopMatrix();
    }

glColor3f(1.0f, 1.0f, 1.0f);
for(int i = 0; i < 2; i++)
    for(int j = 0; j < 2; j++) {
        glPushMatrix();
        glTranslatef(i*3.0,0,-j * 3.0);
        glColor3f(1.0f, 1.0f, 1.0f);
        glCallList(snowman_display_list);
        glPopMatrix();
    }

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

- use small region around cursor for viewport
- assign per-object integer keys (names)
- 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
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

```c
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);
        glPushMatrix();
        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
Integrated vs. Separate Pick Function

- **Integrate**: use same function to draw and pick
  - simpler to code
  - name stack commands ignored in render mode
- **Separate**: 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
Hybrid Picking

- select/hit approach: fast, coarse
  - object-level granularity
- manual ray intersection: slow, precise
  - exact intersection point
- hybrid: both speed and precision
  - use select/hit to find object
  - then intersect ray with that object
OpenGL Picking Hints

- **gluUnproject**
  - transform window coordinates to object coordinates given current projection and modelview matrices
  - use to create ray into scene from cursor location
  - call gluUnProject twice with same (x,y) mouse location
    - \( z = \text{near: } (x,y,0) \)
    - \( z = \text{far: } (x,y,1) \)
    - subtract near result from far result to get direction vector for ray
  - use this ray for line/polygon intersection