Procedural Approaches II, Picking

Week 10, Wed Mar 21

News

- showing up for your project grading slot is **not** optional
  - 5 people have missed their slot, without notifying the TA in advance of the need to change
  - 2% penalty for no-shows for P3 and P4
Review: Environment Mapping

- cheap way to achieve reflective effect
  - generate image of surrounding
  - map to object as texture
- sphere mapping: texture is distorted fisheye view
  - point camera at mirrored sphere
  - use spherical texture coordinates
Review: Cube Environment Mapping

- 6 planar textures, sides of cube
  - point camera outwards to 6 faces
  - use largest magnitude of vector to pick face
  - other two coordinates for (s,t) texel location
Review: Volumetric Texture

- define texture pattern over 3D domain - 3D space containing the object
  - texture function can be digitized or **procedural**
  - for each point on object compute texture from point location in space
- 3D function $\rho(x,y,z)$

```python
function marble(point)
    x = point.x + turbulence(point);
    return marble_color(sin(x))
```
Review: Perlin Noise

- coherency: smooth not abrupt changes
- turbulence: multiple feature sizes
Review: Generating Coherent Noise

• just three main ideas
  • nice interpolation
  • use vector offsets to make grid irregular
  • optimization
    • sneaky use of 1D arrays instead of 2D/3D one
Review: Procedural Modeling

- textures, geometry
  - nonprocedural: explicitly stored in memory
- procedural approach
  - compute something on the fly
    - not load from disk
  - often less memory cost
- visual richness
  - adaptable precision
- noise, fractals, particle systems
Procedural Approaches II
Fractal Landscapes

- fractals: not just for “showing math”
  - triangle subdivision
  - vertex displacement
  - recursive until termination condition

http://www.fractal-landscapes.co.uk/images.html
Self-Similarity

• infinite nesting of structure on all scales
Fractal Dimension

- $D = \frac{\log(N)}{\log(r)}$
- $N = \text{measure}$, $r = \text{subdivision scale}$
- Hausdorff dimension: noninteger

- Koch snowflake

$D = \frac{\log(4)}{\log(3)} = 1.26$

http://www.vanderbilt.edu/AnS/psychology/cogsci/chaos/workshop/Fractals.html
Language-Based Generation

- L-Systems: after Lindenmayer
  - Koch snowflake: F :- FLFRRFLF
    - F: forward, R: right, L: left
  - Mariano’s Bush:
    F=FF-[-F+F+F]+[+F-F-F] }
    - angle 16

http://spanky.triumf.ca/www/fractint/lsys/plants.html
1D: Midpoint Displacement

- divide in half
- randomly displace
- scale variance by half

http://www.gameprogrammer.com/fractal.html
2D: Diamond-Square

- fractal terrain with diamond-square approach
  - generate a new value at midpoint
  - average corner values + random displacement
  - scale variance by half each time
Particle Systems

• loosely defined
  • modeling, or rendering, or animation
• key criteria
  • collection of particles
  • random element controls attributes
    • position, velocity (speed and direction), color, lifetime, age, shape, size, transparency
    • predefined stochastic limits: bounds, variance, type of distribution
Particle System Examples

- objects changing fluidly over time
  - fire, steam, smoke, water
- objects fluid in form
  - grass, hair, dust
- physical processes
  - waterfalls, fireworks, explosion
- group dynamics: behavioral
  - birds/bats flock, fish school, human crowd, dinosaur/elephant stampede
Particle Systems Demos

• general particle systems
  • http://www.wondertouch.com

• boids: bird-like objects
  • flocking/swarming behavior
  • procedural motion
  • http://www.red3d.com/cwr/boids/
Particle Life Cycle

• generation
  • randomly within “fuzzy” location
  • initial attribute values: random or fixed
• dynamics
  • attributes of each particle may vary over time
    • color darker as particle cools off after explosion
  • can also depend on other attributes
    • position: previous particle position + velocity + time
• death
  • age and lifetime for each particle (in frames)
  • or if out of bounds, too dark to see, etc
Particle System Rendering

- expensive to render thousands of particles
- simplify: avoid hidden surface calculations
  - each particle has small graphical primitive (blob)
  - pixel color: sum of all particles mapping to it
- some effects easy
  - temporal anti-aliasing (motion blur)
    - normally expensive: supersampling over time
    - position, velocity known for each particle
    - just render as streak
Procedural Approaches Summary

• Perlin noise
• fractals
• L-systems
• particle systems

• not at all a complete list!
  • big subject: entire classes on this alone
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

• 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
    - first level: box check.
    - second level: object check
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**
  - introduce 2x redraw delay
  - backbuffer readback very slow
glColor3f(1.0f, 1.0f, 1.0f);
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();
    }

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’ Law: time to acquire a target is function of the distance to and size of the target
  • allow several pixels of slop
Viewport

• nontrivial 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

• again, "names" are just integers
  glInitNames()
• flat list
  glLoadName(name)
• or hierarchy supported by stack
  glPushName(name), glPopName
  • can have multiple names per object
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
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 acceleration
    - avoid shading overhead
  - 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 Precision 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 = near: (x,y,0)
    - z = far: (x,y,1)
    - subtract near result from far result to get direction vector for ray
  - use this ray for line/polygon intersection
Picking and P4

- you must implement true 3D picking!
  - you will not get credit if you just use 2D information