EECS 487

January 22, 2007

visibility

- issue: triangle visibility
- solution in p1: “painter’s algorithm”
  - draw Δ’s back to front
  - p1 uses heuristic to sort Δ’s (can see artifacts: demo)
  - Graphics cards: use depth buffer

todo

- special assignment sign-up
- last call: this friday
- if you haven't already, do it now in class
- any volunteers for wednesday?

jot/mlib

- for math intensive stuff (e.g. line clipping), use jot/mlib
- online documentation (see proj1 page)
- read header files
  – mlib/point2.H
  – mlib/vec2.H
  – mlib/points.H
- see: proj1/jot-tips.C

blackboard

computing 2D barycentric coordinates using jot mlib

If **a**, **b**, **c** are triangle vertices, **p** is the pixel you are evaluating (all in PIXEL coordinates), then the barycentric coordinates are:

α = det(**b**-**p**,**c**-**p**)/det(**b**-**a**,**c**-**a**);

β = det(**p**-**a**,**c**-**a**)/det(**b**-**a**,**c**-**a**);

γ = det(**b**-**a**,**p**-**a**)/det(**b**-**a**,**c**-**a**);

OpenGL introduction

- name:
  - 1980’s: SGI’s graphics library: GL
  - basis for OpenGL API
- client/server model
- manages resources (GPU)
  - framebuffer
  - processor
  - memory
- API similar to DirectX

not included

- integration with media (e.g. audio, video…)
- UI widgets, windows, input handling
- high-level objects
  (e.g. meshes, scene graph)
- advanced algorithms, e.g.:
  - collision detection
  - physics
  - global illumination

what *does* it do?

- render 2D and 3D primitives or images
- rasterize Δ’s
- lighting (local illumination)
- texture mapping
- visibility

can add missing stuff

- UI widgets, windows, input handling:
  - via GLUT (as in discussions)
    - standard library
    - now freeglut
  - via GLUI (as in jot)
    - written by ex-Ph.D. student Paul Rademacher
- Meshes, scene graphs:
  - OpenSceneGraph
  - G3D
  - jot

typical program

- create window
- loop:
  - clear frame buffer
  - set state
  - setup lights, camera…
  - draw primitives
  - swap buffers

primitive processing pipeline

1. vertex processing
   - transformations: 3D → 2D
   - lighting
2. clipping, primitive assembly
3. fragment processing
   - rasterize primitives
   - interpolate colors, texture coordinates, etc.
4. fragment text
   - depth, alpha

newly programmable parts

1. vertex processing
   – transformations: 3D → 2D
   – lighting
2. clipping, primitive assembly
3. fragment processing
   – rasterize primitives
   – interpolate colors, texture coordinates, etc.
4. fragment text
   – depth, alpha

evolution of OGL

- originally pipeline was “fixed”
  - could set values, enable/disable state, etc.
- additions to OGL thru “extensions”
- extensions not guaranteed part of API
  - e.g., just supported by 1 vendor, or ARB
  - may be incorporated into later OGL version

evolution of programmable OGL

- originally extensions to bypass fixed-functionality pipeline
- low-level assembly code, e.g.:

```
 "!!VP1.0\
DP4 R0.x,v[OPOS],c[0];\
DP4 R0.y,v[OPOS],c[1];\
DP4 R0.z,v[OPOS],c[2];\
DP4 R0.w,v[OPOS],c[3];\
DP4 R1.x,R0,c[4];\
DP4 R1.y,R0,c[5];\
…
```

Now: GLSL

```
void main() {
   // compute the vertex normal and position in eye
  coordinates:
   N = gl_NormalMatrix * gl_Normal;
   P = gl_ModelViewMatrix * gl_Vertex;

   // output vertex position in clip coordinates
   gl_Position = ftransform();
}
```

GLSL relatively new

- available as extensions in OpenGL 1.5
- part of OpenGL 2.0
  - for project 2: work in newer labs!

Example code

```
// initialize window here…

// clear frame buffer
glClearColor(0,0,0,0);
glClear(GL_COLOR_BUFFER_BIT);

// set up camera
glOrtho(0,1,0,1,-1,1);
```

example, cont’d

```
// set current color
glColor3f(1,1,0);

// draw points
glPointSize(2.0); // must be before glBegin()
glBegin(GL_POINTS);
glVertex3f(x0,y0,z0);
glVertex3f(x1,y1,z1);
…
glEnd();
```

primitives

```
GL_POINTS
GL_LINES
GL_LINE_STRIP
GL_TRIANGLES
GL_TRIANGLE_STRIP
```

...

(see blackboard, or red book)

more example

```
// draw triangles
glBegin(GL_TRIANGLE_STRIP);
glColor3fv(c0);
glNormal3fv(n0);
glVertex3fv(v0);
glColor3fv(c1);
glNormal3fv(n1);
glVertex3fv(v1);
…
glEnd();
```

reading

- OpenGL programming guide (“red book”)
  - chapters 1 and 2

next up

- OpenGL lighting
- GLSL programming
- 2D and 3D transforms
-
- SIGGRAPH 2001 slides on OpenGL...