EECS 487

October 25, 2006

- James Irizarry presentation
- SKETCH video
- project 3 concepts
- how to transform a normal vector
- why is n·l correct for diffuse shading?

JOT: flat scene graph

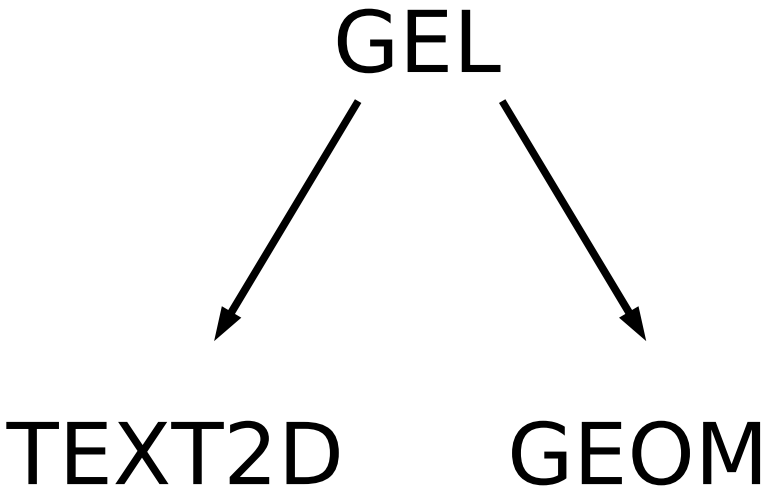
basic type for “geometric elements”: GEL

“scene graph” is just a list of GELs.

each frame:

```
  for (int i=0; i<gels.num(); i++)
      gels[i]->draw();
```

Derived types



TEXT2D: 2D text displayed in the window

GEOM: Sub-class of GEL that has a mesh, and a transform

Object space, world space

The transform maps from object space to world space

E.g. a chair model defined near the origin, aligned to major axes (in object space)

To place the chair somewhere in the world, apply a transform to translate, rotate, or scale the shape

GEOM::draw()

```
GEOM::draw() {
    push current matrix (saves it)
    multiply current matrix by xform
    draw mesh
    pop matrix (restores old matrix)
}
```

BMESH delegates to Patch...

```
BMESH::draw() {
    for each patch p
        p->draw();
}
```

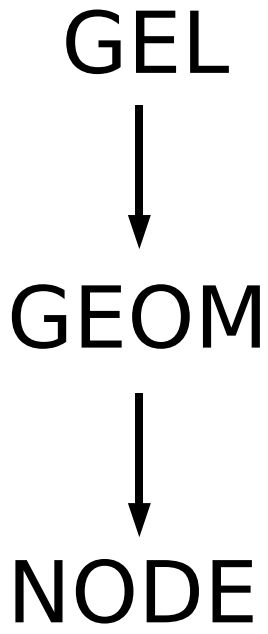
Patch delegates to GTexture...

```
Patch::draw() {
    find GTexture g matching the name
      of the current rendering style
    g->draw();
}
```

```
project 2, shaders.H defines
  GTextures used in project 2
```

project 3: nested scene graph

Project 3 uses a subclass of GEL called NODE that supports a nested scene graph:



NODE

Each NODE has:

transform and BMESH (from GEOM)

list of children NODES

pointer to parent NODE

NODE

For a GEOM, the transform maps from object space to world space

For a NODE, the transform maps from object space to its parent's object space

If A is the parent of B, and B is the parent of C, then object-to-world transform for C is:

```
A.xform() * B.xform() * C.xform()
```

NODE::draw()

```
NODE::draw() {
    push current matrix (saves it)
    multiply current matrix by xform
    draw mesh
    draw each child // new in NODE
    pop matrix (restores old matrix)
}
```

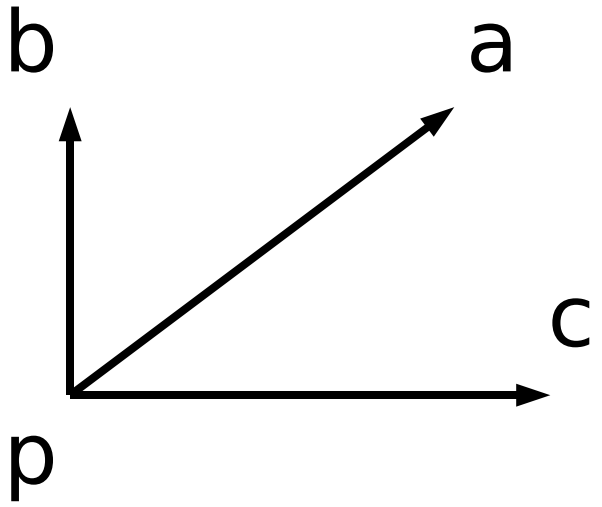
OpenGL matrix stack

```
Draw A:
    push matrix A on stack
    draw A's mesh
    Draw B:
        push matrix B on stack
        draw B's mesh
        ...
        pop matrix from stack
    pop matrix from stack
```

p3: sketching primitives

- Like SKETCH, small number of primitives
  - “cube”
  - cylinder
  - optional: extrude, duct, ...
- Based on user-drawn axes

Cube primitive



Strokes matching 3 perpendicular axes,
ordered so b ⊥ to surface at p, and
vectors {a,b,c} are right-handed

The transform for new cube

map origin to p, and
canonical axes {x, y, z} to {a, b, c}:

M = Translate(p) * [a,b,c]

But M maps object space to world space.
The new cube exists as a child of its parent, which has its *own* transform...

Cube transform, cont'd

Let P be the parent object-to-world transform

Let M' be the matrix we should assign to the cube.

Then: $P * M' = M$

so: $M' = P^{-1} * M$

Cube transform, cont'd

Q: what about scaling?

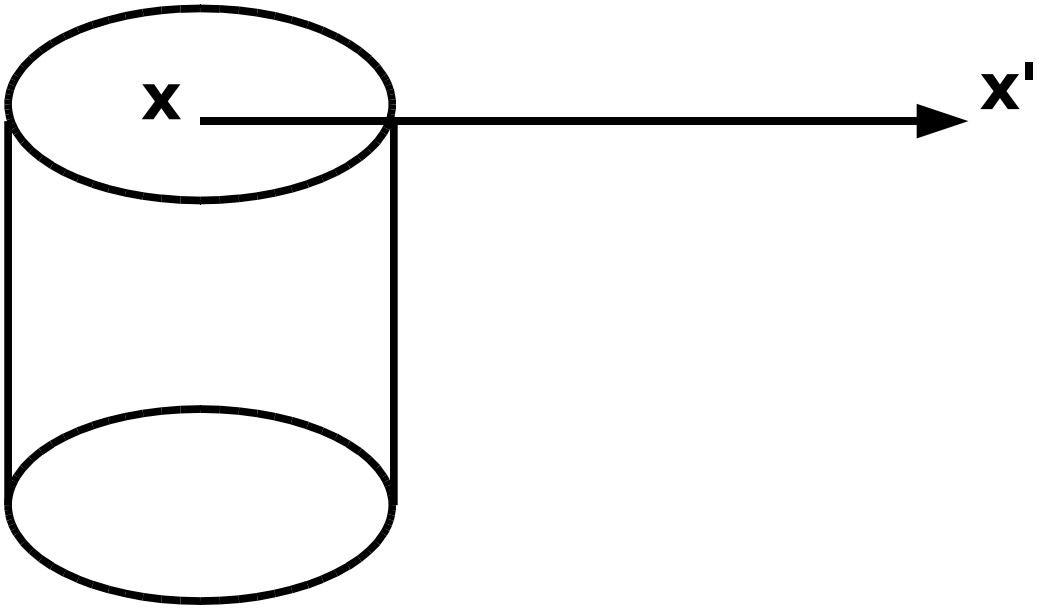
Cube transform, cont'd

Q: What about scaling?

A: It's built-in.

Translation: plane constraint

User clicks with middle button, drags



map x and x' to world space w, w'

translation is: w' – w

Translation: plane constraint

```
Wpt p;          // point in plane (world space)
Wvec n;         // plane normal (world space)
XYpt x;         // screen point
Wline R(x);     // ray into scene at x

// find ray intersection with plane:
x = Wplane(p,n).intersect(R);
```

Translation: line constraint

```
Wpt p;          // point on line (world space)
Wvec n;         // line direction (world space)
XYpt x;         // screen point
Wline R(x);     // ray into scene at x

// find ray intersection with line:
x = Wline(p,n).intersect(R);
```

Q: How to find the intersection of lines in 3D?

Q: How to set transform?

Translation: line constraint

Q: How to find the intersection of lines in 3D?

```
x = Wline(p,n).intersect(R);
```

A: The above finds the point on the constraint line that is closest to line R

Q: How to set transform?

A: Find w, w' in *parent's object space.*
Then replace node's transform M with TM
(T is the translation from w to w')

transforming normals (board)

Diffuse shading: hack or physically based?

Why is n • l the right number to use for diffuse shading (aka lambertian shading) (board)

Curves

Next: Chapter 15 in the text