GL-Hijacking: Reconstructing from Streams for Stylized Rendering

Alex Mohr
Mike Gleicher
Department of Computer Sciences
University of Wisconsin-Madison
http://www.cs.wisc.edu/graphics

The Problem…

- Lots of applications
- Lots of features that we want to try
- O(n^2) things to try, O(1) labor

- Want to try stylized rendering (various types) in *real applications*
  - Not just toy apps!

The solution

- Alex Mohr

(he's spending the summer at Pixar, so I get to give the talk)

- Not afraid of the low-level engineering
- Single-handedly responsible for all the engineering miracles

- Chris Hermann wrote the Toon and Colored pencil renderers

The project history

- Spring ’99 NPRQuake (CS838 project)
  - Hack Quake sources for plug-in renderers
- Fall ’00 GlHijacker 1
  - Engineering proof of concept
  - Simple stream transforms
- Fall ’01 GlHijacker 2
  - Geometry reconstruction
  - Other people write renderers
- End of Project
  - Alex loses interest
  - Next step requires too much engineering

Synopsis

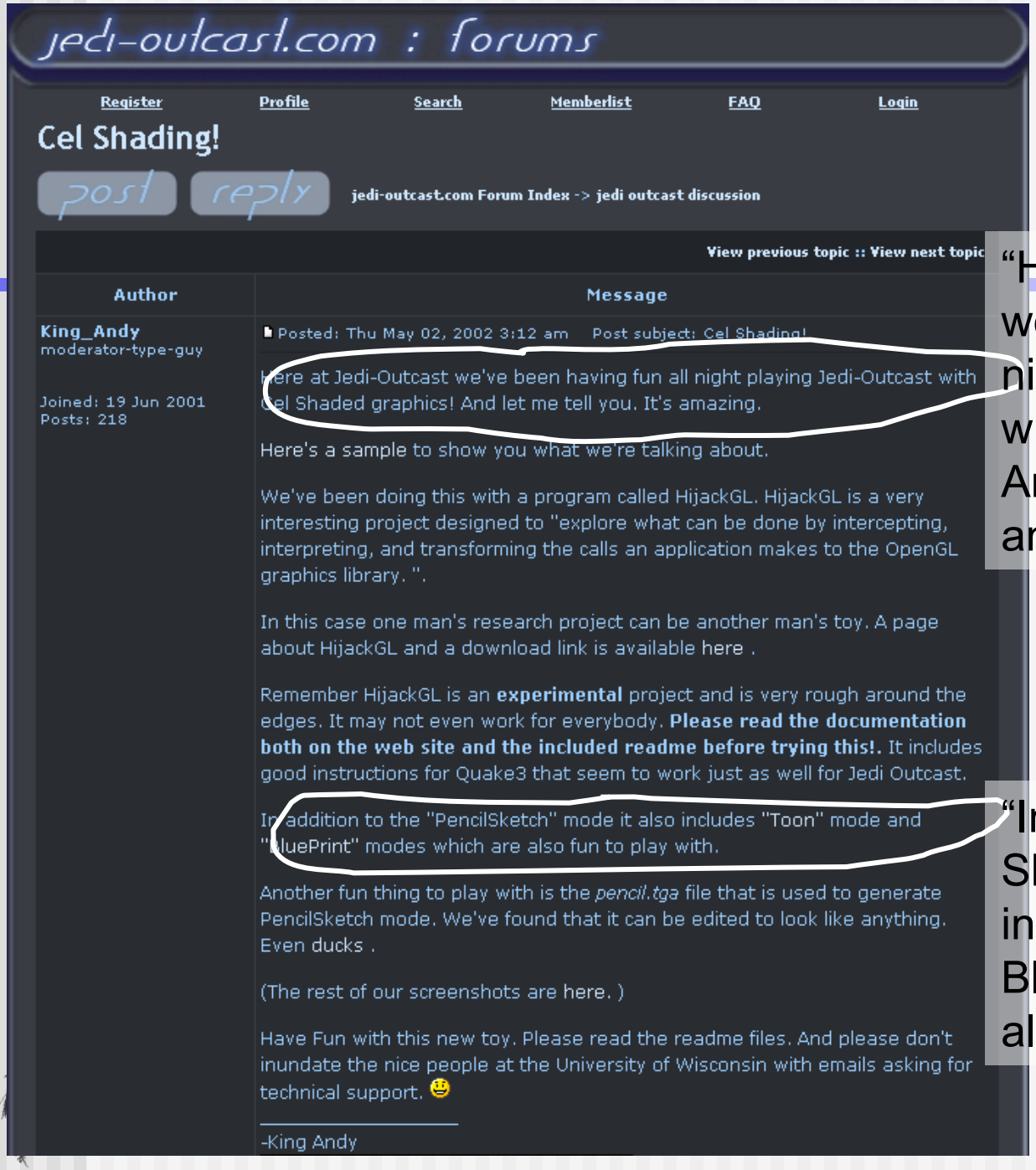
- Intercept all calls to the low-level graphics library (OpenGL)
  - Needs common API to have apps to hijack!
- Build a useful intermediate representation
  - Need what is needed for features/renderers
- Plug-in interesting features

Yes! You can download it!

http://www.cs.wisc.edu/graphics

go to the gallery page…

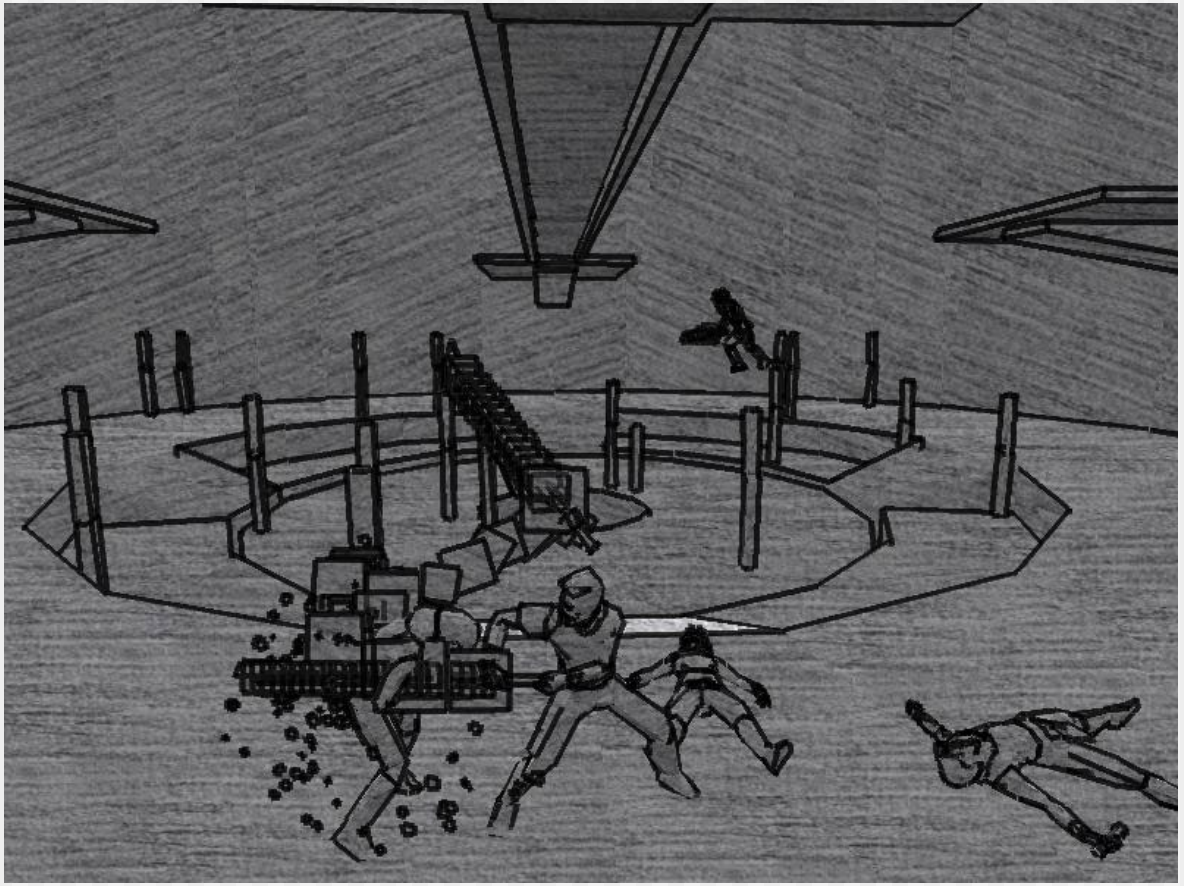
Pencil-Sketch Light Saber Fight?



"Here at Jedi-Outcast we've been having fun all night playing Jedi-Outcast with Cel Shaded Graphics. And let me tell you. It's amazing."

"In addition to the "Pencil Sketch" mode it also includes Toon mode and Blueprint modes which are also fun to play with.."

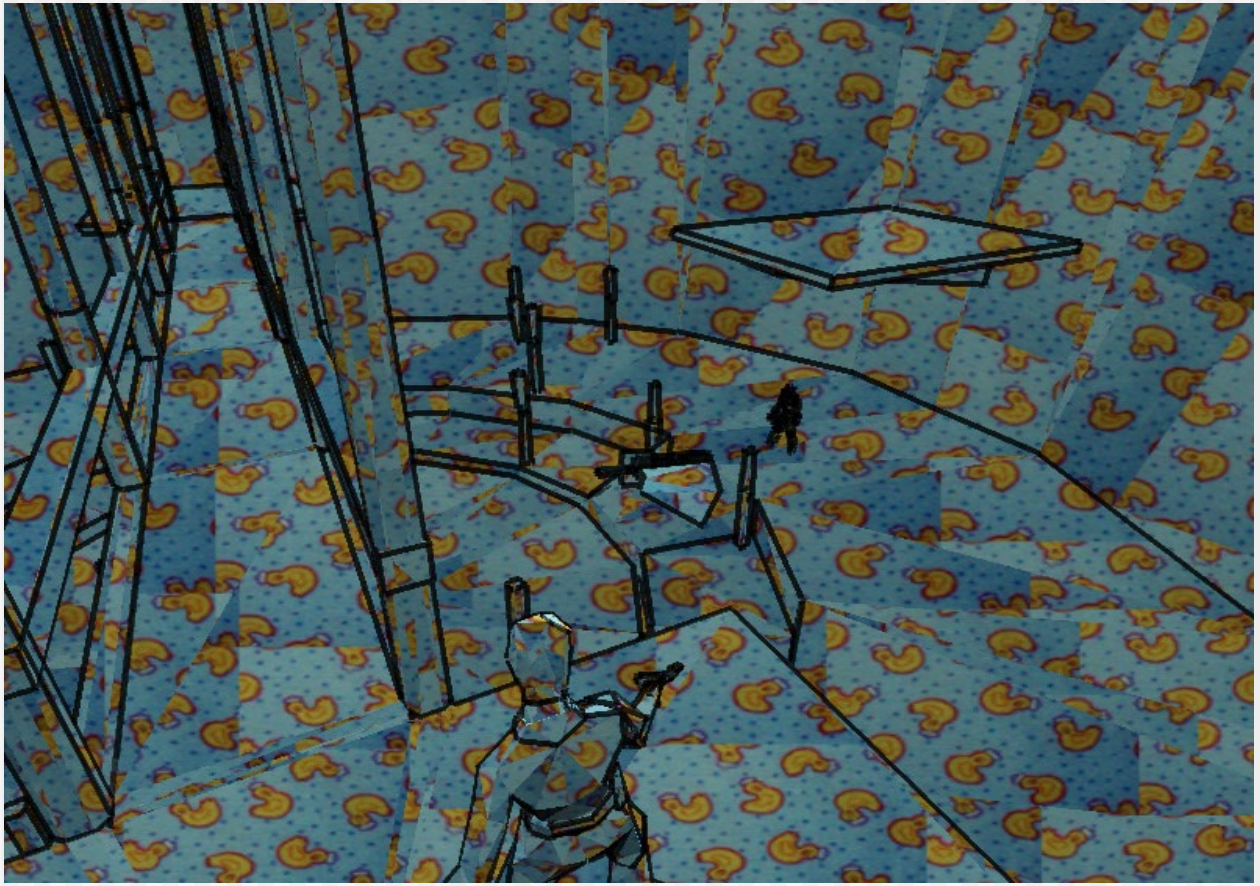
Not too bad for real-time…



Note:

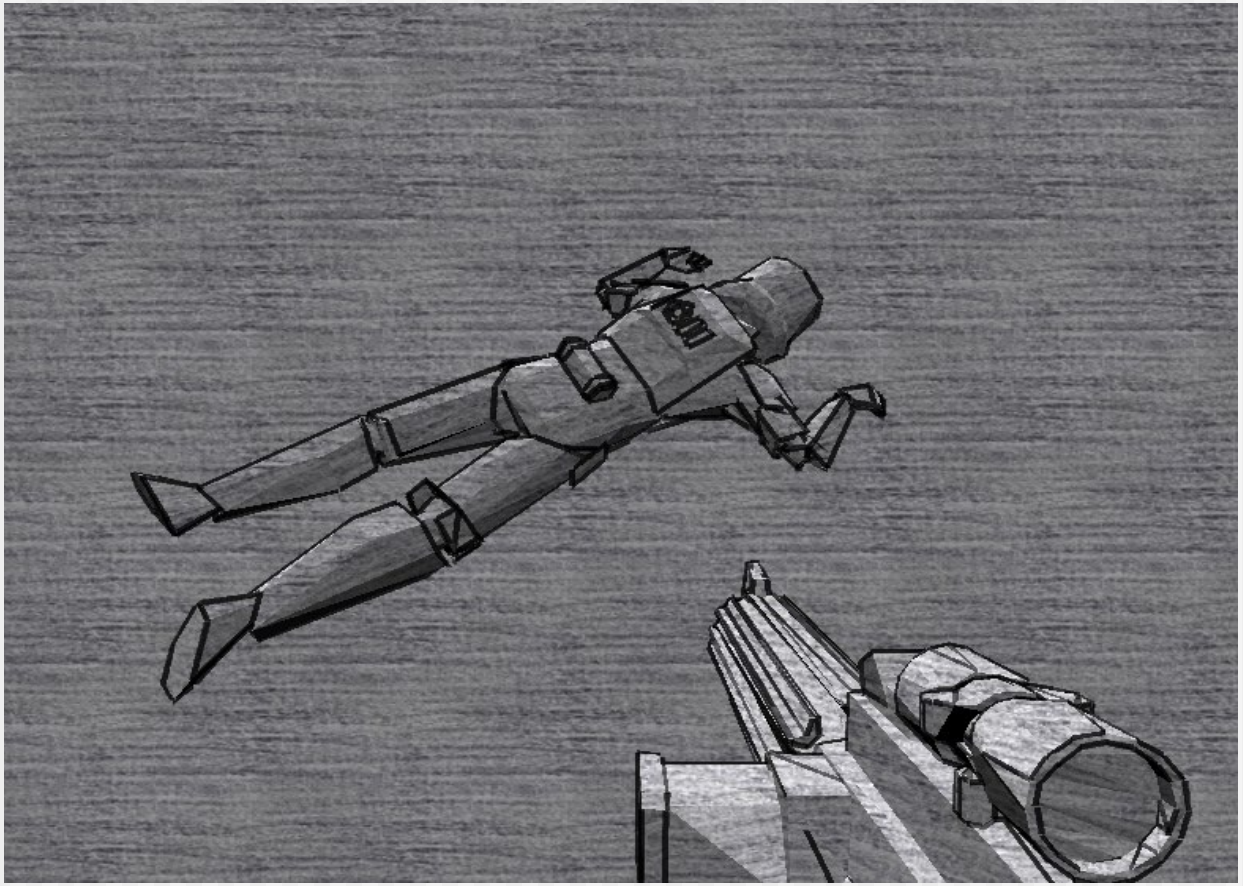
- We do not even have this program
  - We only found out about it from our weblogs!
- “Real people” are trying something
- “Real application” - people really use
  - Much richer than our virtual environments
- And they’re doing weird things with it…

There's no accounting for taste…

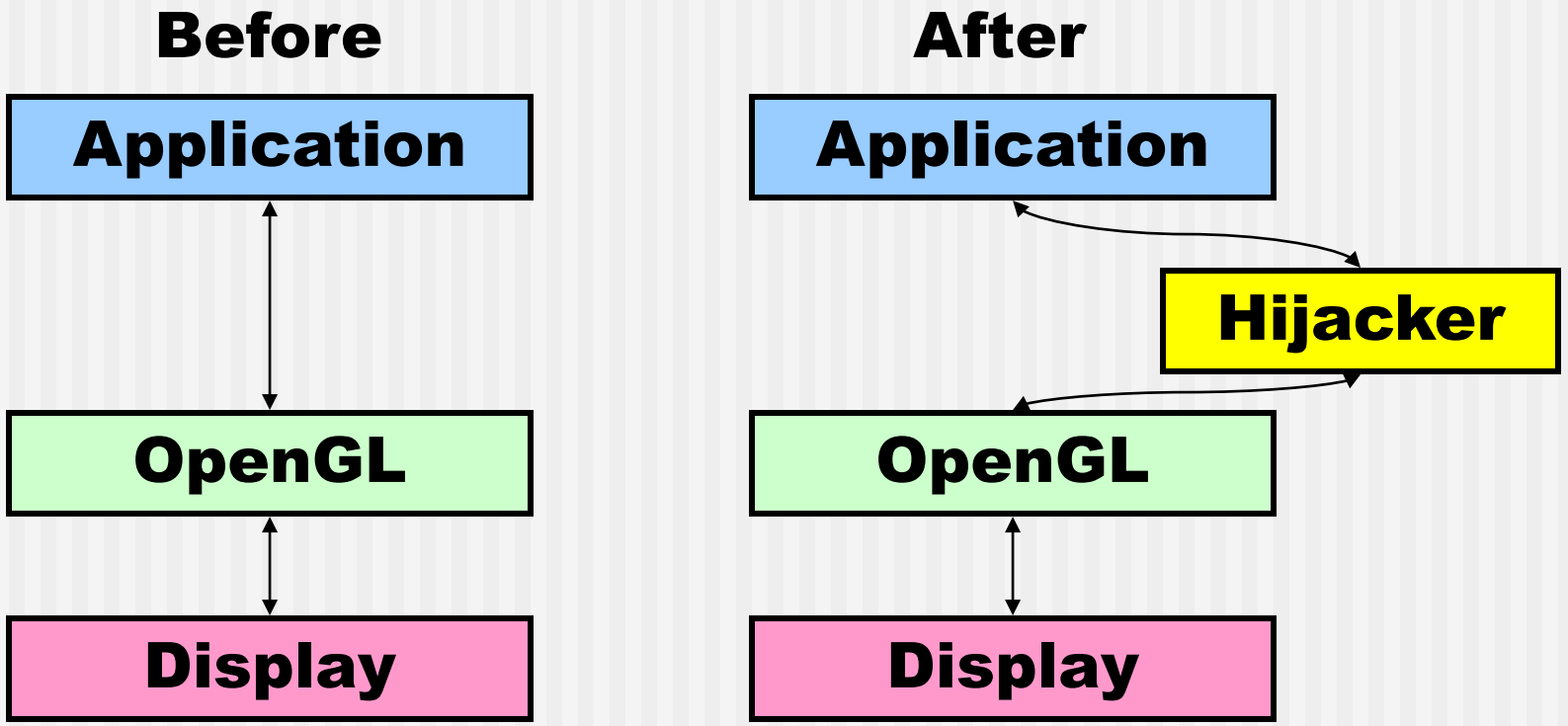


The Jedi-Outcast folks did this, not us!

Is this less painful in pencil sketch?



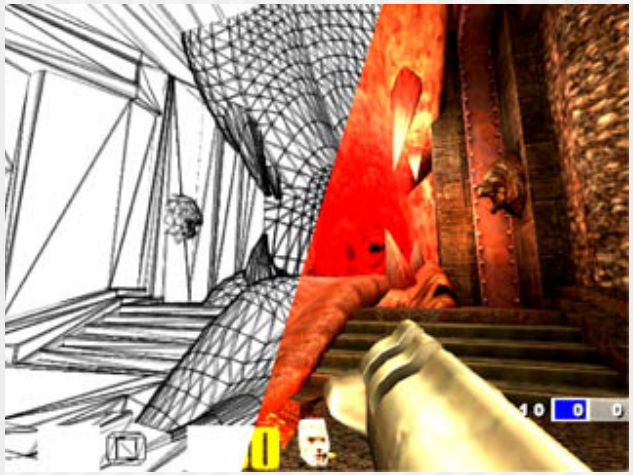
GL-Hijacking: the easy part…

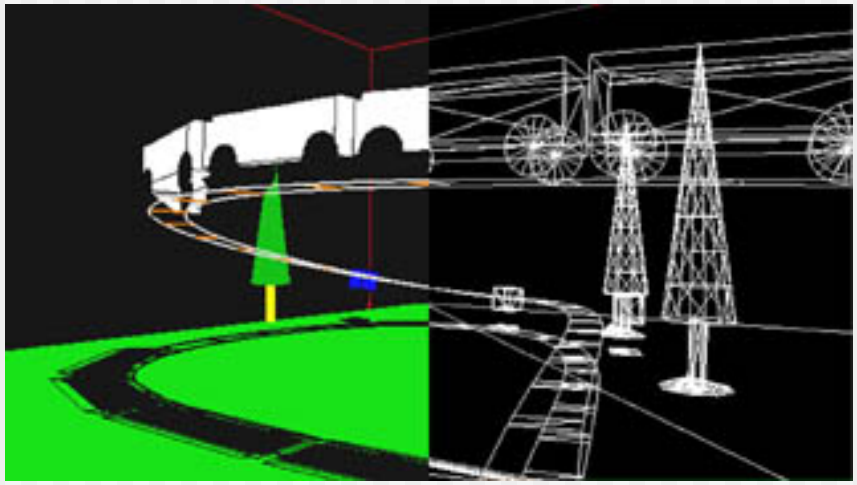


- Intercept call stream (replace shared lib)
- Lots of engineering, others do this too

What can you do with this?
Not too much (nothing interesting)

- Command stream transformations
  - Mohr&Gleicher ‘01, Chromium ‘02
- Simple techniques
  - Perturb data, global state changes





The Problem:

- Stream tells you *how* to draw
  - Procedural, imperative, performance tricks, …
  - Ordered, and order dependent
- Care about *what* was drawn
  - Declarative, geometrical
  - Data structures describing objects
  - Winged edges, object lists, …

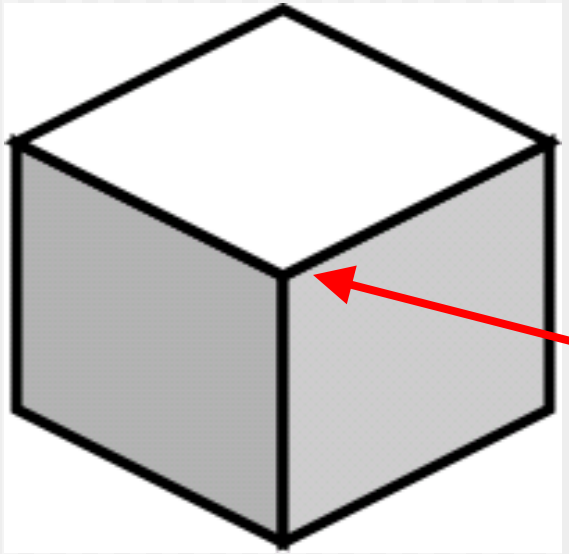
- All we get is a trace of the program!

Example problem:

- How to draw a cube?
  - 8 quads?
  - 16 triangles?
  - 24 triangles?
  - 1 triangle strip?
  - 1 vertex array?
  - 1 plane that is transformed?
  - 1 point that is transformed differently?
  - 1 point in place, camera moves around it?
- And you want me to tell you what edge is a silhouette?
  - Edges are never really represented in OpenGL

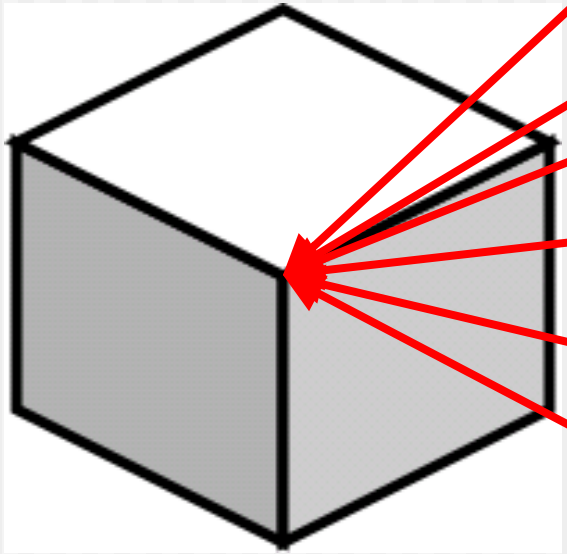
Drawing a cube

Where is this point in the code?



```
glNormal3f(0,1,0);
glBegin(GL_TRIANGLE);
glVertex3f(0,1,0);
glVertex3f(1,1,0);
glVertex3f(1,1,1);
glEnd();
```

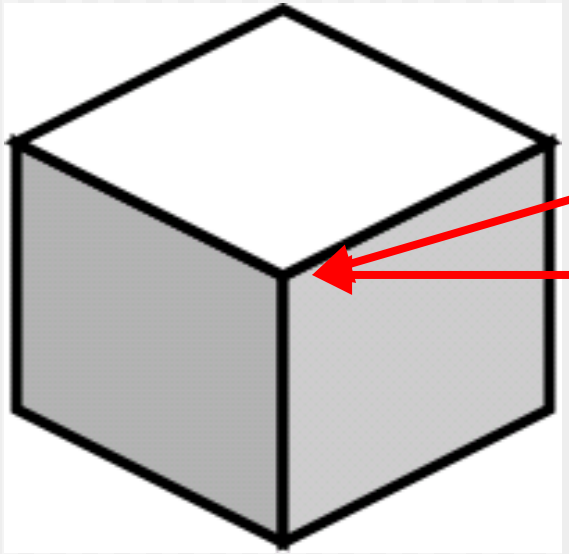
Drawing a cube



```
glNormal3f(0,1,0);
glBegin(GL_TRIANGLE);
glVertex3f(0,1,0);
glVertex3f(1,1,0);
glVertex3f(1,1,1);
glEnd();
glBegin(GL_TRIANGLE);
glVertex3f(1,1,1);
glVertex3f(0,1,1);
glVertex3f(0,1,0);
glEnd();
glNormal3f(0,0,-1);
glBegin(GL_TRIANGLE);
glVertex3f(0,1,0);
glVertex3f(1,1,0);
glVertex3f(1,0,0);
glEnd();
glBegin(GL_TRIANGLE);
glVertex3f(1,0,0);
glVertex3f(0,1,0);
glVertex3f(0,0,0);
glEnd();
glNormal3f(-1 ,0, 0);
glBegin(GL_TRIANGLE);
glVertex3f(0,0,0);
glVertex3f(0,1,1);
glVertex3f(0,1,0);
glEnd();
glBegin(GL_TRIANGLE);
glVertex3f(0,1,1);
glVertex3f(0,1,0);
glVertex3f(0,0,1);
glEnd();
```

Drawing a cube

Where is this point in the code?



```
drawFace() {
    glNormal3f(0,1,0);
    glBegin(GL_TRIANGLE);
    glVertex3f(0,0,0);
    glVertex3f(1,0,0);
    glVertex3f(1,0,1);
    glVertex3f(0,0,0);
    glVertex3f(1,0,1);
    glVertex3f(0,0,1);
    glEnd();
}
```

Goal: Construct from stream

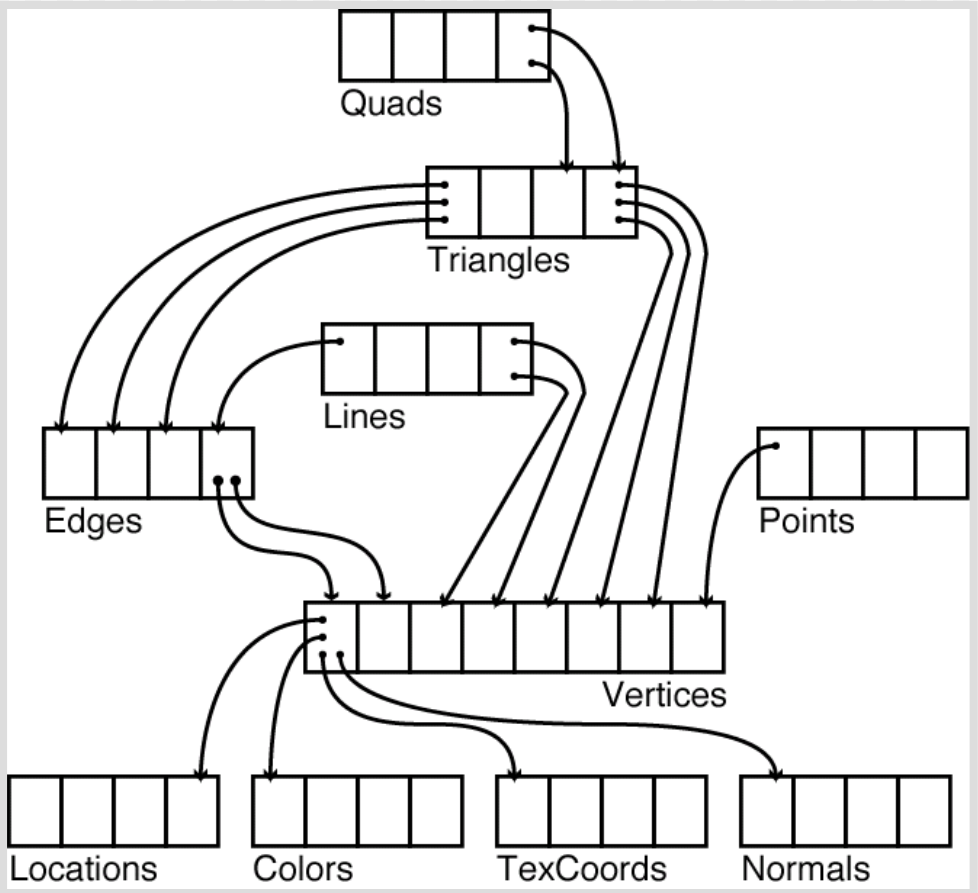
- Want to end up with a “nice” object representation (points, edges, faces, …)
- Need this for most manipulations

- Original program may never have had it!
- Geometry is implicit in code
  - Need to dig it out of execution stream!

Problem:
We need the programmer's intentions!

- We're guessing what program does
- Are you really drawing that quad?
  - Or is it a piece of text with a menu?
  - Or are you playing a weird trick in the stencil buffer?
- Make some "reasonable" assumptions
  - Could add heuristics

Build a geometric representation



Note:

GL doesn't have a notion of a "point."

A vertex is a temporal element. Many GL vertices might be the same point.

How to do this

- Spatial hashing
  - Find points that are the same in world-space
  - Avoids transform issues
  - Minor issue that GL doesn’t have “world”
- Creates geometry that would generate the same GL stream

Now what?

- We have a “standard” geometric data structure – just like any app
- Augment with some analysis
  - Silhouette edge detection
  - Fake normals and lighting
- Pass it to a plug-in “renderer”

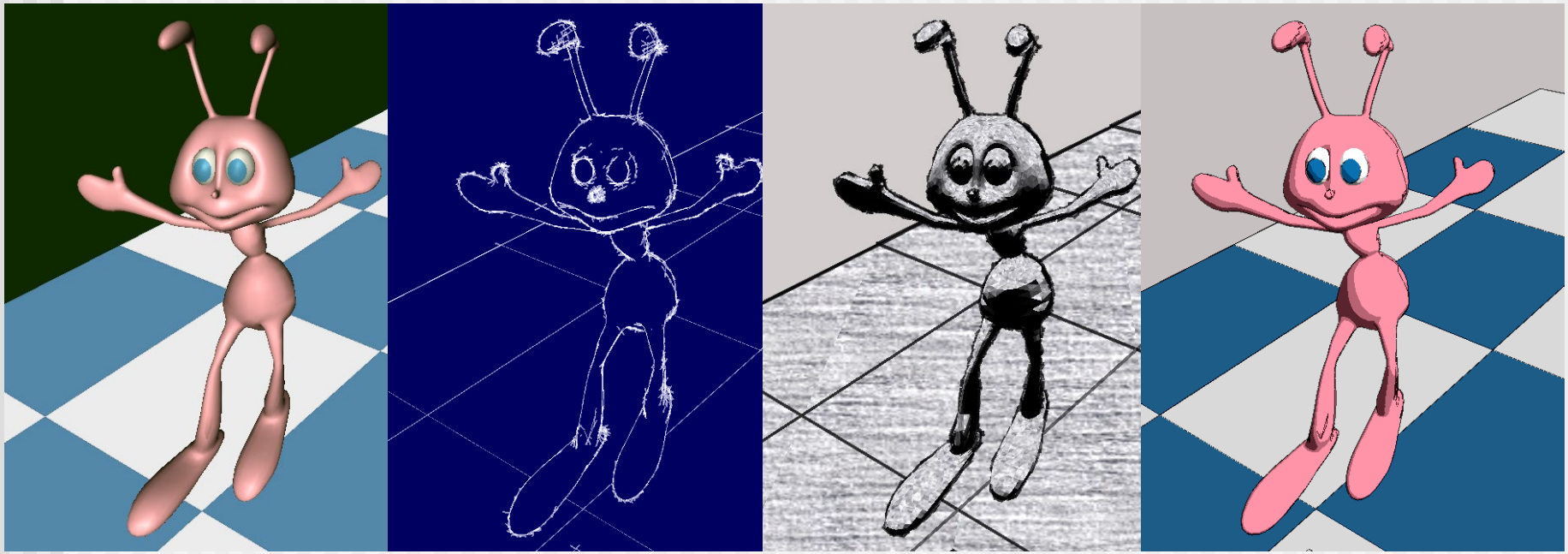
Draw it your own way…

Quake 3 Arena

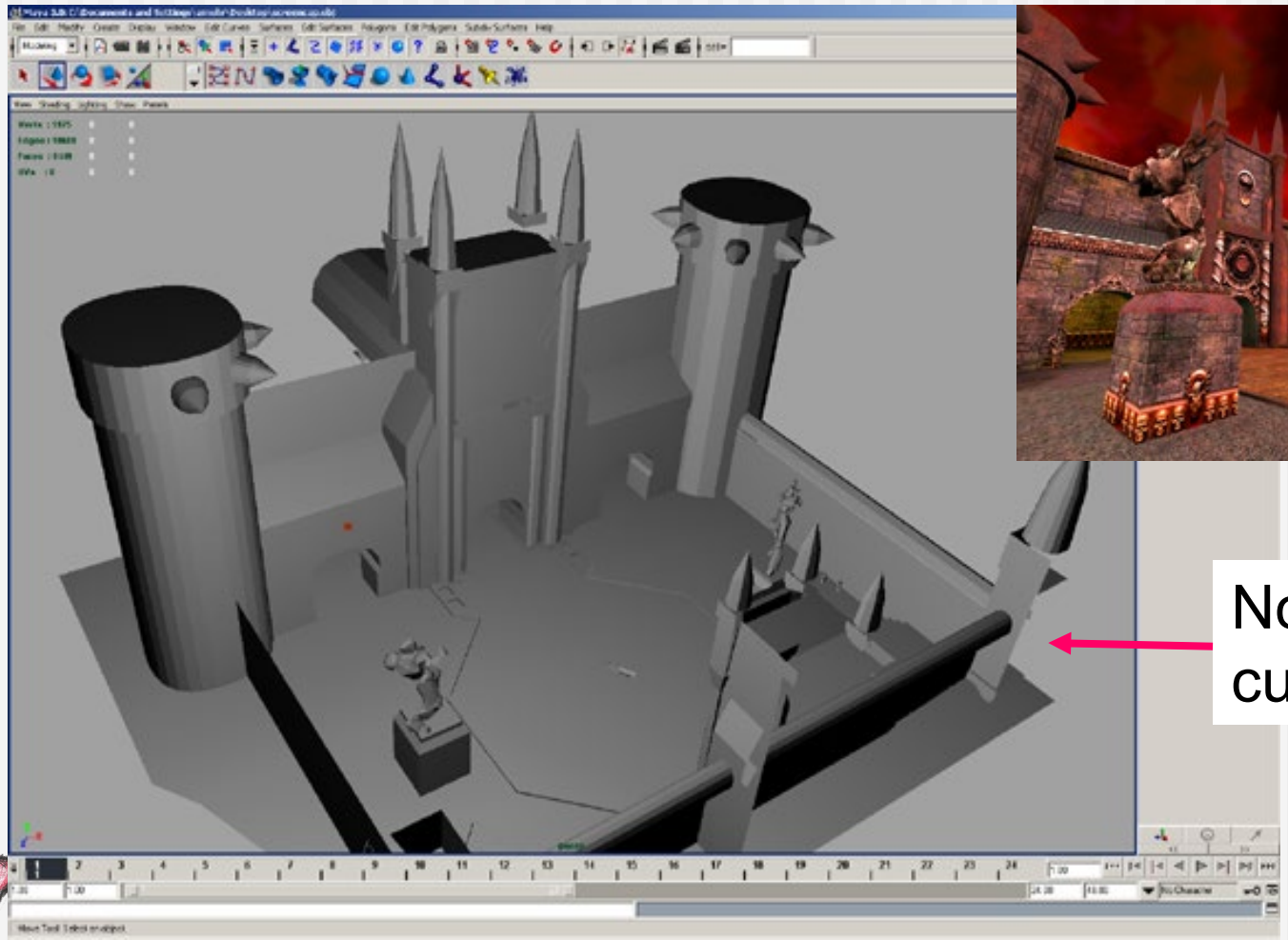
Before

After
(still approx 30fps!)

Plug-in Renderers are easy



Or write it to disk…
Print Screen to Maya!



Note view culling issues

The Limitations

- Engineering
  - Things that just take time
  - (e.g. we know how to do)
- Conceptual
  - Things that take thought
  - (e.g. we believe we could figure out)
- Fundamental
  - Things that just aren’t going to happen

What we could do…

Just more engineering

- Texture sampling
- Handle more of GL
  - Display lists, more vertex arrays
- Complete state tracking
- Geometry replacement
- More renderers and applications

Requires thought

- Identify multi-pass
- Higher-level geometry
- Object boundaries
- Object recognition
- Curve reconstruction
- Compression and transmission

What we can't do

- Really know programmer's intent
- Work 100% of the time
  - Programmers do weird things
- Only get what is drawn

- Answer?
  - "minor invasiveness"
  - Application gives some hints

Summary

- Intercept calls to the graphics library
- Build a geometric representation
- Do something cool with it

- Lots of cool things can be done!
- The are drawbacks

Yes! You can download it!

http://www.cs.wisc.edu/graphics

go to the gallery page…

Thanks!

- UW Graphics group
- National Science Foundation
  - Mike: CCR-9984506, IIS-0097456
- Industrial and University sponsors
  - Microsoft, Intel, Autodesk, Alias/Wavefront, Wisconsin Alumni Research Foundation, IBM, Pixar

Yes! You can download it!

http://www.cs.wisc.edu/graphics

go to the gallery page…