Welcome to COMP 770 (236)

- Instructor: Brandon Lloyd
  - Email: blloyd@cs.unc.edu
  - Office: SN349
  - Office hours: MW 1:00–2:00 (or by appt.)

- Webpage: http://www.cs.unc.edu/~blloyd/comp770

Prerequisites

- COMP 665(235)
  - Displays and 2D Graphics
    - Pixels, lines, images
  - Foundations of 3D Graphics
    - Points, vectors, matrices, transformations
  - Foundations of Image Analysis
    - Sampling, reconstruction, aliasing
  - Foundations of Vision
    - Eye, color, perception

- Know a programming language in the C family (C / C++ / C# / Java / Python)
- Be familiar with event driven application programming
Overview

Computer vision inverts the process

Topics

- Mathematical tools
- 3D models and interaction
- Hidden surface removal
- Rasterization
- Lighting and shading
- Shadows
- Texture mapping
- Ray tracing
- Global illumination
- Curves and surfaces
- Simplification and levels of detail
- Graphics hardware
- Image-based rendering

Mathematical tools

- homogeneous coordinates
- vectors
- planes
- frames
- transformations

$\begin{bmatrix} x' \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

3D models and interaction

- Loading and view models
- Picking and selection
- Modeling a trackball
- VR is all about interaction

COMP 872
### Hidden surface removal
- Classic problem
- BSP trees
- Ray casting
- Depth buffering

### Rasterization
- Clipping
- Scan conversion

### Lighting and shading
- Flat, gouraud, and phong shading
- Empirical and physically-based illumination models
- BRDFs

### Shadows
- Shadow volumes
- Shadow maps
- Soft shadows
Texture mapping

- Surface parameterization
- Mipmaps and filtering
- Reflection and environment mapping

Ray tracing

- Object intersection
- Reflection and refraction
- Depth-of-field, motion blur, glossy reflections, soft shadows

Global illumination

- Rendering equation
- Path tracing, photon mapping, radiosity
- COMP 870 Advanced rendering

Curves and surfaces

- Bezier curves and B-splines
- NURBS and subdivision surfaces
- Parametric solids
- COMP 767 Geometric and solid modeling
Simplification and LOD

- Levels of detail
- Progressive meshes

82 million triangles

Graphics hardware

- History
- Architecture
- Shading languages
- Future

Image based rendering

- Lightfields and lumigraphs
- Reconstruction

Animation

- Keyframing
- Parameteric splines
- Motion capture
- Simulation
- COMP 768
Textbook

- No official textbook. Here are some possible references:

Assignments

- Viewing and manipulating 3D models with OpenGL
- Rasterization and clipping
- Texture mapping and lighting
- Raytracing
- Final Project
Grading

- Assignments: 40%
- Final project: 30%
- 2 Tests: 30%
- Late policy: Assignments will lose 5% each day late (M-F)

Honor Code

- Graduate students are here for the learning not the grade.
- Collaboration encouraged, but assignments must be your own work.
- Cite any other’s work if you use their code.

About your instructor

- From Draper, Utah
- B.S. CS Brigham Young University, 2002
- 5th year Ph.D. student at UNC

My work - Shadows
About you

- Name
- Where are you from (school, state)?
- What are you studying?
- Previous graphics experience

Your new world...

- A 2D square ranging from (-1, -1) to (1, 1)
- You can draw in the box with just a few lines of code

Code example

OpenGL Code:
```c
oglColor3d(0.0, 0.8, 1.0)
glBegin(GL_POLYGON)
glVertex2d(-0.5, -0.5)
glVertex2d( 0.5, -0.5)
glVertex2d( 0.5,  0.5)
glVertex2d(-0.5,  0.5)
glEnd()
glFlush()
```

Another code example

OpenGL Code:
```c
oglColor3d(0.0, 0.8, 1.0)
glBegin(GL_POLYGON)
glVertex2d(-0.5, -0.5)
glVertex2d( 0.5, -0.5)
glVertex2d( 0.5,  0.5)
glVertex2d(-0.5,  0.5)
glEnd()
glFlush()
```
Yet another code example

OpenGL Code:

```python
# Set color
glColor3d(0.8, 0.6, 0.8)

# Begin line loop
glBegin(GL_LINE_LOOP)

for ang in range(0, 360, 2):
    x = cos(ang*pi/180)
    y = sin(ang*pi/180)
    glVertex2d(x, y)

# End line loop
glEnd()

# Flush updates
glFlush()
```

Cool toys

Thinking mathematically

- An outer circle with radius ‘rOuter’
- An inner circle with radius ‘rInner’
- A hole for the pen at ‘rOffset’
- The inner and outer circle rotate through the same arc lengths

Programming graphically
And some code

OpenGL Code:

```python
rOuter = 1.0
rInner = 0.63
rOffset = 0.59
revs = 100;
theta = 0.0
glColor3d(1.0, 0.0, 0.0)
for i in range(int(360*revs)):
    glPushMatrix()
    glRotated(theta, 0, 0, 1.0)
    glTranslated(rOuter-rInner, 0, 0)
    thetap = -theta*rOuter/rInner
    glRotated(thetap,0,0,1)
    glBegin(GL_POINTS)
    glVertex2d(rOffset, 0)
    glEnd()
    glPopMatrix()
    theta += 1.0
```

Next time...

- GLUT – Let’s see some pixels!
- Screen space – the final frontier