CS 4731: Computer Graphics
Midterm review

Emmanuel Agu

Announcement

- Extra help session tomorrow (Wednesday), FL 311, 5-6pm
- TA's will be run the session. Answer any final questions you have

Exam Overview

- Thursday, Sept. 25, in-class
- Will cover up to lecture 12 (lecture 12/24)
- Can bring:
  - One page cheat -sheet
  - Calculator
- Will test:
  - Theoretical concepts
  - Mathematics
  - Algorithms
  - Programming
  - OpenGL knowledge (program structure and some commands)

What really am I Testing?

- Understanding of on concepts (NOT only programming)
- That you can plug in numbers by hand to check your programs
- Understanding of programming (pseudocode/syntax)
- That you did the projects
- That you understand what you did in projects

General Advise

- **Read your projects** and refresh memory of what you did
- **Read the slides**: worst case – if you understand slides, you're more than 50% prepared
- Focus on **Mathematical results, concepts, algorithms**
- Plug numbers: calculate by hand
- Should be able to **predict subtle changes** to algorithm.. What ifs?..
- **Past exams**: my exams will be most similar to last year's exam
- Every lecture has references. Look at refs to focus reading

Grading Policy

- I usually do **ALL** grading myself
- Gives me a measure of where class really is, taylor rest of class
- Give you all the points, take away only what I have to
- In time constraints, laying out outline of solution gets you healthy chunk of points
- Try to write something for each question

Introduction

- Motivation for CG
- Uses of CG (simulation, image processing, movies, viz, etc)
- Elements of CG (polylines, raster images, filled regions, etc)
- Device dependent graphics libraries (OpenGL, DirectX, etc)

OpenGL / GLUT

- High-level:
  - What is OpenGL?
  - What is GLUT?
  - Functionality, how do they work together?
- Design features: low-level API, event-driven, portability, etc
- Sequential Vs. Event-driven programming
- OpenGL/GLUT program structure (create window, init, callback registration, etc)
- GLUT callback functions (registration and response to events)

OpenGL Drawing

- glBegin( ), glEnd( ), glVertex( )
- OpenGL :
    - Drawing primitives: GL_POINTS, GL_LINES, etc (should be conversant with the behaviors of major primitives)
    - Command format
    - Data types
    - Interaction: keyboard, mouse (GLUT_LEFT_BUTTON, etc)
    - OpenGL state
- No miniGL-specific questions (homegrown)

2D Graphics: Coordinate Systems

- Screen coordinate system/ Viewport
- World coordinate system/World window
- Window to Viewport mapping:
    - Motivation: why is it necessary?
    - OpenGL way: gluOrtho2D(**`left, right, bottom, top`**) glViewport **`(left, bottom, right-left, top-bottom)`**
    - Our way: calculate mapping
    - Applications: tiling, zooming, flipping, maintaining aspect ratio
- Cohen-sutherland clipping
    - algorithm operation
    - Why and how to do trivial accept/reject, chop
    - Given vertices, clip!!

Fractals

- What are fractals?
    - Self similarity
    - Applications (clouds, grass, terrain etc)
- Koch curves/snowflakes
    - How to build K1, K2, etc… S1, S2, etc.
    - Pseudocode: how to draw
- Mandelbrot set
    - Complex numbers: s, c, orbits, complex number math
    - Dwell function
    - Assigning colors
    - Mapping mandelbrot to screen

Points, Scalars Vectors

- Vector Operations:
    - Addition, subtraction, scaling
    - Magnitude
    - Normalization
    - Dot product
    - Cross product
    - Finding angle between two vectors
- Standard unit vector
- Normal of a plane

Transforms

- Homogeneous coordinates Vs. Ordinary coordinates
- 2D/3D affine transforms: rotation, scaling, translation, shearing
- Should be able to take problem description and build transforms and apply to vertices
- 2D: rotation (scaling, etc) about arbitrary center:
  - $T(Px,Py) R(\theta) T(-Px,-Py) * P$
- Composing transforms
- OpenGL transform commands (glRotate, glTranslate, etc)
- 3D rotation:
  - x-roll, y-roll, z-roll, about arbitrary vector (Euler theorem) if given azimuth, latitude of vector or (x, y, z) of normalized vector
- Matrix multiplication!!

Modeling

- GLUT models (teapot, sphere, cube, etc)
- Overview of openGL
  - Modelview matrix (M and V part)
  - Projection matrix
  - Clipping
  - Viewport
- Should know high-level what each stage does
- OpenGL matrices: what are they? How to select, initialize, compose
- Synthetic camera basics
- Hierachical modeling using OpenGL (glPopMatrix, glPushMatrix)
- SDL

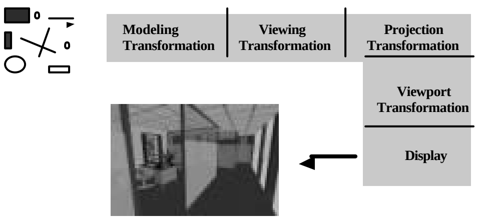
Modeling using Polygonal Meshes

- Mesh representations
  - Data structures (Vertex list, Normal list, face list, indexing)
- Finding normal:
  - Cross product method
  - Newell method
  - Should be able to plug number and get answer
- Pseudocode for manipulating, drawing mesh

3D Viewing

- gluLookat(Eye, COI, Up) to set camera
  - Pitch: nose up-down
  - Roll: roll body of plane
  - Yaw: move nose side to side
- Projection:
  - View volume, near plane, far plane
  - gluPerspective(fovy, aspect, near, far) **or**
  - glFrustum (left, right, bottom, top, near, far)
  - glOrtho(left, right, bottom, top, near, far)

3D viewing



Modeling Transformation

Viewing Transformation

Projection Transformation

Viewport Transformation

Display

3D viewing: Eye Coordinate Frame

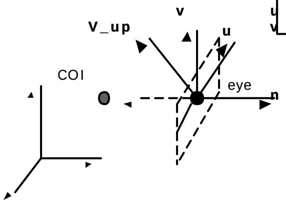
- Given

gluLookat(Eye, COI, up_vector)
How do you build V part of
Modelview matrix?

Eye space **origin:**
(Eye.x , Eye.y, Eye.z)

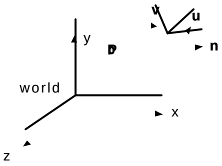
Basis vectors:

n = (eye – COI) / | eye – COI|
u = (V_up x **n**) / | V_up x **n** |
v = **n** x **u**



3D Viewing: World to Eye Transformation

- Transformation matrix (M_{w2e}) ?
  $P' = M_{w2e} \times P$



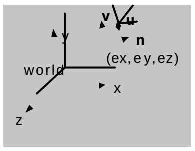
1. Come up with the transformation sequence to move eye coordinate frame to the world

2. And then apply this sequence to the point P in a reverse order

3D Viewing: World to Eye Transformation

- Transformation order: apply the transformation to the object in a reverse order - translation first, and then rotate

$$M_{w2e} = \begin{vmatrix} ux & uy & ux & 0 \\ vx & vy & vz & 0 \\ nx & ny & nz & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & -ex \\ 0 & 1 & 0 & -ey \\ 0 & 0 & 1 & -ez \\ 0 & 0 & 0 & 1 \end{vmatrix}$$



v
u
y
n
(ex,ey,ez)
world
x
z