# CS418
# OpenGL & GLUT
# Programming Tutorial
# (I)

Presented by : Wei-Wen Feng
1/30/2008

# I Am Your TA

- Name : Wei-Wen Feng
- 4th Year Graduate Student in Graphics
- I will be
  - Holding discussion/tutorial session
  - Holding Lab/Office hours
  - Answering technical/administrative questions on newsgroup ( class.cs418 )
  - Most importantly : help you learn these materials
  - Graphics is sometimes tough, but it's fun !

# How This Session Works

- Format :
  - 20~30 min Tutorial
  - Q&A
- Cover materials such as :
  - OpenGL programming
  - Review HWs/MPs/Exams
  - Things you find interesting ( Game programming, GPGPU, etc ? )

# Today's Agenda

- Setup Your Program
- A GLUT Skeleton Code
- Load Your Mesh
- Display Your Mesh
- Color Your Mesh
- Q & A

# Setup Program

- Libraries for OpenGL programming
  - GL : core Graphics API
  - GLU : Utility library
  - GLUT : Help you setup a window application

  - GL/GLU should come with VS or Your OS.
  - GLUT can be found at : Nate Robin's page . (http://www.xmission.com/~nate/glut.html)

# Setup Program

- https://agora.cs.uiuc.edu/display/cs418/OpenGL+Setup+for+Visual+Studio
- Visual Studio is available for free to CS students through the MSDN Academic Alliance.
- Other platform should have OpenGL by default.

- We prefer you hand-in a VS project, but other platform ( Linux, MacOS, etc) with source code is ok.

# GLUT Skeleton Code

- Support vector/matrix class
  - You can implement everything in float only, but it might turn messy for later MPs.
  - Some generic operation will make debugging much easier.
    - vector dot-product, matrix multiplication, etc
  - Make use of the provided class or write your own class/functions. It pays to plan ahead.

```
#include "gfx/vec3.h"
#include "gfx/mat3.h“
```

# GLUT Skeleton Code

- Main ()
  - Set up window properties and fill in event call-back functions.

```
glutInit(&argc, (char**)argv);
glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB |
GLUT_DEPTH);
glutInitWindowSize (500, 500);
glutInitWindowPosition (100, 100);
glutCreateWindow ((const char*)argv[0]);
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutKeyboardFunc(keyboard);
glutMainLoop();
```

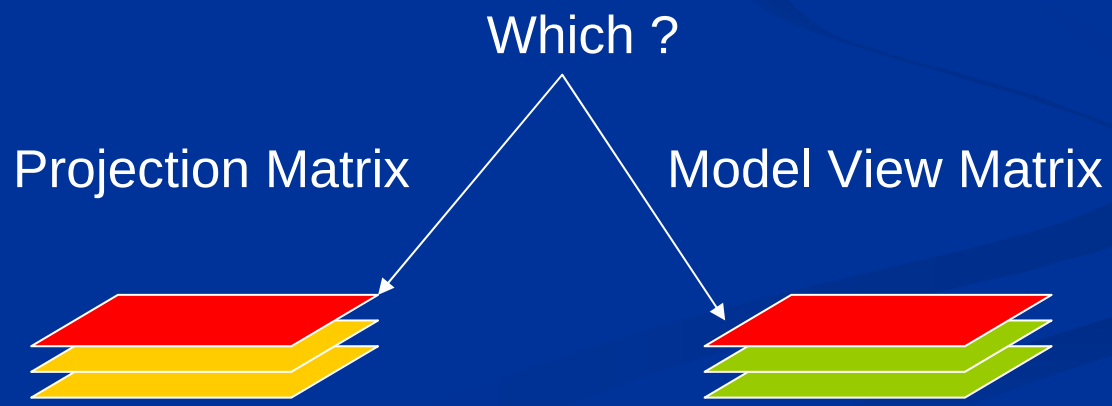
# GLUT Skeleton Code

- Reshape function
  - Called whenever your window is resized
  - Usually, adjust viewport size & projection matrix here.

```
glViewport(0, 0, w, h);
float fAspect = ((float)w)/h;
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(70.f,fAspect,0.001f,30.f);
```

# GLUT Skeleton

- Matrix in OpenGL
  - OpenGL maintains two stacks of matrices in system.
  - Always specify which stack you are working on.
  - **glMatrixMode (Mode)**



# GLUT Skeleton Code

- Display function
  - Core part of your code → How should you draw
  - Called when your window needs redraw.
    - Or call "glutPosRedisplay()" to force a redraw
  - Usually, clear the buffer before drawing anything
    - glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
  - Also a good place for setup transformation.
    - Not the main point of this MP.

# GLUT Skeleton Code

- Some useful functions
  - **gluLookAt, gluPerspective** → Save you a lot of painful work to figure out view/projection transformation.
  - **glutSolidSphere** → When in doubt, draw a sphere !
  - **glFlush, glutSwapBuffer** → Check if your triangles are really into graphics hardware & if you are looking at right frame.
  - **glLoadIdentity** → Set your current matrix into identity.

# Load Your Mesh

- We provide .obj 3D models with colors.
- You should have skills to write a simple parser loading the files.
- Mesh structure
  - Organize the vertex & face data
  - Index Array List
  - Half-Edge

# Mesh Structure

- You will have a list of vertex attributes : Positions, Colors, etc.
- Another indexed list for triangles
  - Ex : Triangle 1 is formed by vertex 1,3,4
- The way I usually do :
  - Store vertex attributes in one array
  - Store Triangle Indices in another.
  - When needed, iterate through each triangle, and grab vertex attributes based on Indices.
  - More complicated structure is possible → Half-Edge, etc.

# Display Your Mesh

- Assuming you've set up the view/projection transformation.
- Display one triangles

```
glBegin(GL_TRIANGLES);
glVertex3f(x1,y1,z1);
glVertex3f(x2,y2,z2);
glVertex3f(x3,y3,z3);
glEnd();
```

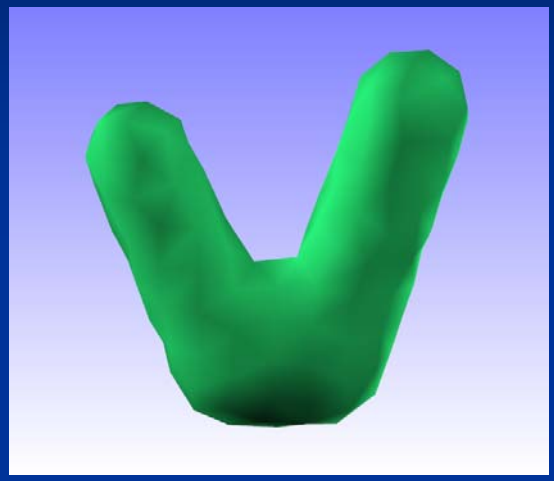
- glBegin→Decide which primitive you will display.
  - GL_POINTS, GL_LINES, GL_TRIANGLES, etc.
- Display a mesh is similar, just go through each triangle in the mesh.

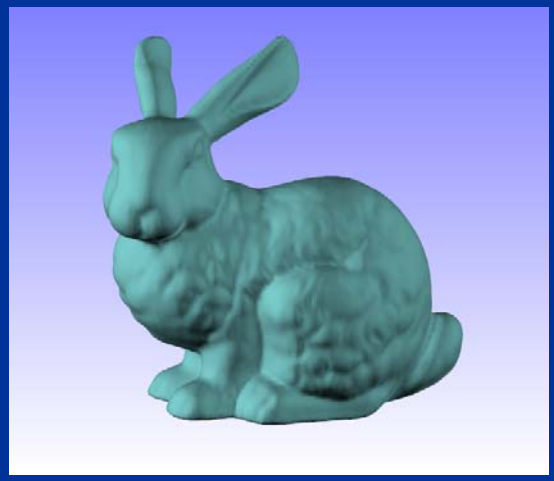
# Color Your Mesh

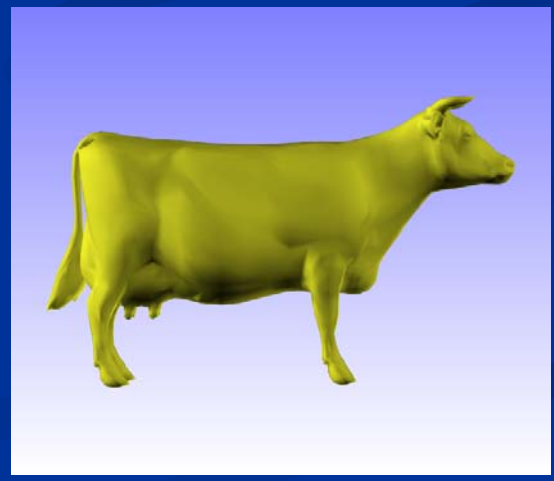
- glColor3f→Set R,G,B color
  - Range from 0.0~1.0. (1.0,1.0,1.0) is white.
- Use the provided colors, or generate your own.
- Color one triangle with Red, Green, Blue at each vertex

```
glBegin(GL_TRIANGLES);
glColor3f(1.0,0.0,0.0); //red
glVertex3f(x1,y1,z1);
glColor3f(0.0,1.0,0.0); // green
glVertex3f(x2,y2,z2);
glColor3f(0.0,0.0,1.0); // blue
glVertex3f(x3,y3,z3);
glEnd();
```

# Reference Results







# Q&A

- Tips :
  - I See Nothing ? → Check your camera transformations, check your primitives, draw a testing triangle.
  - Make sure you load correct things into your mesh structure.
  - Please…..Don't load your mesh in Display Function.