

Graphics Software Portfolio

by

Jacob Maurer

maur4137@fredonia.edu

CSIT462

Fall 2019

Course Faculty: Dr. Junaid Ahmed Zubairi, Professor  
Computer and Information Sciences, SUNY at Fredonia,   
Fredonia NY 14063, email: [zubairi@fredonia.edu](mailto:zubairi@fredonia.edu)

Table of Contents

[HW1: Question Text 4](#_Toc387240586)

[Score: 110 out of 100 4](#_Toc387240587)

[Source Code: 4](#_Toc387240588)

[HW2: Question Text 15](#_Toc387240589)

[Score: 100 out of 100 15](#_Toc387240590)

[Source Code: 15](#_Toc387240591)

[HW3: Question Text 27](#_Toc387240592)

[Score: 100 out of 100 27](#_Toc387240593)

[Source Code: 27](#_Toc387240594)

[HW4: Question Text 41](#_Toc387240595)

[Score: 100 out of 100 41](#_Toc387240596)

[Source Code: 41](#_Toc387240597)

[HW5: Question Text 62](#_Toc387240598)

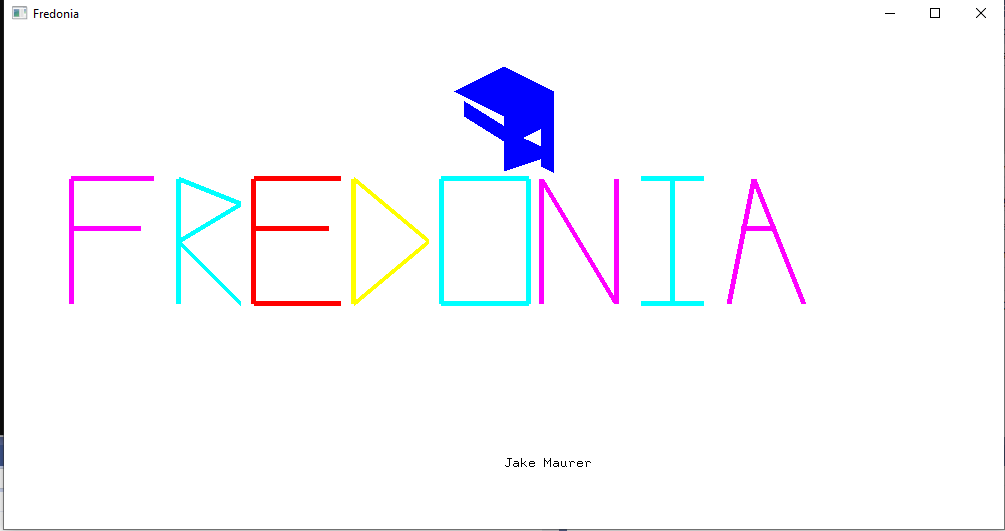
[Score: 100 out of 100 62](#_Toc387240599)

[Source Code: 62](#_Toc387240600)

**OVERALL SCORE IN ASSIGNMENTS: 100 out of 100**

# HW1:

Download and install FREEGLUT and run the program on page 46 of the textbook (program contains functions init( ), lineSegment( ) besides main ( ). Modify the program so that it draws several line segments with different colors. The line segments should join together to form the letters “FREDONIA”. Try to match the style with the logo as shown below as much as you can, realizing the limitation of using only straight lines but you are free to use other drawing primitives. Next, you should use the following code segment and modify it to show your full name on the screen. For help on displaying text on screen, refer to section 10 on font rendering in GLUT specs document inside the course information folder.



## Score: 100 out of 100

## Source Code:

#include <stdio.h>

#include <gl\freeglut.h>

void init(void)

{

glClearColor(0.0, 0.0, 0.0, 1.0);

glClearDepth(1.0);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

glEnable(GL\_DEPTH\_TEST);

}

void display()

{

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT);

/\*glLineWidth(10.0);

glBegin(GL\_LINES);

glColor3f(255, 0, 0); glVertex2f(-0.5f, -0.3f);

glColor3f(255, 0, 0); glVertex2f(0.5f, 0.3f);

glColor3f(255, 0, 0); glVertex3f(-0.865, 0, 0);

glColor3f(255, 0, 0); glVertex3f(-0.865, .5, 0);

glColor3f(255, 0, 0); glVertex3f(-0.9, .5, 0);

glColor3f(255, 0, 0); glVertex3f(-0.9, 0, 0);

glEnd();

\*/

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(-.1, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .65, 0);

glColor3f(0, 0, 255); glVertex3f(.1, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .85, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(.025, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .6, 0);

glColor3f(0, 0, 255); glVertex3f(.025, .55, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(.1, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .45, 0);

glColor3f(0, 0, 255); glVertex3f(.1, .425, 0);

glEnd();

/\*

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(.1, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.1, .39, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .42, 0);

glEnd();

\*/

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(0, .6, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .53, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .48, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .43, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(0, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.025, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.025, .5, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .5, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(-.08, .712, 0);

glColor3f(0, 0, 255); glVertex3f(-.08, .65, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .55, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .612, 0);

glEnd();

//F

glLineWidth(5);

glColor3f(255.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.865, -.10, 0.0);

glVertex3f(-.865, .4, 0);

glEnd();

glColor3f(255.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.865, .4, 0.0);

glVertex3f(-.7, .4, 0);

glEnd();

glColor3f(255.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.865, .2, 0.0);

glVertex3f(-.725, .2, 0);

glEnd();

//R

glColor3f(0, 255.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.65, .4, 0.0);

glVertex3f(-.65, -.1, 0);

glEnd();

glColor3f(0, 255.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.65, .4, 0.0);

glVertex3f(-.525, .3, 0);

glEnd();

glColor3f(0, 255.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.65, .15, 0.0);

glVertex3f(-.525, .3, 0);

glEnd();

glColor3f(0, 255.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.65, .15, 0.0);

glVertex3f(-.525, -.1, 0);

glEnd();

//E

glColor3f(255.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.5, .4, 0.0);

glVertex3f(-.5, -.1, 0);

glEnd();

glColor3f(255.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.5, .4, 0.0);

glVertex3f(-.325, .4, 0);

glEnd();

glColor3f(255.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.5, .2, 0.0);

glVertex3f(-.35, .2, 0);

glEnd();

glColor3f(255.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.5, -.1, 0.0);

glVertex3f(-.325, -.1, 0);

glEnd();

//D

glColor3f(255.0, 50.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.3, .4, 0.0);

glVertex3f(-.3, -.1, 0);

glEnd();

glColor3f(255.0, 50.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.3, .4, 0.0);

glVertex3f(-.15, .15, 0);

glEnd();

glColor3f(255.0, 50.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.3, -.1, 0.0);

glVertex3f(-.15, .15, 0);

glEnd();

//O

glColor3f(0.0, 255.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(-.125, .4, 0.0);

glVertex3f(-.125, -.1, 0);

glEnd();

glColor3f(0.0, 255.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.05, .4, 0.0);

glVertex3f(.05, -.1, 0);

glEnd();

glColor3f(0.0, 255.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.05, .4, 0.0);

glVertex3f(-.125, .4, 0);

glEnd();

glColor3f(0.0, 255.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.05, -.1, 0.0);

glVertex3f(-.125, -.1, 0);

glEnd();

//N

glColor3f(200.0, 0.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.075, .4, 0.0);

glVertex3f(.075, -.1, 0);

glEnd();

glColor3f(200.0, 0.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.225, .4, 0.0);

glVertex3f(.225, -.1, 0);

glEnd();

glColor3f(200.0, 0.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.075, .4, 0.0);

glVertex3f(.225, -.1, 0);

glEnd();

//I

glColor3f(0.0, 180.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.275, .4, 0.0);

glVertex3f(.4, .4, 0);

glEnd();

glColor3f(0.0, 180.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.275, -.1, 0.0);

glVertex3f(.4, -.1, 0);

glEnd();

glColor3f(0.0, 180.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.337, .4, 0.0);

glVertex3f(.337, -.1, 0);

glEnd();

//A

glColor3f(200.0, 0.0, 90.0);

glBegin(GL\_LINES);

glVertex3f(.45, -.1, 0.0);

glVertex3f(.5, .4, 0);

glEnd();

glColor3f(200.0, 0.0, 90.0);

glBegin(GL\_LINES);

glVertex3f(.6, -.1, 0.0);

glVertex3f(.5, .4, 0);

glEnd();

glColor3f(200.0, 0.0, 90.0);

glBegin(GL\_LINES);

glVertex3f(.475, .2, 0.0);

glVertex3f(.545, .2, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(0, -.75);

char mesg[12] = "Jake Maurer";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg[i]);

//Top Dongle

glFlush();

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(100, 100);

glutInitWindowSize(1000, 500);

glutCreateWindow("Fredonia");

glutDisplayFunc(display);

glutMainLoop();

}

# HW2:

Check the "House Template" that shows how to choose coordinates for drawing various shapes. Download the template from course website. Build your own template and develop a program using OpenGL to draw one of the following shapes

*(For students with even ID#) A pantry with 5 shelves and each shelf shows food plates (magically standing straight up, different sizes and filled with different colors)*

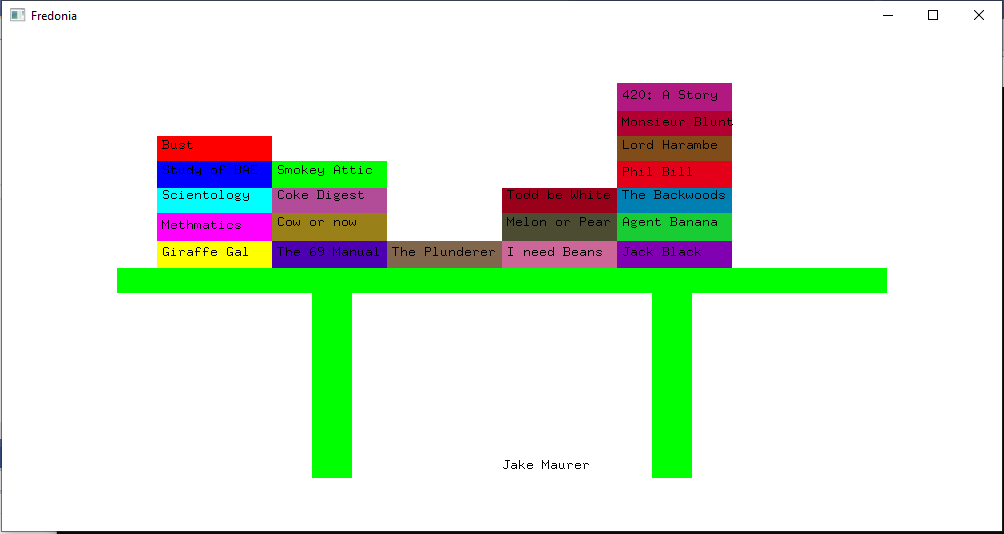
*(For students with odd ID#) Front view of a table with five stacks of books each with different fill colors*

In each diagram, the number of books or plates (in each shelf or stack) equals last 5 digits of your ID Number. A digit must be incremented by 1 if it is a zero.

Books must show short titles such as “intro to python”, “data processing with C”, “Java primer”, “Javascript for beginners” etc.. Plates show text such as “dinner”, “salad”, “dessert” etc.

Submit only the .cpp files (Source code files). Do not submit project or executable files. I would build and run your program on my machine.

GRADING GUIDE: Program is able to run and open a graphic window 40, program shows circular plates and rectangular books on rectangular pantry shelves or table 70, program is also able to show text on plates and book sides 85, program can fill colors in all the shapes drawn in addition to the steps listed earlier. 100 out of 100.



## Score: 100 out of 100

## Source Code:

#include <stdio.h>

#include <gl\freeglut.h>

void init(void)

{

glClearColor(0.0, 0.0, 0.0, 1.0);

glClearDepth(1.0);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

glEnable(GL\_DEPTH\_TEST);

}

void display()

{

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT);

/\*glLineWidth(10.0);

glBegin(GL\_LINES);

glColor3f(255, 0, 0); glVertex2f(-0.5f, -0.3f);

glColor3f(255, 0, 0); glVertex2f(0.5f, 0.3f);

glColor3f(255, 0, 0); glVertex3f(-0.865, 0, 0);

glColor3f(255, 0, 0); glVertex3f(-0.865, .5, 0);

glColor3f(255, 0, 0); glVertex3f(-0.9, .5, 0);

glColor3f(255, 0, 0); glVertex3f(-0.9, 0, 0);

glEnd();

\*/

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(-.1, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .65, 0);

glColor3f(0, 0, 255); glVertex3f(.1, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .85, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(.025, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .6, 0);

glColor3f(0, 0, 255); glVertex3f(.025, .55, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(.1, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .45, 0);

glColor3f(0, 0, 255); glVertex3f(.1, .425, 0);

glEnd();

/\*

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(.1, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.1, .39, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .42, 0);

glEnd();

\*/

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(0, .6, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .53, 0);

glColor3f(0, 0, 255); glVertex3f(.075, .48, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .43, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(0, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.025, .75, 0);

glColor3f(0, 0, 255); glVertex3f(.025, .5, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .5, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 0, 255); glVertex3f(-.08, .712, 0);

glColor3f(0, 0, 255); glVertex3f(-.08, .65, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .55, 0);

glColor3f(0, 0, 255); glVertex3f(.0, .612, 0);

glEnd();

//F

glLineWidth(5);

glColor3f(255.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.865, -.10, 0.0);

glVertex3f(-.865, .4, 0);

glEnd();

glColor3f(255.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.865, .4, 0.0);

glVertex3f(-.7, .4, 0);

glEnd();

glColor3f(255.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.865, .2, 0.0);

glVertex3f(-.725, .2, 0);

glEnd();

//R

glColor3f(0, 255.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.65, .4, 0.0);

glVertex3f(-.65, -.1, 0);

glEnd();

glColor3f(0, 255.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.65, .4, 0.0);

glVertex3f(-.525, .3, 0);

glEnd();

glColor3f(0, 255.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.65, .15, 0.0);

glVertex3f(-.525, .3, 0);

glEnd();

glColor3f(0, 255.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.65, .15, 0.0);

glVertex3f(-.525, -.1, 0);

glEnd();

//E

glColor3f(255.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.5, .4, 0.0);

glVertex3f(-.5, -.1, 0);

glEnd();

glColor3f(255.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.5, .4, 0.0);

glVertex3f(-.325, .4, 0);

glEnd();

glColor3f(255.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.5, .2, 0.0);

glVertex3f(-.35, .2, 0);

glEnd();

glColor3f(255.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.5, -.1, 0.0);

glVertex3f(-.325, -.1, 0);

glEnd();

//D

glColor3f(255.0, 50.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.3, .4, 0.0);

glVertex3f(-.3, -.1, 0);

glEnd();

glColor3f(255.0, 50.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.3, .4, 0.0);

glVertex3f(-.15, .15, 0);

glEnd();

glColor3f(255.0, 50.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.3, -.1, 0.0);

glVertex3f(-.15, .15, 0);

glEnd();

//O

glColor3f(0.0, 255.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(-.125, .4, 0.0);

glVertex3f(-.125, -.1, 0);

glEnd();

glColor3f(0.0, 255.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.05, .4, 0.0);

glVertex3f(.05, -.1, 0);

glEnd();

glColor3f(0.0, 255.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.05, .4, 0.0);

glVertex3f(-.125, .4, 0);

glEnd();

glColor3f(0.0, 255.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.05, -.1, 0.0);

glVertex3f(-.125, -.1, 0);

glEnd();

//N

glColor3f(200.0, 0.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.075, .4, 0.0);

glVertex3f(.075, -.1, 0);

glEnd();

glColor3f(200.0, 0.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.225, .4, 0.0);

glVertex3f(.225, -.1, 0);

glEnd();

glColor3f(200.0, 0.0, 150.0);

glBegin(GL\_LINES);

glVertex3f(.075, .4, 0.0);

glVertex3f(.225, -.1, 0);

glEnd();

//I

glColor3f(0.0, 180.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.275, .4, 0.0);

glVertex3f(.4, .4, 0);

glEnd();

glColor3f(0.0, 180.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.275, -.1, 0.0);

glVertex3f(.4, -.1, 0);

glEnd();

glColor3f(0.0, 180.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.337, .4, 0.0);

glVertex3f(.337, -.1, 0);

glEnd();

//A

glColor3f(200.0, 0.0, 90.0);

glBegin(GL\_LINES);

glVertex3f(.45, -.1, 0.0);

glVertex3f(.5, .4, 0);

glEnd();

glColor3f(200.0, 0.0, 90.0);

glBegin(GL\_LINES);

glVertex3f(.6, -.1, 0.0);

glVertex3f(.5, .4, 0);

glEnd();

glColor3f(200.0, 0.0, 90.0);

glBegin(GL\_LINES);

glVertex3f(.475, .2, 0.0);

glVertex3f(.545, .2, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(0, -.75);

char mesg[12] = "Jake Maurer";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg[i]);

//Top Dongle

glFlush();

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(100, 100);

glutInitWindowSize(1000, 500);

glutCreateWindow("Fredonia");

glutDisplayFunc(display);

glutMainLoop();

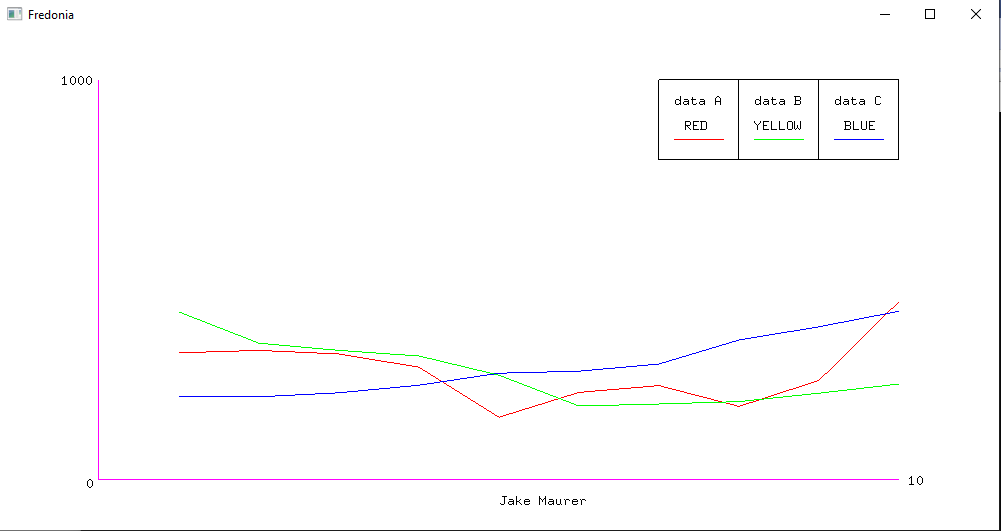
}

# HW3:

If your Fredonia ID ends in an odd digit, plot the line graphs of data sets A, B and C on the same chart.  **Final program must show each data set graph with different color lines. Pressing a key such as ‘s’ should zoom up the graphs.** In bottom right corner, a legend should be shown just as in Microsoft Excel.

If your Fredonia ID ends in an even digit, plot the bar graphs of data sets A, B and C. **The bars must be filled with different colors for each set**. **Pressing a key such as ‘s’ should zoom up the graphs.** In bottom right corner, a legend should be shown just as in Microsoft Excel.

## Score: 100 out of 100



## Source Code:

#include <stdio.h>

#include <GL\freeglut.h>

void init(void)

{

glClearColor(0.0, 0.0, 0.0, 1.0);

glClearDepth(1.0);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

glEnable(GL\_DEPTH\_TEST);

}

void display()

{

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT);

/\*glLineWidth(10.0);

glBegin(GL\_LINES);

glColor3f(255, 0, 0); glVertex2f(-0.5f, -0.3f);

glColor3f(255, 0, 0); glVertex2f(0.5f, 0.3f);

glColor3f(255, 0, 0); glVertex3f(-0.865, 0, 0);

glColor3f(255, 0, 0); glVertex3f(-0.865, .5, 0);

glColor3f(255, 0, 0); glVertex3f(-0.9, .5, 0);

glColor3f(255, 0, 0); glVertex3f(-0.9, 0, 0);

glEnd();

\*/

glColor3f(200.0, 0.0, 90.0);

glBegin(GL\_LINES);

glVertex3f(-.8, .8, 0.0);

glVertex3f(-.8, -.8, 0);

glEnd();

glColor3f(200.0, 0.0, 90.0);

glBegin(GL\_LINES);

glVertex3f(-.8, -.8, 0.0);

glVertex3f(.8, -.8, 0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.64, -.2912, 0.0);

glVertex3f(-.48, -.2816, 0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.48, -.2816, 0);

glVertex3f(-.32, -.296, 0.0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.32, -.296, 0.0);

glVertex3f(-.16, -.3488, 0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.16, -.3488, 0);

glVertex3f(0, -.5488, 0.0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(0, -.5488, 0.0);

glVertex3f(.16, -.4496, 0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.16, -.4496, 0.0);

glVertex3f(.32, -.4224, 0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.32, -.4224, 0.0);

glVertex3f(.48, -.5056, 0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.48, -.5056, 0.0);

glVertex3f(.64, -.4016, 0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.64, -.4016, 0.0);

glVertex3f(.8, -.0864, 0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.64, -.128, 0.0);

glVertex3f(-.48, -.2528, 0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.48, -.2528, 0);

glVertex3f(-.32, -.2816, 0.0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.32, -.2816, 0.0);

glVertex3f(-.16, -.304, 0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(-.16, -.304, 0);

glVertex3f(0, -.3808, 0.0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(0, -.3808, 0.0);

glVertex3f(.16, -.504, 0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.16, -.504, 0.0);

glVertex3f(.32, -.496, 0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.32, -.496, 0.0);

glVertex3f(.48, -.4864, 0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.48, -.4864, 0.0);

glVertex3f(.64, -.4528, 0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.64, -.4528, 0.0);

glVertex3f(.8, -.416, 0);

glEnd();

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.64, -.464, 0.0);

glVertex3f(-.48, -.4672, 0);

glEnd();

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.48, -.4672, 0);

glVertex3f(-.32, -.4512, 0.0);

glEnd();

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.32, -.4512, 0.0);

glVertex3f(-.16, -.4208, 0);

glEnd();

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(-.16, -.4208, 0);

glVertex3f(0, -.3728, 0.0);

glEnd();

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(0, -.3728, 0.0);

glVertex3f(.16, -.3648, 0);

glEnd();

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.16, -.3648, 0.0);

glVertex3f(.32, -.336, 0);

glEnd();

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.32, -.336, 0.0);

glVertex3f(.48, -.24, 0);

glEnd();

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.48, -.24, 0.0);

glVertex3f(.64, -.1872, 0);

glEnd();

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.64, -.1872, 0.0);

glVertex3f(.8, -.1248, 0);

glEnd();

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.32, .48, 0.0);

glVertex3f(.8, .48, 0);

glEnd();

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.8, .48, 0.0);

glVertex3f(.8, .8, 0);

glEnd();

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.8, .8, 0.0);

glVertex3f(.32, .8, 0);

glEnd();

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.32, .8, 0.0);

glVertex3f(.32, .48, 0);

glEnd();

glColor3f(100.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.35, .56, 0.0);

glVertex3f(.45, .56, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.35, .7);

char mesgA[12] = "data A";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgA[i]);

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.37, .6);

char mesgRed[12] = "RED";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgRed[i]);

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.48, .48, 0.0);

glVertex3f(.48, .8, 0);

glEnd();

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.64, .48, 0.0);

glVertex3f(.64, .8, 0);

glEnd();

glColor3f(0.0, 100.0, 0.0);

glBegin(GL\_LINES);

glVertex3f(.51, .56, 0.0);

glVertex3f(.61, .56, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.51, .7);

char mesgB[12] = "data B";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgB[i]);

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.51, .6);

char mesgYellow[12] = "YELLOW";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgYellow[i]);

glColor3f(0.0, 0.0, 100.0);

glBegin(GL\_LINES);

glVertex3f(.67, .56, 0.0);

glVertex3f(.77, .56, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.67, .7);

char mesgC[12] = "data C";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgC[i]);

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.69, .6);

char mesgGreen[12] = "BLUE";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgGreen[i]);

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.825, -.83);

char mesgZero[12] = "0";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgZero[i]);

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.875, .78);

char mesgMax[12] = "1000";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgMax[i]);

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.82, -.82);

char mesgMaxX[12] = "10";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgMaxX[i]);

glColor3f(0, 0.0, 0.0);

glRasterPos2d(0, -.9);

char mesg[12] = "Jake Maurer";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg[i]);

//Top Dongle

glFlush();

}

void keyboard(unsigned char key, int x, int y) {

if (key == 'z') {

glScaled(1.2, 1.2, 1.2);

glutPostRedisplay();

}

else if (key == 'x') {

glScaled(.835, .835, .835);

glutPostRedisplay();

}

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(100, 100);

glutInitWindowSize(1000, 500);

glutCreateWindow("Fredonia");

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

glutMainLoop();

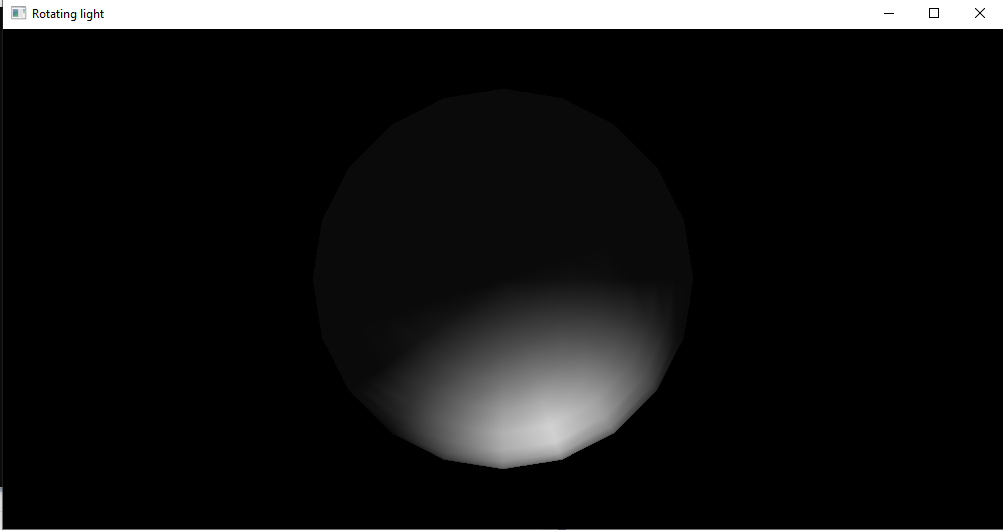
}

# HW4:

You should show the books originally in a small size. User is to be given options of pressing "z" and "t". Pressing "z" will zoom or enlarge one of the books by about 10%. Therefore successive "z" commands should enlarge one of the books. Pressing “t” would move one of the books to the right by about 10 pixels in horizontal direction. Successive “t” commands would move the book away.



Build a program that draws a Soccer ball and adds a light source to the scene. On successive strokes of “m” key, the light source moves and the reflected light from the ball changes direction. Bonus additional 10 points for using a spot light.



## Score: 100 out of 100

## Source Code:

#include <stdio.h>

#include <GL\freeglut.h>

/\*9) Now open Visual Studios and create an empty C++ project

10) Add new item, main.cpp file

11) Right click on your project name in the Solution Explorer panel on the right of your screen and select properties

12) Make sure the Platform is set to x64

13) Click on VC++ Directories

14) Click on Include Directories and then on the drop-down arrow on the right side and click edit

15) Click on the new folder icon at the top and then the “…”

16) Navigate to: “C:\Program Files\Common Files\MSVC\freeglut\include” so that the include folder is selected and click select folder.

17) Click on the new folder icon at the top and then the “…” again

18) Navigate to: “C:\Program Files\Common Files\MSVC\freeglut\lib\x64” so that the x64 folder is selected and click select folder.

19) Click OK

20) Now Click on Library Directories and then on the drop-down arrow on the right side and click edit

21) Click on the new folder icon at the top and then the “…”

22) Navigate to: “C:\Program Files\Common Files\MSVC\freeglut\lib\x64” so that the x64 folder is selected and click select folder.

23) Click OK

24) Make sure to click “Apply”!

25) Click OK

26) At the top of the screen next to Debug set it to x64

You should be all setup. You can paste the test code from the freeglut\_help doc on OnCourse to make sure it works. For every new project you create you will have to repeat steps 9 through 26.

\*/

double x1 = .23; // - 10%

double y1 = .68; // no movement

double x2 = .23; // - 10%

double y2 = .79; // + 10%

double x3 = .46; // + 10%

double y3 = .79; // + 10%

double x4 = .46; // + 10%

double y4 = .68; // no movement

double xName = .24;

double yName = .73;

void init(void)

{

glClearColor(0.0, 0.0, 0.0, 1.0);

glClearDepth(1.0);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

glEnable(GL\_DEPTH\_TEST);

}

void display()

{

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT);

/\*glLineWidth(10.0);

glBegin(GL\_LINES);

glColor3f(255, 0, 0); glVertex2f(-0.5f, -0.3f);

glColor3f(255, 0, 0); glVertex2f(0.5f, 0.3f);

glColor3f(255, 0, 0); glVertex3f(-0.865, 0, 0);

glColor3f(255, 0, 0); glVertex3f(-0.865, .5, 0);

glColor3f(255, 0, 0); glVertex3f(-0.9, .5, 0);

glColor3f(255, 0, 0); glVertex3f(-0.9, 0, 0);

glEnd();

\*/

glBegin(GL\_POLYGON);

glColor3f(0, 165, 0); glVertex3f(-.77, .05, 0);

glColor3f(0, 165, 0); glVertex3f(-.77, -.05, 0);

glColor3f(0, 165, 0); glVertex3f(.77, -.05, 0);

glColor3f(0, 165, 0); glVertex3f(.77, .05, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 165, 0); glVertex3f(-.38, -.05, 0);

glColor3f(0, 165, 0); glVertex3f(-.38, -.79, 0);

glColor3f(0, 165, 0); glVertex3f(-.3, -.79, 0);

glColor3f(0, 165, 0); glVertex3f(-.3, -.05, 0);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 165, 0); glVertex3f(.3, -.05, 0);

glColor3f(0, 165, 0); glVertex3f(.3, -.79, 0);

glColor3f(0, 165, 0); glVertex3f(.38, -.79, 0);

glColor3f(0, 165, 0); glVertex3f(.38, -.05, 0);

glEnd();

//First Stack

glBegin(GL\_POLYGON);

glColor3f(165, 80, 0); glVertex3f(-.69, .16, 0);

glColor3f(165, 80, 0); glVertex3f(-.69, .05, 0);

glColor3f(165, 80, 0); glVertex3f(-.46, .05, 0);

glColor3f(165, 80, 0); glVertex3f(-.46, .16, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.68, .1);

char mesg[19] = "Giraffe Gal";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg[i]);

glBegin(GL\_POLYGON);

glColor3f(165, 0, 50); glVertex3f(-.69, .27, 0);

glColor3f(165, 0, 50); glVertex3f(-.69, .16, 0);

glColor3f(165, 0, 50); glVertex3f(-.46, .16, 0);

glColor3f(165, 0, 50); glVertex3f(-.46, .27, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.68, .21);

char mesg2[19] = "Methmatics";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg2[i]);

glBegin(GL\_POLYGON);

glColor3f(0, 80, 160); glVertex3f(-.69, .37, 0);

glColor3f(0, 80, 160); glVertex3f(-.69, .27, 0);

glColor3f(0, 80, 160); glVertex3f(-.46, .27, 0);

glColor3f(0, 80, 160); glVertex3f(-.46, .37, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.68, .33);

char mesg3[19] = "Scientology";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg3[i]);

glBegin(GL\_POLYGON);

glColor3f(0, 00, 80); glVertex3f(-.69, .48, 0);

glColor3f(0, 00, 80); glVertex3f(-.69, .37, 0);

glColor3f(0, 00, 80); glVertex3f(-.46, .37, 0);

glColor3f(0, 00, 80); glVertex3f(-.46, .48, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.68, .43);

char mesg4[19] = "Study of BAE";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg4[i]);

glBegin(GL\_POLYGON);

glColor3f(165, 0, 0); glVertex3f(-.69, .58, 0);

glColor3f(165, 0, 0); glVertex3f(-.69, .48, 0);

glColor3f(165, 0, 0); glVertex3f(-.46, .48, 0);

glColor3f(165, 0, 0); glVertex3f(-.46, .58, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.68, .53);

char mesg5[19] = "Bust";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg5[i]);

//SECOND STACK

glBegin(GL\_POLYGON);

glColor3f(.3, 0, .7); glVertex3f(-.46, .16, 0);

glColor3f(.3, 0, .7); glVertex3f(-.46, .05, 0);

glColor3f(.3, 0, .7); glVertex3f(-.23, .05, 0);

glColor3f(.3, 0, .7); glVertex3f(-.23, .16, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.45, .1);

char mesg6[19] = "The 69 Manual";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg6[i]);

glBegin(GL\_POLYGON);

glColor3f(.6, .5, .1); glVertex3f(-.46, .27, 0);

glColor3f(.6, .5, .1); glVertex3f(-.46, .16, 0);

glColor3f(.6, .5, .1); glVertex3f(-.23, .16, 0);

glColor3f(.6, .5, .1); glVertex3f(-.23, .27, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.45, .22);

char mesg7[19] = "Cow or now";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg7[i]);

glBegin(GL\_POLYGON);

glColor3f(.7, .3, .6); glVertex3f(-.46, .37, 0);

glColor3f(.7, .3, .6); glVertex3f(-.46, .27, 0);

glColor3f(.7, .3, .6); glVertex3f(-.23, .27, 0);

glColor3f(.7, .3, .6); glVertex3f(-.23, .37, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.45, .33);

char mesg8[19] = "Coke Digest";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg8[i]);

glBegin(GL\_POLYGON);

glColor3f(0, 80, 0); glVertex3f(-.46, .48, 0);

glColor3f(0, 80, 0); glVertex3f(-.46, .37, 0);

glColor3f(0, 80, 0); glVertex3f(-.23, .37, 0);

glColor3f(0, 80, 0); glVertex3f(-.23, .48, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.45, .43);

char mesg9[19] = "Smokey Attic";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg9[i]);

//THIRD STACK4

glBegin(GL\_POLYGON);

glColor3f(.5, .4, .3); glVertex3f(-.23, .16, 0);

glColor3f(.5, .4, .3); glVertex3f(-.23, .05, 0);

glColor3f(.5, .4, .3); glVertex3f(0, .05, 0);

glColor3f(.5, .4, .3); glVertex3f(0, .16, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(-.22, .10);

char mesg10[19] = "The Plunderer";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg10[i]);

//Fourth stack

glBegin(GL\_POLYGON);

glColor3f(.8, .4, .6); glVertex3f(0, .16, 0);

glColor3f(.8, .4, .6); glVertex3f(0, .05, 0);

glColor3f(.8, .4, .6); glVertex3f(.23, .05, 0);

glColor3f(.8, .4, .6); glVertex3f(.23, .16, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.01, .10);

char mesg11[19] = "I need Beans";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg11[i]);

glBegin(GL\_POLYGON);

glColor3f(.3, .3, .2); glVertex3f(0, .27, 0);

glColor3f(.3, .3, .2); glVertex3f(0, .16, 0);

glColor3f(.3, .3, .2); glVertex3f(.23, .16, 0);

glColor3f(.3, .3, .2); glVertex3f(.23, .27, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.01, .22);

char mesg12[19] = "Melon or Pear";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg12[i]);

glBegin(GL\_POLYGON);

glColor3f(.6, 0, .1); glVertex3f(0, .37, 0);

glColor3f(.6, 0, .1); glVertex3f(0, .27, 0);

glColor3f(.6, 0, .1); glVertex3f(.23, .27, 0);

glColor3f(.6, 0, .1); glVertex3f(.23, .37, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.01, .33);

char mesg13[19] = "Todd be White";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg13[i]);

//Fifth stack

glBegin(GL\_POLYGON);

glColor3f(.5, 0, .7); glVertex3f(.23, .16, 0);

glColor3f(.5, 0, .7); glVertex3f(.23, .05, 0);

glColor3f(.5, 0, .7); glVertex3f(.46, .05, 0);

glColor3f(.5, 0, .7); glVertex3f(.46, .16, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.24, .10);

char mesg14[19] = "Jack Black";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg14[i]);

glBegin(GL\_POLYGON);

glColor3f(.1, .8, .2); glVertex3f(.23, .27, 0);

glColor3f(.1, .8, .2); glVertex3f(.23, .16, 0);

glColor3f(.1, .8, .2); glVertex3f(.46, .16, 0);

glColor3f(.1, .8, .2); glVertex3f(.46, .27, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.24, .22);

char mesg15[19] = "Agent Banana";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg15[i]);

glBegin(GL\_POLYGON);

glColor3f(0, .5, .7); glVertex3f(.23, .37, 0);

glColor3f(0, .5, .7); glVertex3f(.23, .27, 0);

glColor3f(0, .5, .7); glVertex3f(.46, .27, 0);

glColor3f(0, .5, .7); glVertex3f(.46, .37, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.24, .33);

char mesg16[19] = "The Backwoods";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg16[i]);

glBegin(GL\_POLYGON);

glColor3f(.9, 0, .1); glVertex3f(.23, .48, 0);

glColor3f(.9, 0, .1); glVertex3f(.23, .37, 0);

glColor3f(.9, 0, .1); glVertex3f(.46, .37, 0);

glColor3f(.9, 0, .1); glVertex3f(.46, .48, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.24, .42);

char mesg17[19] = "Phil Bill";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg17[i]);

glBegin(GL\_POLYGON);

glColor3f(.5, .3, .1); glVertex3f(.23, .58, 0);

glColor3f(.5, .3, .1); glVertex3f(.23, .48, 0);

glColor3f(.5, .3, .1); glVertex3f(.46, .48, 0);

glColor3f(.5, .3, .1); glVertex3f(.46, .58, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.24, .53);

char mesg18[19] = "Lord Harambe";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg18[i]);

glBegin(GL\_POLYGON);

glColor3f(.7, 0, .2); glVertex3f(.23, .68, 0);

glColor3f(.7, 0, .2); glVertex3f(.23, .58, 0);

glColor3f(.7, 0, .2); glVertex3f(.46, .58, 0);

glColor3f(.7, 0, .2); glVertex3f(.46, .68, 0);

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(.24, .62);

char mesg19[19] = "Monsieur Blunt";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg19[i]);

//

glBegin(GL\_POLYGON);

glColor3f(.7, .1, .5); glVertex3f(x1, y1, 0); //x1, y1

glColor3f(.7, .1, .5); glVertex3f(x2, y2, 0); //x2, y2

glColor3f(.7, .1, .5); glVertex3f(x3, y3, 0); //x3, y3

glColor3f(.7, .1, .5); glVertex3f(x4, y4, 0); //x4, y4

glEnd();

glColor3f(0, 0.0, 0.0);

glRasterPos2d(xName, yName);

char mesg20[19] = "420: A Story";

for (int i = 0; i < 19; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesg20[i]);

glColor3f(0, 0.0, 0.0);

glRasterPos2d(0, -.75);

char mesgName[12] = "Jake Maurer";

for (int i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, mesgName[i]);

//Top Dongle

glFlush();

}

void keyboard(unsigned char key, int x, int y) {

switch (key)

{

case 'z':

x1 = x1 - (x1 \* .1); // - 10%

y1 = .68; // no movement

x2 = x2 - (x2 \* .1); // - 10%

y2 = y2 + (y2 \* .1); // + 10%

x3 = x3 + (x3 \* .1); // + 10%

y3 = y3 + (y3 \* .1); // + 10%

x4 = x4 + (x4 \* .1); // + 10%

y4 = .68; // no movement

xName = xName + (xName \* .05);

yName = yName + (yName \* .05);

break;

case 't':

x1 = x1 + .1;

x2 = x2 + .1;

x3 = x3 + .1;

x4 = x4 + .1;

xName = xName + .1;

default:

break;

}

glutPostRedisplay();

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(100, 100);

glutInitWindowSize(1000, 500);

glutCreateWindow("Fredonia");

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

glutMainLoop();

}

#include <stdio.h>

#include <GL\freeglut.h>

int rotation = 0;

void initialized(void)

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glShadeModel(GL\_SMOOTH);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_DEPTH\_TEST);

}

void display(void)

{

GLfloat lightPos[] = { 0.0, 0.0, 1.5, 1.0 };

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glPushMatrix();

glTranslatef(0.0, 0.0, -5.0);

glPushMatrix();

glRotated((GLdouble)rotation, .50, 0.1, 0.10);

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPos);

glTranslated(0.0, 0.0, 1.5);

glDisable(GL\_LIGHTING);

glEnable(GL\_LIGHTING);

glPopMatrix();

glutSolidSphere(1.0,20,20);

glPopMatrix();

glFlush();

}

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(30.0, (GLfloat)w / (GLfloat)h, .75, 15.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void keyboard(unsigned char key, int x, int y) {

if (key == 'm') {

rotation = (rotation + 10) % 360;

glutPostRedisplay();

}

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowPosition(100, 100);

glutInitWindowSize(1000, 500);

glutCreateWindow("Rotating light");

initialized();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

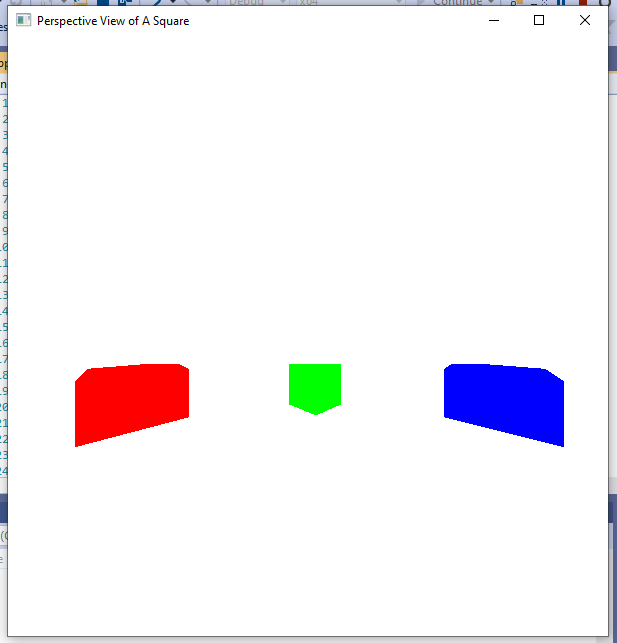
}

# HW5:

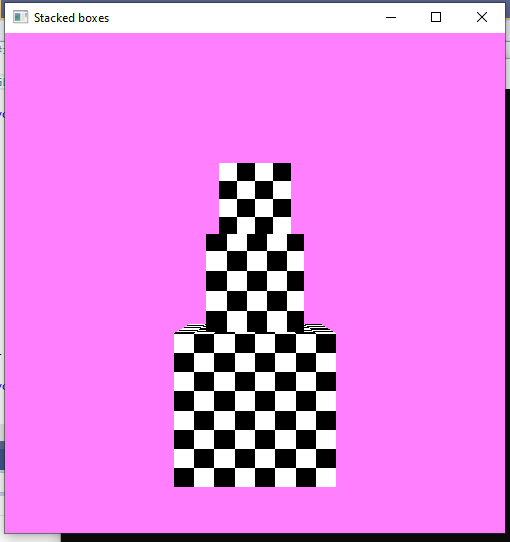
## Score: 100 out of 100

Refer to program listing below. It demonstrates perspective projection of a square using gluLookAt and gluFrustum functions.

Build the project as given and observe the results. Change the drawing to three identical and equally sized cubes. Place one cube farther from the view plane. When the program runs, it must show three cubes one of which is shown smaller than the other two. For 10% extra bonus points, at least two faces of the cubes should be shown with different colors. (Note gluSolidCube would not allow you to show different colors).



Refer to program listing given below from chapter 18 of the textbook. This program demonstrates how to generate a texture pattern and then how to apply it to your drawn object. Modify the pattern colors and create a 2-dimensional checkerboard pattern and re-apply it to three quad shapes resting on one another. Each upper shape is reduced in width as compared to the shape below it.



## Source Code:

#include <GL/glut.h>

GLint winWidth = 600, winHeight = 600; // Initial display-window size.

GLfloat x0 = 100.0, y0 = 50.0, z0 = 50.0; // Viewing-coordinate origin.

GLfloat xref = 50.0, yref = 50.0, zref = 0.0; // Look-at point.

GLfloat Vx = 0.0, Vy = 1.0, Vz = 0.0; // View-up vector.

/\* Set coordinate limits for the clipping window: \*/

GLfloat xwMin = -40.0, ywMin = -60.0, xwMax = 40.0, ywMax = 60.0;

/\* Set positions for near and far clipping planes: \*/

GLfloat dnear = 25.0, dfar = 125.0;

void init(void)

{

glClearColor(1.0, 1.0, 1.0, 0.0);

glMatrixMode(GL\_MODELVIEW);

gluLookAt(x0, y0, z0, xref, yref, zref, Vx, Vy, Vz);

glMatrixMode(GL\_PROJECTION);

glFrustum(xwMin, xwMax, ywMin, ywMax, dnear, dfar);

}

void displayFcn(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

/\* Set parameters for a square fill area. \*/

glColor3f(0.0, 1.0, 0.0); // Set fill color to green.

glPolygonMode(GL\_FRONT, GL\_FILL);

glPolygonMode(GL\_BACK, GL\_LINE); // Wire-frame back face.

glColor3f(1, 0, 0);

glPushMatrix();

glTranslatef(-60, 2, 1);

glutSolidCube(40);

glPopMatrix();

glColor3f(0, 1, 0);

glPushMatrix();

glTranslatef(40, 2, 1);

glutSolidCube(40);

glPopMatrix();

glColor3f(0, 0, 1);

glPushMatrix();

glTranslatef(140, 2, 1);

glutSolidCube(40);

glPopMatrix();

glFlush();

}

void reshapeFcn(GLint newWidth, GLint newHeight)

{

glViewport(0, 0, newWidth, newHeight);

winWidth = newWidth;

winHeight = newHeight;

}

void main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(50, 50);

glutInitWindowSize(winWidth, winHeight);

glutCreateWindow("Perspective View of A Square");

init();

glutDisplayFunc(displayFcn);

glutReshapeFunc(reshapeFcn);

glutMainLoop();

}

#include <gl/glut.h>

GLubyte image[128][128][3];

void make\_texture() {

GLubyte color;

for (int i = 0; i < 128; i++) {

for (int j = 0; j < 128; j++) {

if (((i & 8) == 0) ^ ((j & 8) == 0)) {

color = 255;

}

else {

color = 0;

}

image[i][j][0] = color;

image[i][j][1] = color;

image[i][j][2] = color;

}

}

}

void init(GLvoid) {

glClearColor(1.0f, .5f, 1.0f, 1.0f);

glEnable(GL\_DEPTH\_TEST);

make\_texture();

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, 128, 128, 0, GL\_RGB,

GL\_UNSIGNED\_BYTE, image);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP);

glEnable(GL\_TEXTURE\_2D);

}

void display(void) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

glTranslatef(0.0f, 0.0f, -5.0f);

glBegin(GL\_QUADS);

//MEDIUM SQUARE

glTexCoord2f(0.0f, 0.0f); glVertex3f(-.33f, -.33f, .33f);

glTexCoord2f(.3f, 0.0f); glVertex3f(.33f, -.33f, .33f);

glTexCoord2f(.3f, .3f); glVertex3f(.33f, .33f, .33f);

glTexCoord2f(0.0f, .3f); glVertex3f(-.33f, .33f, .33f);

glTexCoord2f(.3f, 0.0f); glVertex3f(-.33f, -.33f, -.33f);

glTexCoord2f(.3f, .3f); glVertex3f(-.33f, .33f, -.33f);

glTexCoord2f(0.0f, .3f); glVertex3f(.33f, .33f, -.33f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(.33f, -.33f, -.33f);

glTexCoord2f(0.0f, .3f); glVertex3f(-.33f, .33f, -.33f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-.33f, .33f, .33f);

glTexCoord2f(.3f, 0.0f); glVertex3f(.33f, .33f, .33f);

glTexCoord2f(.3f, .3f); glVertex3f(.33f, .33f, -.33f);

glTexCoord2f(.3f, .3f); glVertex3f(-.33f, -.33f, -.33f);

glTexCoord2f(0.0f, .3f); glVertex3f(.33f, -.33f, -.33f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(.33f, -.33f, .33f);

glTexCoord2f(.3f, 0.0f); glVertex3f(-.33f, -.33f, .33f);

glTexCoord2f(.3f, 0.0f); glVertex3f(.33f, -.33f, -.33f);

glTexCoord2f(.3f, .3f); glVertex3f(.33f, .33f, -.33f);

glTexCoord2f(0.0f, .3f); glVertex3f(.33f, .33f, .33f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(.33f, -.33f, .33f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-.33f, -.33f, -.33f);

glTexCoord2f(.3f, 0.0f); glVertex3f(-.33f, -.33f, .33f);

glTexCoord2f(.3f, .3f); glVertex3f(-.33f, .33f, .33f);

glTexCoord2f(0.0f, .3f); glVertex3f(-.33f, .33f, -.33f);

//LARGE SQUARE

glTexCoord2f(0.0f, 0.0f); glVertex3f(-.53f, -1.33f, .53f);

glTexCoord2f(.5f, 0.0f); glVertex3f(.53f, -1.33f, .53f);

glTexCoord2f(.5f, .5f); glVertex3f(.53f, -.33f, .53f);

glTexCoord2f(0.0f, .5f); glVertex3f(-.53f, -.33f, .53f);

glTexCoord2f(.5f, 0.0f); glVertex3f(-.53f, -1.33f, -.53f);

glTexCoord2f(.5f, .5f); glVertex3f(-.53f, -.33f, -.53f);

glTexCoord2f(0.0f, .5f); glVertex3f(.53f, -.33f, -.53f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(.53f, -1.33f, -.53f);

glTexCoord2f(0.0f, .5f); glVertex3f(-.53f, -.33f, -.53f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-.53f, -.33f, .53f);

glTexCoord2f(.5f, 0.0f); glVertex3f(.53f, -.33f, .53f);

glTexCoord2f(.5f, .5f); glVertex3f(.53f, -.33f, -.53f);

glTexCoord2f(.5f, .5f); glVertex3f(-.53f, -1.33f, -.53f);

glTexCoord2f(0.0f, .5f); glVertex3f(.53f, -1.33f, -.53f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(.53f, -1.33f, .53f);

glTexCoord2f(.5f, 0.0f); glVertex3f(-.53f, -1.33f, .53f);

glTexCoord2f(.5f, 0.0f); glVertex3f(.53f, -1.33f, -.53f);

glTexCoord2f(.5f, .5f); glVertex3f(.53f, -.33f, -.53f);

glTexCoord2f(0.0f, .5f); glVertex3f(.53f, -.33f, .53f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(.53f, -1.33f, .53f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-.53f, -1.33f, -.53f);

glTexCoord2f(.5f, 0.0f); glVertex3f(-.53f, -1.33f, .53f);

glTexCoord2f(.5f, .5f); glVertex3f(-.53f, -.33f, .53f);

glTexCoord2f(0.0f, .5f); glVertex3f(-.53f, -.33f, -.53f);

//SMALL SQUARE

glTexCoord2f(0.0f, 0.0f); glVertex3f(-.25f, .33f, .25f);

glTexCoord2f(.25f, 0.0f); glVertex3f(.25f, .33f, .25f);

glTexCoord2f(.25f, .25f); glVertex3f(.25f, .83f, .25f);

glTexCoord2f(0.0f, .25f); glVertex3f(-.25f, .83f, .25f);

glTexCoord2f(.25f, 0.0f); glVertex3f(-.25f, .33f, -.25f);

glTexCoord2f(.25f, .25f); glVertex3f(-.25f, .83f, -.25f);

glTexCoord2f(0.0f, .25f); glVertex3f(.25f, .83f, -.25f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(.25f, .33f, -.25f);

glTexCoord2f(0.0f, .25f); glVertex3f(-.25f, .83f, -.25f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-.25f, .83f, .25f);

glTexCoord2f(.25f, 0.0f); glVertex3f(.25f, .83f, .25f);

glTexCoord2f(.25f, .25f); glVertex3f(.25f, .83f, -.25f);

glTexCoord2f(.25f, .25f); glVertex3f(-.25f, .33f, -.25f);

glTexCoord2f(0.0f, .25f); glVertex3f(.25f, .33f, -.25f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(.25f, .33f, .25f);

glTexCoord2f(.25f, 0.0f); glVertex3f(-.25f, .33f, .25f);

glTexCoord2f(.25f, 0.0f); glVertex3f(.25f, .33f, -.25f);

glTexCoord2f(.25f, .25f); glVertex3f(.25f, .83f, -.25f);

glTexCoord2f(0.0f, .25f); glVertex3f(.25f, .83f, .25f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(.25f, .33f, .25f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-.25f, .33f, -.25f);

glTexCoord2f(.25f, 0.0f); glVertex3f(-.25f, .33f, .25f);

glTexCoord2f(.25f, .25f); glVertex3f(-.25f, .83f, .25f);

glTexCoord2f(0.0f, .25f); glVertex3f(-.25f, .83f, -.25f);

glEnd();

glutSwapBuffers();

}

void reshape(GLsizei width, GLsizei height) {

glMatrixMode(GL\_PROJECTION);

gluPerspective(40.0, (float)width / (float)height, 1.0, 50.0);

glMatrixMode(GL\_MODELVIEW);

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Stacked boxes");

glutDisplayFunc(display);

glutReshapeFunc(reshape);

init();

glutMainLoop();

return 0;

}