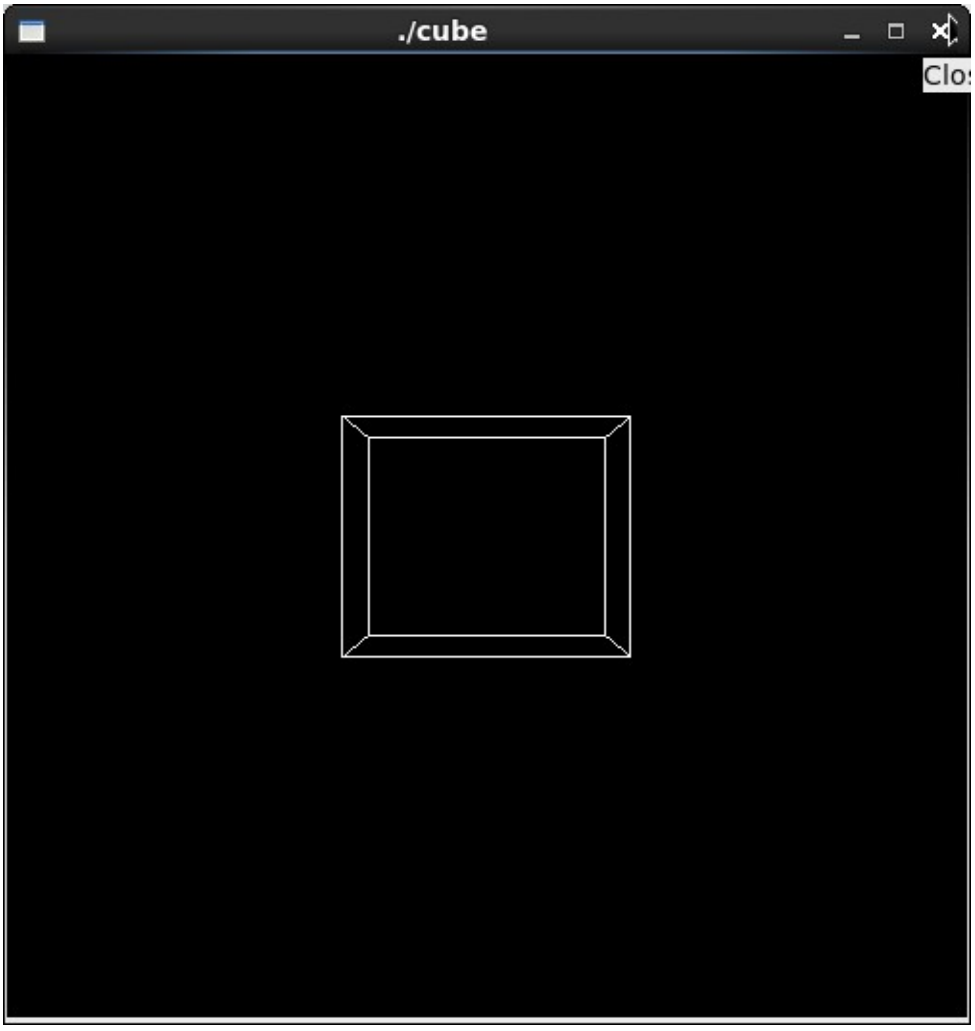
CSE420
Samuel Marrujo
Professor Yu
Lab 12

Viewing

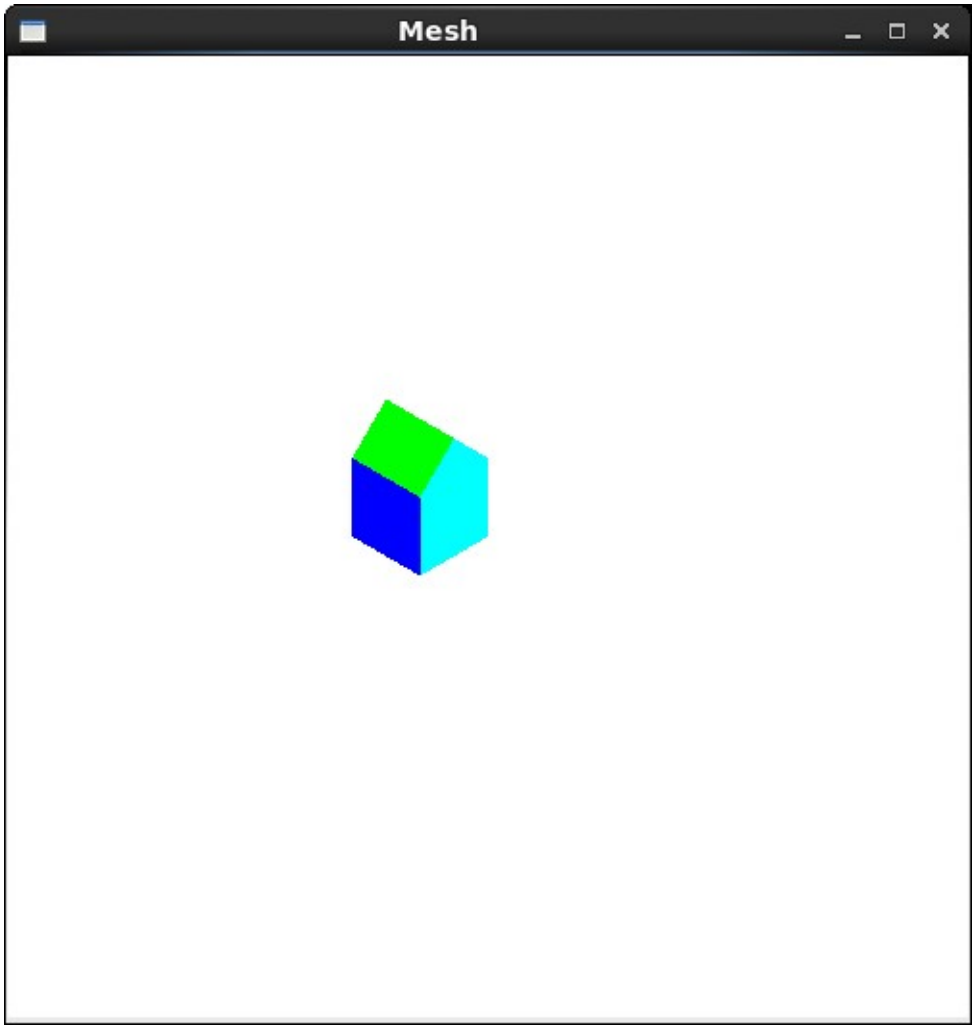
In this part of the lab, it was a modification of the original cube program. After some modifications to the program, and adding in the functions asked for, the result was the following reproduction of the figure. This was very similar to the previous part of the lab. Because of this change to the functions and the addition of the figure, I believe I was able to accomplish this task successfully, since there were no errors and the display window was changed. Here are my results for this program:



```
//cube.cpp
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
}

void display(void)
{
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 1.0);
   glLoadIdentity ();             /* clear the matrix */
           /* viewing transformation  */
   gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); //switched it back
   glScalef (1.0, 1.5, 1.0);      /* modeling transformation */
   glutWireCube (1.0);
   glFlush ();
}

void reshape (int w, int h)
{
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity ();
   glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
   glMatrixMode (GL_MODELVIEW);
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

Rotation Animation

In this part of the lab, it was a modification of the original cube program. After some modifications to the program, and adding in the functions asked for, the result the following screenshot. Because of this change to the functions and the addition of the figure, I believe I was able to accomplish this task successfully, since there were no errors and the display window was changed. Here are my results for this program:



```
//mesh.cpp
#include "mesh.h"
#include <SDL/SDL.h>

using namespace std;

Mesh::Mesh() //constructor
{
  numVerts = numFaces = numNormals = 0;
  pt = NULL;
  norm  =  NULL;
  face = NULL;
}

bool Mesh::isEmpty()
{
  return (numVerts == 0) || (numFaces == 0) || (numNormals == 0);
}

void Mesh::setColor( int n )
{
  if ( n == 1 )
    glColor3f( 1, 0, 0 );
  else if ( n == 2 )
    glColor3f( 0, 1, 0 );
  else if ( n == 3 )
    glColor3f( 0, 0, 1 );
  else if ( n == 4 )
    glColor3f( 1, 1, 0 );
  else if ( n == 5 )
    glColor3f( 1, 0, 1 );
  else if ( n == 6 )
    glColor3f( 0, 1, 1 );
  else
    glColor3f( 0, 0, 0 );
}

void Mesh::drawMesh()        // use OpenGL to draw this mesh
{
  // draw each face of this mesh using OpenGL: draw each polygon.
  if( isEmpty() ) return; // mesh is empty

  glEnable( GL_CULL_FACE );
  glCullFace ( GL_BACK );
  for(int f = 0; f < numFaces; f++) // draw each face
  //for(int f = 6; f < numFaces; f++) // draw each face
  {
    glBegin(GL_POLYGON);
    cout << endl;
```

```
    setColor( f );
    for(int v = 0; v < face[f].nVerts; v++) // for each vertex
    {
          int in = face[f].vert[v].normIndex ; // index of this normal
        int iv =  face[f].vert[v].vertIndex ; // index of this vertex
        glNormal3f(norm[in].x, norm[in].y, norm[in].z);
          cout << "[" << norm[in].x << "," << norm[in].y << "," <<
                norm[in].z << "]" << "    ";
        glVertex3f(pt[iv].x, pt[iv].y, pt[iv].z);
          cout << "(" << pt[iv].x << "," << pt[iv].y << "," <<
                pt[iv].z << ")" << "    ";

    }

    glEnd();
  //  SDL_Delay ( 1000 );
    glFlush ();
    cout << endl;
  }
} //drawMesh

//read Mesh data from file
int Mesh:: readFile(char * fileName)
{
  fstream infile;
  infile.open(fileName, ios::in);
  cout << "opening file " << endl;
  if(infile.fail()) return -1; // error - can't open file
  if(infile.eof())  return -1; // error - empty file
  infile >> numVerts >> numNormals >> numFaces;
  pt = new Point3[numVerts];
  norm = new Vector3[numNormals];
  face = new Face[numFaces];
  //check that enough memory was found:
  if( !pt || !norm || !face)return -1; // out of memory
  cout << "file open O.K. " << endl;

  for(int p = 0; p < numVerts; p++) // read the vertices
    infile >> pt[p].x >> pt[p].y >> pt[p].z;
  for(int n = 0; n < numNormals; n++) // read the normals
    infile >> norm[n].x >> norm[n].y >> norm[n].z;
  cout << "numFaces = " << numFaces << endl;
  for(int f = 0; f < numFaces; f++)// read the faces
  {
    infile >> face[f].nVerts;

    face[f].vert = new VertexID[face[f].nVerts];
    for(int i = 0; i < face[f].nVerts; i++)
          infile >> face[f].vert[i].vertIndex;
```

```
   for(int i = 0; i < face[f].nVerts; i++)
          infile  >> face[f].vert[i].normIndex;
  }
  return 0; // success
} //readFile
```

```
//display.cpp
#include "mesh.h"
#include <SDL/SDL.h>

#define DATA_FILE  "data.txt"

void display(void)
{
  glMatrixMode( GL_PROJECTION );
  glLoadIdentity();
  glOrtho(-5.0, 5.0, -5.0, 5.0, 0.1, 100 );
  glMatrixMode(GL_MODELVIEW); // position and aim the camera
  glLoadIdentity();
  gluLookAt(8.0, 8.0, 8.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

  glClear(GL_COLOR_BUFFER_BIT);
  Mesh msh;
  msh.readFile( DATA_FILE );
  msh.drawMesh();
  glFlush();
  SDL_Delay ( 1000 );
  glClear(GL_COLOR_BUFFER_BIT);
  //glPushMatrix();                 //save current matrix M
  //glTranslatef ( 0.0, 0, 0 );     //move in x-direction
  glRotatef( 90, 0, 1, 0 );
  msh.drawMesh();
  glFlush();
  SDL_Delay ( 1000 );
  glClear(GL_COLOR_BUFFER_BIT);
  glRotatef ( 90, 0, 1, 0 );        //rotate about  y-axis for 90
  msh.drawMesh();                   //rotate then translate
  glFlush();
  SDL_Delay ( 1000 );
  glPopMatrix();                    //restore maxtrix M
  glClear(GL_COLOR_BUFFER_BIT);
  glRotatef ( 90, 0, 1, 0 );
  glTranslatef (0.0, 0, 0 );        //translate then rotate
  msh.drawMesh();
  glFlush();
  SDL_Delay(1000);
}
```