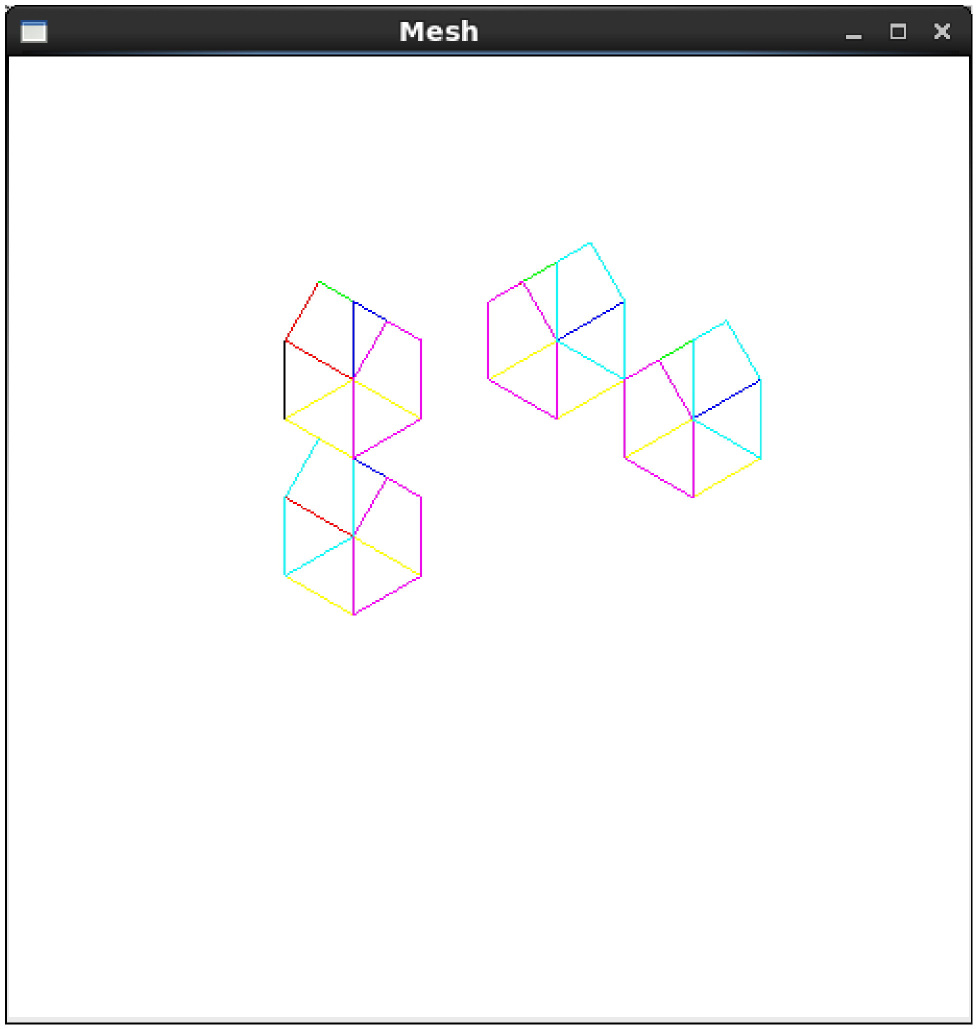
CSE420
Samuel Marrujo
Professor Yu
Lab 10

Mesh1

In this part of the lab, it was a modification of the original mesh1 program. After some modifications to the program, and adding in the functions asked for, the result was the following reproduction of the figure. 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 );
} //changed the colors back

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_LINE_LOOP);
    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-2, pt[iv].y, pt[iv].z-3);
        cout << "(" << pt[iv].x << "," << pt[iv].y << "," <<
              pt[iv].z << ")" << "    ";
    }
    glEnd();
    SDL_Delay ( 2000 );
    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
```