Yazhuo Liu
CSE 420
Homework 2

1. (10 points) For each of the following triplets of points, find the normal vectors manually to the plane (if it exists) that passes through the triplet. Show your steps.

- ***a. (1, 1, 1), (1, 2, 1), (3, 0, 4)***
- ***b. (6, 3, -4), (0, 0, 0), (2, 1, -1)***

a. A = P2 – P1 = (0, 1, 0) B = P3 – P1 = (2, -1, 3)

i	j	k	-2k		
0	1	0			i = 3
2	-1	3	3i		j = 0
i	j	k			k = -2
0	1	0			

A x B = (3, 0, -2)

b. A = P2 – P1 = (-6, -3, 4) B = P3 – P1 = (-4, -2, 3)

i	j	k	-12k		
-6	-3	4	8i		i = -1
-4	-2	3	-9i	18j	j = 2
i	j	k	12k		k = 0
-6	-3	4	-16j		

A x B = (-1, 2, 0)

Find the normalized normal to the plane 5x - 3y + 6z = 7 and determine if the points P1 = (1, 5, 2) and P2 = (-3, -1, 2) are on the same side of the plane.

$70^{1/2}$ (5, -3, 6)

5*1 – 3*5 + 6*2 -7 = -5

5(-3) – 3(-1) + 6*2 -7 = -7

Therefore, P1 and P2 are on the same side of the plane, and they are both under the plane.

2. (10 points) Find the normalized normal at the point (1, 2, 3) for each of the following two cases:

a) A sphere: $x^2 + y^2 + z^2 = 14$
b)A plane: 3x - 4y + 2z - 1 = 0

a) $14^{1/2}$ (1, 2, 3)

b) $29^{1/2}$ (3, 4, 2)

3. (10 points) A robust method to find the normal to any polygon with N vertices is called the Newell's method. It computes the components of the normal n according to the following formulas:

$$n_x = \sum_{i=0}^{n-1} (y_i - y_{i+1})(z_i + z_{i+1})$$

$$n_y = \sum_{i=0}^{n-1} (z_i - z_{i+1})(x_i + x_{i+1})$$

$$n_z = \sum_{i=0}^{n-1} (x_i - x_{i+1})(y_i + y_{i+1})$$

a) Apply the Newell's method to a. of Question 1, and see whether you get the same answer.

b) Find the normal to the polygon (1, 1, 2), (2, 0, 5), (5, 1, 4), (6, 0, 7).

a) 1. $x_0 = 1, x_1 = 1, x_2 = 3,$

$y_0 = 1, y_1 = 2, y_2 = 0,$

$z_0 = 1, z_1 = 1, z_2 = 4,$

$n_x = (1-2)(1+1) + (2-0)(1+4) + (0-1)(4+1) = 3$

$n_y = (1-1)(1+1) + (1-4)(1+3) + (4-1)(3+1) = 0$

$n_z = (1-1)(1+2) + (1-3)(2+0) + (3-1)(0+1) = -2$

n = (3, 0, -2)

2. $x_0 = 6, x_1 = 0, x_2 = 2,$

$y_0 = 3, y_1 = 0, y_2 = 1,$

$z_0 = -4, z_1 = 0, z_2 = -1,$

$n_x = (3-0)(-4+0) + (0-1)(0-1) + (1-3)(-1-4) = -1$

$n_y = (-4-0)(6+0) + (0+1)(0+2) + (-1+4)(2+6) = 2$

$n_z = (6-0)(3+0) + (0-2)(0+1) + (2-6)(1+3) = 0$

n = (-1, 2, 0)

Therefore, they have the same answer.

b) $x_0 = 1, x_1 = 2, x_2 = 5, x_3 = 6,$

$y_0 = 1, y_1 = 0, y_2 = 1, y_3 = 0,$

$z_0 = 2, z_1 = 5, z_2 = 4, z_3 = 7,$

$n_x = (1-0)(2+5) + (0-1)(5+4) + (1-0)(4+7) + (0-1)(7+2) = 0$

$n_y = (2-5)(1+2) + (5-4)(2+5) + (4-7)(5+6) + (7-2)(6+1) = 0$

$n_z = (1-2)(1+0) + (2-5)(0+1) + (5-6)(1+0) + (6-1)(0+1) = 0$

n = (0, 0, 0)

4. (10 points) Let vectors A = (2, -1, 1)^T and B = (1, 1, -1)^T. Find

- ***a. the angle between A and B,***
- ***b. a unit vector perpendicular to both A and B.***

a) A . B = | A | | B | cos θ

A . B = 2 – 1 – 1 = 0 | A | = sqrt 6 | B | = sqrt 3

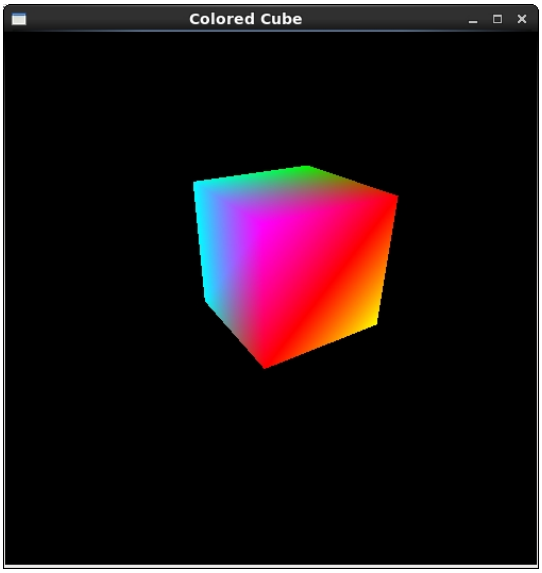
0 = sqrt 6 * sqrt 3 * cos θ θ = 90°

b) n = A x B = (0, 3, 3)

length of n = |n| = ($0 + 3^2 + 3^2$)$^{1/2}$ = $3*2^{1/2}$

n / $3*2^{1/2}$ = (0, sqrt 2, sqrt 2)

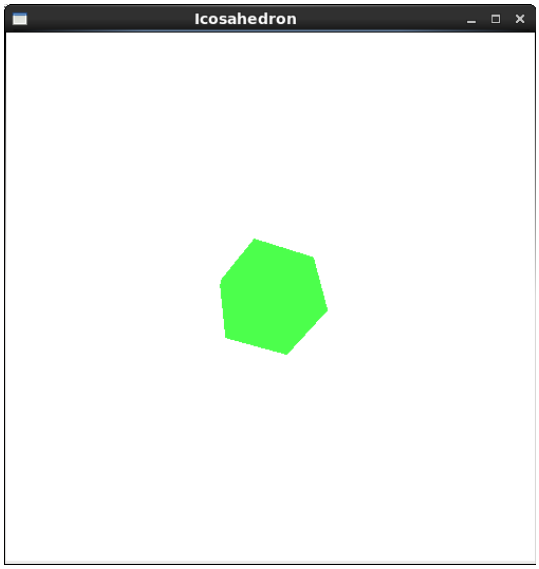
5. (10 points) Use glDrawElements() to draw the following cube with each face having a different color.




```
void display(void)
{
  glClear (GL_COLOR_BUFFER_BIT);
  glColor3f (1.0, 1.0, 1.0);
  glLoadIdentity ();
  gluLookAt (-3.0, 3.5, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  static GLint vertices[] = {-1, -1, -1,   //0
                  1, -1, -1,
                  1, 1, -1,
                  -1, 1, -1,       //3
                      -1, -1, 1,
                      1, -1, 1,
                      1, 1, 1,        //6
                      -1, 1, 1};      //7
```

```
  static GLfloat colors[] = {0.0, 1.0, 1.0,
                    1.0, 1.0, 0.0,
                    0.0, 1.0, 0.0,
                    0.0, 1.0, 1.0,
                    1.0, 0.0, 0.0,
                    1.0, 1.0, 0.0,
                     1.0, 0.0, 0.0,
                     1.0, 0.0, 1.0
                     };
  glVertexPointer (3, GL_INT, 0, vertices);
  glEnable( GL_CULL_FACE );
  glCullFace ( GL_BACK );
  glColorPointer (3, GL_FLOAT, 0, colors);
  glTranslatef ( 0, 1, 0.5 );
  static GLubyte allIndices[] = {4, 5, 6, 7, 1, 2, 6, 5,
        0, 1, 5, 4, 0, 3, 2, 1,
        0, 4, 7, 3, 2, 3, 7, 6};
  glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, allIndices);
  glFlush ();
}
```

6. (10 points) Write a program or programs to reproduce one of the following figures of icosahedron and dodecahedron (extra credit for both).



```
#include <GL/glut.h>
#include <stdlib.h>
#include <cmath>

#define a .525731112119133606
#define b .850650808352039932

void init(void)
{
   glClearColor (1.0f, 1.0f, 1.0f, 1.0f);
   glShadeModel (GL_SMOOTH);
   glEnableClientState (GL_VERTEX_ARRAY);
   glEnableClientState (GL_COLOR_ARRAY);
}

void normalize(float v[3]) {
   GLfloat d = sqrt(v[0]*v[0]+v[1]*v[1]+v[2]*v[2]);
   if (d == 0.0) {
      return;
   }
   v[0] /= d; v[1] /= d; v[2] /= d;
}

void normcrossprod(float v1[3], float v2[3], float out[3])
{
   GLint i, j;
   GLfloat length;

   out[0] = v1[1]*v2[2] - v1[2]*v2[1];
   out[1] = v1[2]*v2[0] - v1[0]*v2[2];
   out[2] = v1[0]*v2[1] - v1[1]*v2[0];
   normalize(out);
}
```

```
static GLfloat vdata[12][3] = {
   {-a, 0.0, b}, {a, 0.0, b}, {-a, 0.0, -b}, {a, 0.0, -b},
   {0.0, b, a}, {0.0, b, -a}, {0.0, -b, a}, {0.0, -b, -a},
   {b, a, 0.0}, {-b, a, 0.0}, {b, -a, 0.0}, {-b, -a, 0.0}
};

static GLuint tindices[20][3] = {
   {0,4,1}, {0,9,4}, {9,5,4}, {4,5,8}, {4,8,1},
   {8,10,1}, {8,3,10}, {5,3,8}, {5,2,3}, {2,7,3},
   {7,10,3}, {7,6,10}, {7,11,6}, {11,0,6}, {0,1,6},
   {6,1,10}, {9,0,11}, {9,11,2}, {9,2,5}, {7,2,11}
};

void display(void)
{
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (0.0, 0.0, 0.0);
   glLoadIdentity ();
   gluLookAt (-3.0, 3.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

   int i, j;

   glBegin(GL_TRIANGLES);
     for (i = 0; i < 20; i++) {
        glColor3f(0.3, 1.0, 0.3);
        glVertex3fv(&vdata[tindices[i][0]][0]);
        glVertex3fv(&vdata[tindices[i][1]][0]);
        glVertex3fv(&vdata[tindices[i][2]][0]);
     }
   glEnd();

   GLfloat d1[3], d2[3], norm[3];
   for (j = 0; j < 3; j++) {
      d1[j] = vdata[tindices[i][0]][j] - vdata[tindices[i][1]][j];
```

```
      d2[j] = vdata[tindices[i][2]][j] - vdata[tindices[i][1]][j];
    }
    normcrossprod(d1, d2, norm);
    glNormal3fv(norm);

    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 ("Icosahedron");
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
```

Report:

I finished the first 5 problems without so much trouble, but the last one is kind of tricky. All I got was a shape of the icosahedron but not the lines on it. I can't figure out where I did wrong but I will try to fix it later.