**CSE 4200 Homework 2- Spencer Wallace**

**Summary:**

All parts completed successfully. Because I was able to complete all parts, I am giving myself a full score **60/60.**

**Part 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.

1. **P1**(1, 1, 1), **P2**(1, 2, 1), **P3**(3, 0, 4)

A = P2 – P1 = (1,2,1) - (1,1,1) = (0,1,0)

B = P3 – P1 = (3,0,4) - (1,1,1) = (2,-1,3)

n = AxB

|0 1 0| = i(3-0) - j(0-0) + k(0-2) = 3i – 2k = **(3,0,-2)**

|2 -1 3|

n = n/|n| = (3,0,-2)/(3­­­­2+0+(-2)2)1/2 = (3,0,-2)/(131/2) = **(0.832, 0, -0.555) <= normalized**

1. **P1**(6, 3, -4), **P2**(0, 0, 0), **P3**(2, 1, -1)

A = P2 – P1 = (0,0,0) - (6,3,-4) = (-6, -3, 4)

B = P3 – P1 = (2,1,-1) - (6,3,-4) = (-4, -2, 3)

n = AxB

|-6 -3 4| = i(-9 - -8) - j(-18 - -16) + k(12 - 12) = -i – -2j = **(-1,2,0)**

|-4 -2 3|

n = n/|n| = (-1,2,0)/( (-1)­­­­2+22+0)1/2 = (-1,2,0)/(51/2) = **(-0.447, 0.894, 0) <= normalized**

( You may check your answer using the program you wrote in the lab. )

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.

n = n/|n| = (5, -3, 6)/( 52 + (-3)2 + 62)1/2 = (5, -3, 6)/(701/2) =

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

a) A sphere: x2 + y2 + z2 = 14

(1,2,3) – (0,0,0) = (1,2,3)

n = n/|n| = (1,2,3)/(1+4+9)1/2 =   
b) A plane: 3x - 4y + 2z - 1 = 0

“For a plane, one perpendicular direction is the same for every point on the surface” – Notes pt.6

n = n/|n| = (3,-4,2)/(9+16+4)1/2 =

**Part 3)** ( 10 points ) A robust method to find the normal to any polygon with N vertices is called the Newell's method.

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

(1,1,1) (1,2,1) (3,0,4)

Nx = (1-2)\*(1+1) + (2-0)\*(1+4) + (0-1)\*(4+1)= -2 + 10 - 5 = **3**

Ny = (1-1) **0**  + (1-4)\*(1+3) + (4-1)\*(3+1)= 0 + -12 + 12 = **0**

Nz = (1-1) **0** + (1-3)\*(2+0) + (3-1)\*(0+1) = 0 + -4 + 2 = **-2**

Using Newell’s method does give the same answer ✓

b) Find the normal to the polygon (0, 0, 3), (3, 0, 0), (2, 2, -1), (-1, 5, -1), (1, 1, 1)

Nx = (0-0) **0** + (0-2)\*(0+ -1) + (2-5)\*(-1 + -1) + (5-1)\*(**0**) + (1-0)\*(1+3) = 0 + 2 + 6 + 0 + 4 = **12**

Ny = (3-0)\*(0+3) + (0 - -1)\*(3+2) + (-1 - -1)**0** + (-1 – 1)\*(-1+1)**0** + (1-3)\*(1+0) = 9 + 5 + -2 = **12**

Nz = (0-3)\*(**0**) + (3-2)\*(0+2) + (2 - -1)\*(2+5) + (-1 - 1)\*(5+1) + (1 - 0)\*(1+0) = 0 +2 + 21 + -12 +1 = **12**

**N = (12,12,12)**

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

1. the angle between A and B,

angle = cos‑1( A\*B/(|A||B|) )

A\*B = 2 + -1 + -1 = **0**

(check that bottom isn’t 0 to check for undefined)

|A| = (4 + 1 + 1)1/2 = 61/2

|B| = (1 + 1 + 1)1/2 = 31/2

angle = cos‑1(0) = **90 degrees or 1/2π**

1. a unit vector perpendicular to both A and B.

AxB = | 2 -1 1 | = (1 - 1)i - (-2 – 1)j + (2 - -1)k = **(0, 3, 3)T**

| 1 1 -1 |

V = v/|v| = (0,3,3)/(0+9+9)1/2 = **(0, 0.707, 0.707)T**

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

Rectangle

Description automatically generated with low confidenceShape

Description automatically generated

**Code**

[007463307@csusb.edu@jb359-4 hw2]$ cat cube.cpp

#include <stdio.h>

#define GL\_GLEXT\_PROTOTYPES

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glx.h>

#include <GL/glext.h>

#include <GL/glut.h>

GLfloat cube[] = {

//back

125, 175, 0.7, 0.7, 0,//0

350, 175, 0.7, 0, 0.7, //1

350, 375, 0.7, 0, 0, //2

125, 375, 0, 0, 0.7,//3

//front

200, 100, 0, 0.7, 0.7,//4

450, 100, 0.7, 0.7, 0.7,//5

450, 300, 0.9, 0, 0.9,//6

200, 300, 0, 0.9, 0.9//7

};

GLubyte front[] = {4,5,6,7};

GLubyte right[] = {1,2,6,5};

GLubyte left[] = {0,4,7,3};

GLubyte back[] = {0,3,2,1};

GLubyte top[] = {2,3,7,6};

GLubyte bottom[] = {0,1,5,4};

unsigned int vbo; //vertex buffer object

unsigned int ind;

//initialization

void init( void )

{

glClearColor( 1.0, 1.0, 1.0, 0.0 ); //get white background color

glColor3f( 0.0f, 0.0f, 0.0f ); //set drawing color

gluOrtho2D( 0.0, 500.0, 0.0, 500.0 );

glGenBuffers ( 1, &vbo ); //handle to vertex buffer object

glBindBuffer ( GL\_ARRAY\_BUFFER, vbo );

glBufferData ( GL\_ARRAY\_BUFFER, sizeof ( cube ), cube, GL\_STATIC\_DRAW );

}

void display( void )

{

glClear( GL\_COLOR\_BUFFER\_BIT ); //clear screen

glEnableClientState(GL\_VERTEX\_ARRAY);

glEnableClientState(GL\_COLOR\_ARRAY);

glVertexPointer(2, GL\_FLOAT, 5\*sizeof(GLfloat), 0);

glEnable(GL\_CULL\_FACE);

glCullFace(GL\_BACK);

glColorPointer ( 3, GL\_FLOAT, 5\*sizeof(GLfloat), (void\*)(2\*sizeof(GLfloat)));

glDrawElements ( GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, front );

// glColorPointer ( 3, GL\_FLOAT, 3\*sizeof(float), right);

//glColor3f(0.7,0.7,0);

glDrawElements ( GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, right );

//glColorPointer ( 3, GL\_FLOAT, 3\*sizeof(float), left);

// glColor3f(0.7,0,0.7);

glDrawElements ( GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, left );

//glColorPointer ( 3, GL\_FLOAT, 3\*sizeof(float), back);

//glColor3f(0.7,0,0);

glDrawElements ( GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, back );

//glColorPointer ( 3, GL\_FLOAT, 3\*sizeof(float), top);

//glColor3f(0,0.7,0.7);

glDrawElements ( GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, top );

//glColorPointer ( 3, GL\_FLOAT, 3\*sizeof(float), bottom);

//glColor3f(0,0.7,0);

glDrawElements ( GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, bottom );

glDisableClientState ( GL\_VERTEX\_ARRAY );

glDisableClientState ( GL\_COLOR\_ARRAY );

glFlush(); //send all output to screen

}

void keyboard ( unsigned char key, int mousex, int mousey )

{

switch ( key ) {

case 27: // escape

exit ( -1 );

}

glutPostRedisplay();

}

void specialKey ( int key, int mousex, int mousey )

{

switch ( key ) {

case GLUT\_KEY\_UP:

break;

case GLUT\_KEY\_DOWN:

break;

case GLUT\_KEY\_LEFT:

break;

case GLUT\_KEY\_RIGHT:

break;

}

glutPostRedisplay();

}

void myMouse( int button, int state, int x, int y )

{

glFlush(); //send all output to screen

}

/\* Main Loop

\* Open window with initial window size, title bar,

\* RGBA display mode, depth buffer.

\*/

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

{

glutInit(&argc, argv); //initialize toolkit

glutInitDisplayMode (GLUT\_SINGLE| GLUT\_RGB ); //set display mode

glutInitWindowSize(500, 500); //set window size on screen

glutInitWindowPosition( 100, 150 ); //set window position on screen

glutCreateWindow(argv[0]); //open screen widow

init();

glutDisplayFunc (display); //points to display function

glutKeyboardFunc ( keyboard );

// glutSpecialFunc( specialKey );

glutMouseFunc( myMouse );

glutMainLoop(); //go into perpetual loop

return 0;

}

**Part 6)** ( 10 points ) Write a program or programs to reproduce one of the following figures of **icosahedron** and dodecahedron

A close-up of a green leaf

Description automatically generated with low confidence A picture containing text, umbrella, accessory

Description automatically generated

**Code**

#include <GL/glut.h>

#include <GL/gl.h>

#include <stdlib.h>

using namespace std;

#define a .525731112119133606

#define b .850650808352039932

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}

};

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 init(void)

{

glClearColor(1.0, 1.0, 1.0, 0.0); //get white background color

glShadeModel(GL\_SMOOTH);

glEnable(GL\_SMOOTH);

glEnable(GL\_POINT\_SMOOTH);

glEnableClientState(GL\_VERTEX\_ARRAY);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

glEnable( GL\_BLEND );

}

void display()

{

glLoadIdentity();

gluLookAt(2.0, 2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

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

glRotatef(25, 0, 0, 1);

glBegin(GL\_TRIANGLES);

glColor4f(0.3, 0.5, 0.2, 0.7);

for (int i = 0; i < 20; i++) {

glVertex3fv(&vdata[tindices[i][0]][0]);

glVertex3fv(&vdata[tindices[i][1]][0]);

glVertex3fv(&vdata[tindices[i][2]][0]);

}

glEnd();

glColor4f(0.0, 0.2, 0.1, 1);

glLineWidth(2.25);

for (int i = 0; i < 20; i++) {

glBegin(GL\_LINE\_LOOP);

glVertex3fv(&vdata[tindices[i][0]][0]);

glVertex3fv(&vdata[tindices[i][1]][0]);

glVertex3fv(&vdata[tindices[i][2]][0]);

}

glEnd();

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); //initialize toolkit

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); //set display mode: single bufferring, RGB model

glutInitWindowPosition(500, 150); //set window position on screen

glutInitWindowSize(500, 500);

glutCreateWindow(\*argv); //open screen window

init();

glutDisplayFunc(display); //points to display function

glutReshapeFunc(reshape);

glutMainLoop(); //go into perpetual loop

return 0;

}