**CSE 4200 Lab 9 – Spencer Wallace**

**Summary:** All parts completed successfully. For problem one I was able to modify Dr. Yu’s program to print the different matrices after applying transformations and rotations. For problem two I was able to draw the wire cube and view it using different values for the look at function. I was also able to complete the extra credit using an idle function which would transform and rotate the earth and sun. Because of this I am giving myself a full score as well as the extra credit, 40/20.

**Problem 1**

**Output**

A black screen with white text

Description automatically generated with low confidence

**Code**

#include <GL/glut.h>

#include <stdlib.h>

#include <iostream>

#include <stdio.h>

using namespace std;

void init(void)

{

glClearColor(1.0, 1.0, 1.0, 0.0);

glShadeModel(GL\_FLAT);

}

//print the transformation matrix

template<class T>

void print\_mat(T m[][4])

{

cout.precision(4);

cout << fixed;

for (int i = 0; i < 4; ++i) {

cout << "\t";

for (int j = 0; j < 4; ++j)

cout << m[j][i] << "\t";

cout << endl;

}

cout << endl;

}

void display(void)

{

float p[4][4];

double pd[4][4];

double pdc[4][4];

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 0.0); //black color

glLoadIdentity(); // clear the matrix

glGetDoublev(GL\_MODELVIEW\_MATRIX, &pdc[0][0]);

cout << "Matrix:" << endl;

print\_mat(pdc);

glLoadIdentity();

glTranslatef(2.0, 0, 0); //move 2 unit on x axis

glGetFloatv(GL\_MODELVIEW\_MATRIX, &p[0][0]);

cout << "Translation | Cube 1" << endl;

print\_mat(p);

glLoadIdentity();

glRotatef(30, 0, 0, 1); // rotate 30 degrees from z axis

glGetDoublev(GL\_MODELVIEW\_MATRIX, &pd[0][0]);

cout << "Rotation | Cube 1" << endl;

print\_mat(pd);

glLoadIdentity();

glRotatef(30, 0, 0, 1); // rotate 30 degrees from z axis

glTranslatef(2.0, 0, 0); //move 2 unit on x axis

glGetDoublev(GL\_MODELVIEW\_MATRIX, &pd[0][0]);

cout << "Composite | Cube 1" << endl;

print\_mat(pd);

glLoadIdentity();

glRotatef(30, 0, 0, 1); // rotate 30 degrees from z axis

glGetDoublev(GL\_MODELVIEW\_MATRIX, &pd[0][0]);

cout << "Rotation | Cube 2" << endl;

print\_mat(p);

glLoadIdentity();

glTranslatef(2.0, 0, 0); //move 2 unit on x axis

glGetFloatv(GL\_MODELVIEW\_MATRIX, &p[0][0]);

cout << "Translation | Cube 2" << endl;

print\_mat(pd);

glLoadIdentity();

glTranslatef(2.0, 0, 0); //move 2 unit on x axis

glRotatef(30, 0, 0, 1); // rotate 30 degrees from z axis

glGetDoublev(GL\_MODELVIEW\_MATRIX, &pd[0][0]);

cout << "Composite | Cube 2" << endl;

print\_mat(pd);

glFlush();

}

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

{

switch (key) {

case 27:

exit(0);

break;

}

}

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);

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

}

**Problem 2**

**Output**

Graphical user interface, application, Word

Description automatically generatedGraphical user interface, application

Description automatically generated

Engineering drawing

Description automatically generated with medium confidenceA picture containing diagram

Description automatically generated

A picture containing application

Description automatically generated

**Code**

#include <GL/glut.h>

#include <stdlib.h>

#include <iostream>

#include <stdio.h>

using namespace std;

int lookVersion = 0;

void init(void)

{

glClearColor(1.0, 1.0, 1.0, 0.0);

glShadeModel(GL\_FLAT);

}

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);

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 0.0); //black color

glLoadIdentity(); // clear the matrix

switch(lookVersion)

{

case 0: gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0); //see nothing

break;

case 1: gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); //see cube

break;

case 2: gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0); //see cube

break;

case 3: gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); //see cube

break;

case 4: gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 1.0, 0.5, 0.0); //see cube

break;

default: cout << "Invalid value for cube to display."; return;

}

glutWireCube(1.0);

glFlush();

}

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

{

switch (key) {

case 27:

exit(0);

break;

}

}

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

{

if(argc > 1){

lookVersion = int(\*argv[1])-48;

cout << argv[1] << endl;

}

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow(argv[0]);

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

}

**Extra Credit**

**Output**

Graphical user interface

Description automatically generatedA picture containing graphical user interface

Description automatically generated

**Code**

#include <GL/glut.h>

#include <stdlib.h>

#include <iostream>

#include <stdio.h>

void init()

{

glShadeModel(GL\_FLAT);

glClearColor(0.0, 0.08, 0.15, 0.0); //bluish black

}

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, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

float earth = 1.0;

float sun = 1.0;

void display(void)

{

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(0.0, 0.0, -100.0);

glRotatef(sun, 0.0, 0.0, 1.0);

glColor3f(1.0, 0.6, 0.0);

glutWireSphere(10.0, 20, 20);

glColor3f(0.0, 1.0, 0.0); //green earth wireframe

glRotatef(earth, 0.0, 0.0, 1.0);

glTranslatef(40.0, 0.0, 0.0);

glutWireSphere(5.0, 12, 12);

glColor3f(0.0, 0.0, 1.0); //blue earth wireframe

glRotatef(earth, 0.0, 0.0, 0.9);

glutWireSphere(5.0, 12, 12);

glFlush();

glutSwapBuffers();

}

void spin(void)

{

earth += 2.0;

sun += 0.5;

display();

}

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

{

switch (key) {

case 27:

exit(0);

break;

}

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB );

glutInitWindowPosition(100, 100);

glutInitWindowSize(500, 500);

glutCreateWindow(argv[0]);

glutReshapeFunc(reshape);

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

glutIdleFunc(spin);

init();

glutMainLoop();

return 0;

}