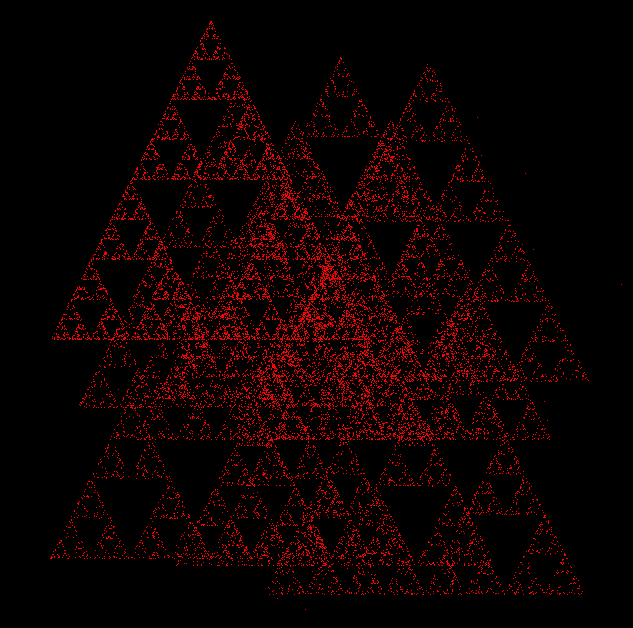
**Assignment 2B**

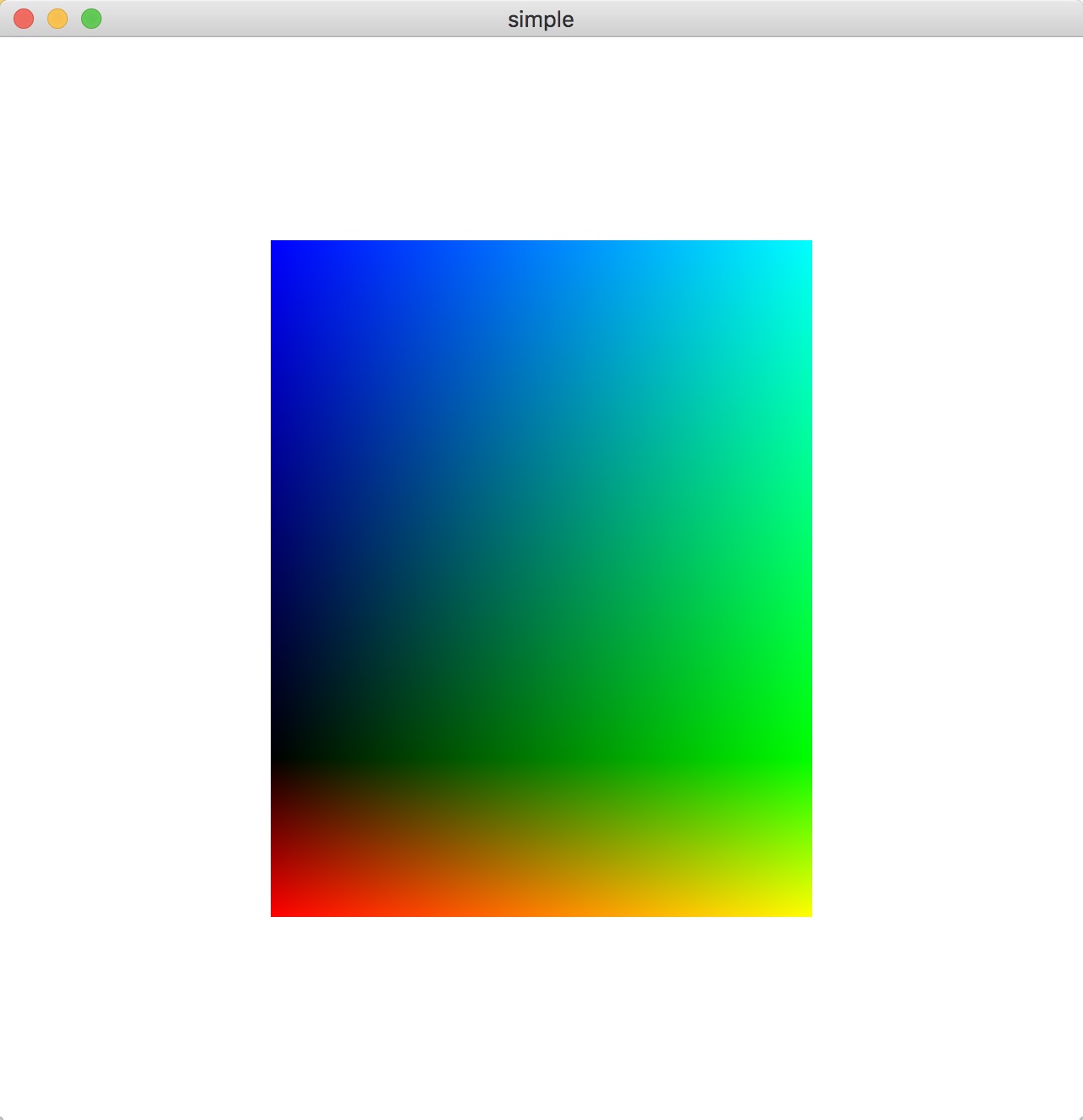
Student Name: Han Song

Student pawprint: hs267

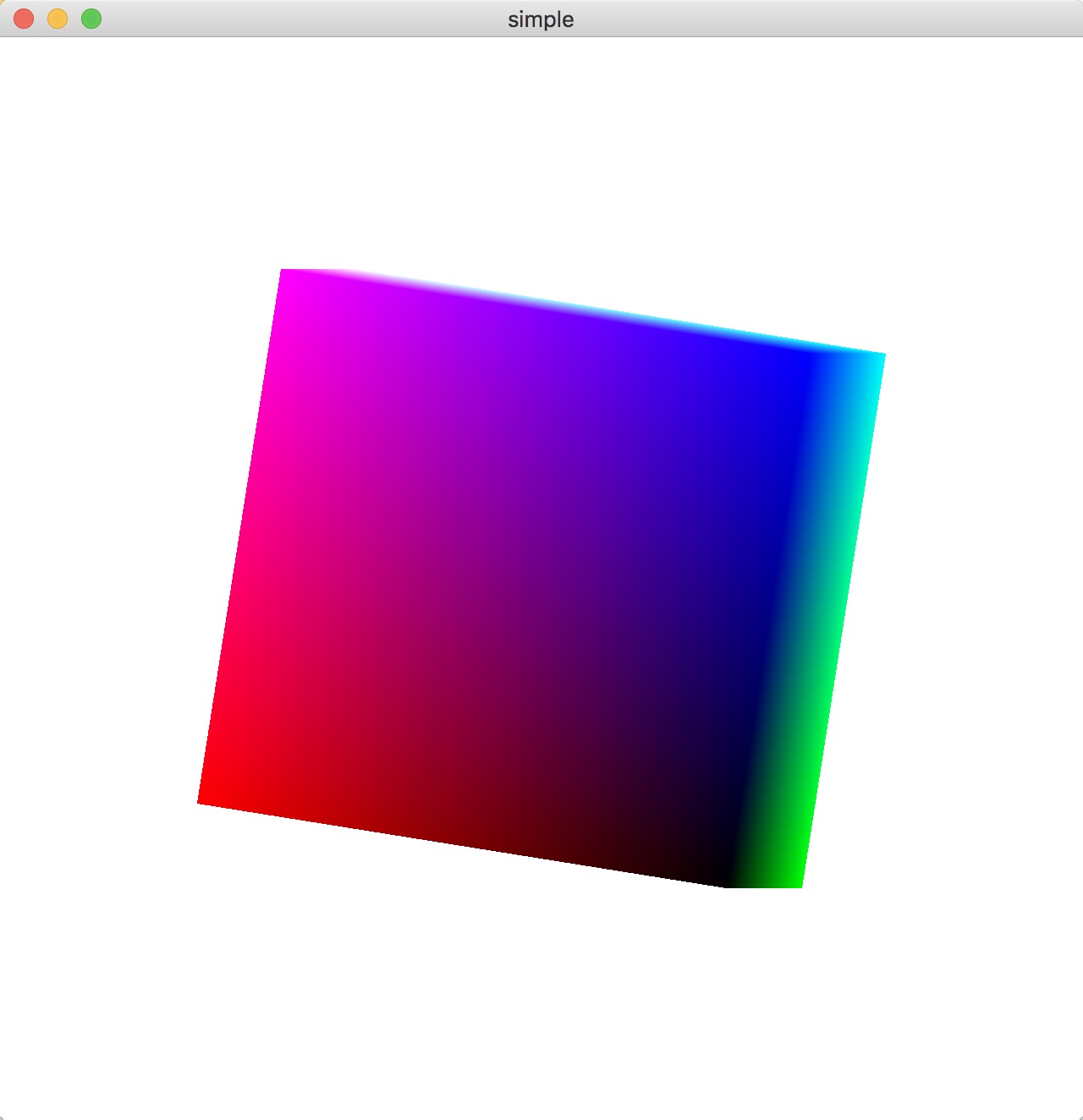
Part a:

Every click creates a red triangle.

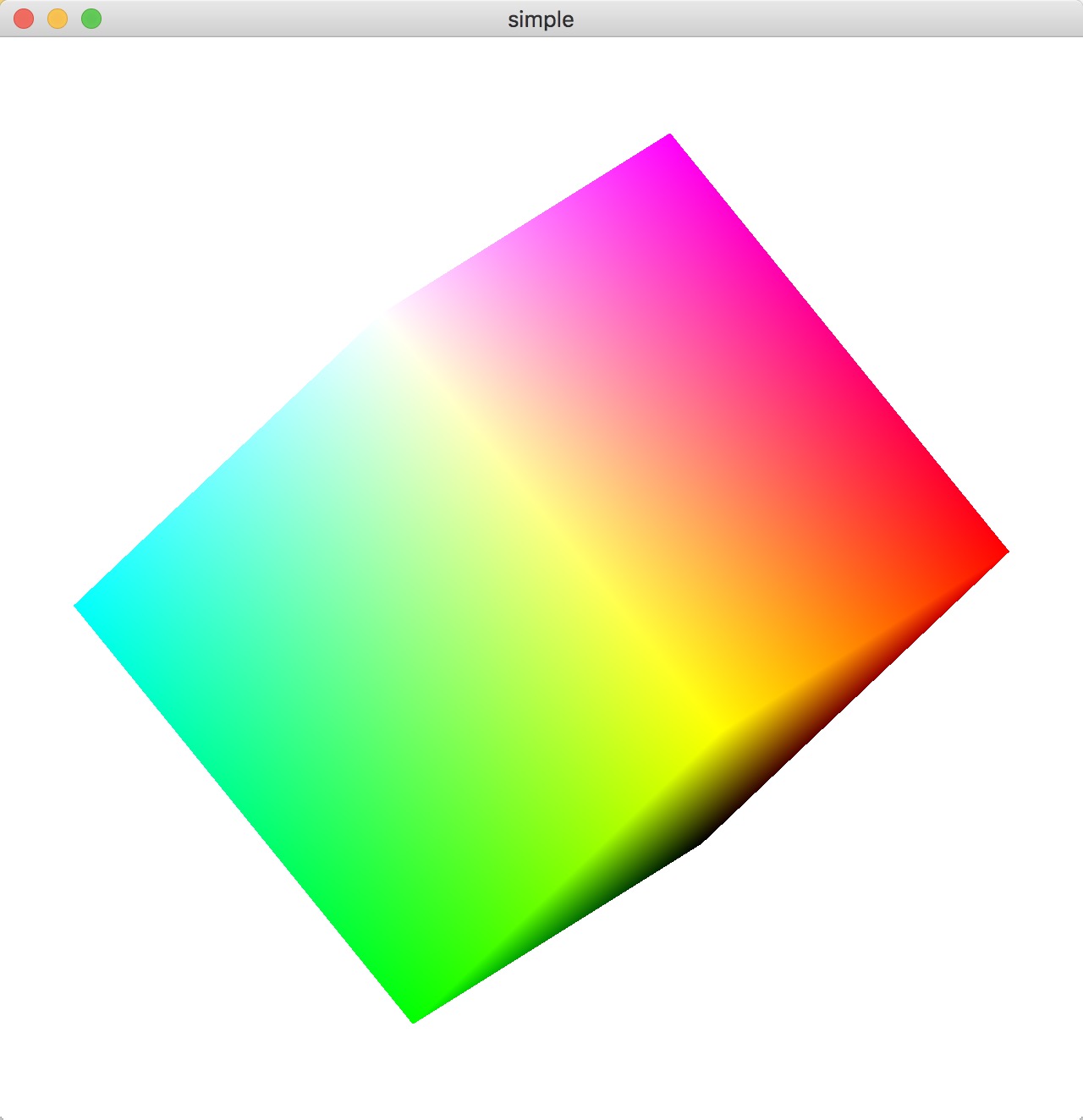
Part b:

Press “x”, and cube will rotate in the x axis.

Press “y”, and cube will rotate in the y axis.



Press “z”, and cube will rotate in the z axis.

 Press “s”, and cube will stop.

//

// main.cpp

// rotat

//

// Created by Han Song on 3/13/18.

// Copyright © 2018 Han Song. All rights reserved.

//

// RotateCube.cpp

// Generate a rotating cube on a solid background

#include "GL/glew.h"

#include <stdio.h>

#include <stdlib.h>

#include "GLFW/glfw3.h"

#include "Angel.h"

typedef vec4 point4;

typedef vec4 color4;

GLuint vao;

int Index = 0;

const int NumVertices = 36;

int pause=0;

point4 points[NumVertices];

color4 colors[NumVertices];

//----------------------------------------------------------------------------

point4 vertices[8] = {

point4(-0.5, -0.5, 0.5, 1.0),

point4(-0.5, 0.5, 0.5, 1.0),

point4(0.5, 0.5, 0.5, 1.0),

point4(0.5, -0.5, 0.5, 1.0),

point4(-0.5, -0.5, -0.5, 1.0),

point4(-0.5, 0.5, -0.5, 1.0),

point4(0.5, 0.5, -0.5, 1.0),

point4(0.5, -0.5, -0.5, 1.0)

};

color4 vertex\_colors[8] = {

color4(0.0, 0.0, 0.0, 1.0), // black

color4(1.0, 0.0, 0.0, 1.0), // red

color4(1.0, 1.0, 0.0, 1.0), // yellow

color4(0.0, 1.0, 0.0, 1.0), // green

color4(0.0, 0.0, 1.0, 1.0), // blue

color4(1.0, 0.0, 1.0, 1.0), // magenta

color4(1.0, 1.0, 1.0, 1.0), // white

color4(0.0, 1.0, 1.0, 1.0) // cyan

};

enum { Xaxis = 0, Yaxis = 1, Zaxis = 2, NumAxes = 3 };

int Axis = Xaxis;

GLfloat Theta[NumAxes] = { 0.0, 0.0, 0.0 };

GLuint theta; // The location of the "theta" shader uniform variable

void

quad(int a, int b, int c, int d)

{

colors[Index] = vertex\_colors[a]; points[Index] = vertices[a]; Index++;

colors[Index] = vertex\_colors[b]; points[Index] = vertices[b]; Index++;

colors[Index] = vertex\_colors[c]; points[Index] = vertices[c]; Index++;

colors[Index] = vertex\_colors[a]; points[Index] = vertices[a]; Index++;

colors[Index] = vertex\_colors[c]; points[Index] = vertices[c]; Index++;

colors[Index] = vertex\_colors[d]; points[Index] = vertices[d]; Index++;

}

void

colorcube()

{

quad(1, 0, 3, 2);

quad(2, 3, 7, 6);

quad(3, 0, 4, 7);

quad(6, 5, 1, 2);

quad(4, 5, 6, 7);

quad(5, 4, 0, 1);

}

void init(void)

{

colorcube();

// Create a vertex array object

GLuint vao;

glGenVertexArrays(1, &vao);

glBindVertexArray(vao);

// Create and initialize a buffer object

GLuint buffer;

glGenBuffers(1, &buffer);

glBindBuffer(GL\_ARRAY\_BUFFER, buffer);

glBufferData(GL\_ARRAY\_BUFFER, sizeof(points)+sizeof(colors),

NULL, GL\_STATIC\_DRAW);

glBufferSubData(GL\_ARRAY\_BUFFER, 0, sizeof(points), points);

glBufferSubData(GL\_ARRAY\_BUFFER, sizeof(points), sizeof(colors), colors);

// Load shaders and use the resulting shader program

GLuint program = InitShader("/Users/thehanemperor/Desktop/SCHOOL/4610SP2018/assignment2b/rotat/rotat/vshader36.glsl", "/Users/thehanemperor/Desktop/SCHOOL/4610SP2018/assignment2b/rotat/rotat/fshader36.glsl");

glUseProgram(program);

// set up vertex arrays

GLuint vPosition = glGetAttribLocation(program, "vPosition");

glEnableVertexAttribArray(vPosition);

glVertexAttribPointer(vPosition, 4, GL\_FLOAT, GL\_FALSE, 0,

BUFFER\_OFFSET(0));

GLuint vColor = glGetAttribLocation(program, "vColor");

glEnableVertexAttribArray(vColor);

glVertexAttribPointer(vColor, 4, GL\_FLOAT, GL\_FALSE, 0,

BUFFER\_OFFSET(sizeof(points)));

theta = glGetUniformLocation(program, "theta");

glEnable(GL\_DEPTH\_TEST);

glClearColor(1.0, 1.0, 1.0, 1.0);

}

void mymouse(GLFWwindow\* window, int button, int action, int mods)

{

if (action == GLFW\_PRESS) {

if (button == GLFW\_MOUSE\_BUTTON\_LEFT) {

Axis = Xaxis;

}

if (button == GLFW\_MOUSE\_BUTTON\_MIDDLE) {

Axis = Yaxis;

}

if (button == GLFW\_MOUSE\_BUTTON\_RIGHT) {

Axis = Zaxis;

}

}

}

void mykey(GLFWwindow\* window, int key, int scancode, int action, int mods)

{

if (action == GLFW\_PRESS) {

if (key == GLFW\_KEY\_ESCAPE || key == GLFW\_KEY\_Q) {

glfwSetWindowShouldClose(window, GLFW\_TRUE);

}

// Press the key 'x' to start rotate in the x axis.

if (key == GLFW\_KEY\_X) {

Axis = Xaxis;

}

//Press the key 'y' to start rotate in the y axis.

if (key == GLFW\_KEY\_Y) {

Axis = Yaxis;

}

//Press the key 'z' to start rotate in the z axis.

if (key == GLFW\_KEY\_Z) {

Axis = Zaxis;

}

//Press the key 's' to stop the rotation.

if (key == GLFW\_KEY\_S) {

if (pause == 0) {

pause = 1;

}

else if (pause == 1) {

pause = 0;

}

}

}

}

//----------------------------------------------------------------------------

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

{

GLFWwindow \*window = NULL;

const GLubyte \*renderer;

const GLubyte \*version;

/\* start GL context and O/S window using the GLFW helper library \*/

if (!glfwInit()) {

fprintf(stderr, "ERROR: could not start GLFW3\n");

return 1;

}

/\* We must specify 3.2 core if on Apple OS X -- other O/S can specify

anything here. I defined 'APPLE' in the makefile for OS X

Remove the #ifdef #endif and play around with this - you should be starting

an explicit version anyway, and some non-Apple drivers will require this too.

\*/

glfwWindowHint(GLFW\_CONTEXT\_VERSION\_MAJOR, 3);

glfwWindowHint(GLFW\_CONTEXT\_VERSION\_MINOR, 2);

glfwWindowHint(GLFW\_OPENGL\_FORWARD\_COMPAT, GL\_TRUE);

glfwWindowHint(GLFW\_OPENGL\_PROFILE, GLFW\_OPENGL\_CORE\_PROFILE);

window = glfwCreateWindow(640, 640, "simple", NULL, NULL);

if (!window) {

fprintf(stderr, "ERROR: could not open window with GLFW3\n");

glfwTerminate();

return 1;

}

glfwMakeContextCurrent(window);

/\* start GLEW extension handler \*/

glewExperimental = GL\_TRUE;

glewInit();

renderer = glGetString(GL\_RENDERER); /\* get renderer string \*/

version = glGetString(GL\_VERSION); /\* version as a string \*/

printf("Renderer: %s\n", renderer);

printf("OpenGL version supported %s\n", version);

init();

glfwSetKeyCallback(window, mykey);

glfwSetMouseButtonCallback(window, mymouse);

//view = glGetUniformLocation(shader\_programme, "view");

do{

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT); // clear the window

if (pause == 0) {

Theta[Axis] += 1.0;

if (Theta[Axis] > 360.0) {

Theta[Axis] -= 360.0;

}

}

glUniform3fv(theta, 1, Theta);

glDrawArrays(GL\_TRIANGLES, 0, NumVertices); // draw the points

/\* update other events like input handling \*/

glfwPollEvents();

glfwSwapBuffers(window);

} while (!glfwWindowShouldClose(window));

// Close OpenGL window and terminate GLFW

glfwTerminate();

return 0;

}