UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SECV2213-01

FUNDAMENTAL OF COMPUTER GRAPHICS

FINAL PROJECT REPORT

LECTURER: DR. GOH EG SU

NAME	MATRIC NUMBER
ONG YIN REN	A19EC0204
NG JING ER	A19EC0115

PROJECT REPORT

1. INTRODUCTION

This project aims to implement 3D object modeling, hierarchical modeling, proper camera, lighting and shading models using C/C++, OpenGL and GLUT based on this theme which is "**Cartoon Moving Character**". Hence, in our project, we have created a cartoon model which has SpongeBob as our main character.

In order to better understand this report, we will be divided into six parts which are introduction, overall concept and design, implementation and achievement, discussion and conclusion. In part 2 overall concept and design, we will discuss the function of the keyboard button to control the movement of the output. While in part3 implementation and achievements, we will be discussing how we use this technique 3D modelling, hierarchical modelling and 3D transformation to create the SpongeBob component and using lighting. Shading, projection and camera model techniques to make the output more attractive.

Furthermore, in part 4 discussion, we will be discussing more details about the function of OpenGL that we implement in this project. In conclusion part will be the conclusion and appreciation of our report. At the end of this report, we have provided the video link that displays the entire project output.

2. OVERALL CONCEPT/DESIGN

For our model component, it is made up of different parts which are body, eyes, bottom, hands and face. The body and legs are in cube shape and cuboid shape respectively. Hence, we use glutSolidCube to create the leg object and GL_POLYGON to create the body object which required glVertex3f to set the position. The transformation is implemented using openGL functions such as glRotatef and glTranslatef to perform rotation of angle degrees around the vector x,y,z which is adjusted according to our output requirement.

Light and bright colours are chosen as the color for our model design. The color effect is adjusted by the glColor3f function. The reason for choosing these kinds of colors is because it will display a significant change on our 3D model of SpongBob which involves the lighting and shadow techniques that we had applied. Besides using lightning and

shadowing techniques, we also applied the sound effect into our project to result in a more attractive output. In order to create a more realistic 3D environment, we had applied the projection and camera model technique into our project. This technique is used to compute the 3D scene.

In order to make interaction when displaying the output, we had applied the keyboard function to design the movement of our project. For example, users can press the button of the alphabet " w" to adjust the movement of the left hand of the character , "o" to enable the music on , "n" to enable the lightning and GLUT_Key_Up to enlarge the scene. Besides using the keyboard function, we also design using a mouse for the user to move the scene freely left and right by applying the glutMouseFunc.

Keyboard Button	Function
a	**Move the left palm of the character in an anticlockwise direction.**
d	**Move the left palm of the character in a clockwise direction.**
w	**Move up and down the left hand of the character**
A	**Move the right palm of the character in an anticlockwise direction.**
D	**Move the right palm of the character in a clockwise direction.**
W	**Move up and down the right hand of the character**
O & o	**Enable for music on**
N & n	**Enable for music off**
l	**Enable for lightning**
m	**Disable for lightning**

Keyboard Button	Function
glut_Key_Up	**Enlarge the output**
glut_Key_Down	**Smaller the output**
glut_Key_Left	**Move up the camera view**
glut_Key_Right	**Move down the camera view**
mouseMove (left)	**Move the camera view to left**
mouseMove (right)	**Move the camera view to right**

3. IMPLEMENTATION & ACHIEVEMENTS

To create an attractive output of the project, we applied different GLUT functions and techniques that we had learnt in previous exercises. We created the cartoon model of SpongBob using the technique 3D modelling, hierarchical modelling and 3D transformation. The cartoon model is created with different shapes which are formed using some simple primitive types such as lines to obtain the nose and mouth of the Sponge Bob; polygon and 3D primitives to draw the body, hands and legs of the Sponge Bob.

The basic transformations such as rotate, translate, and scale are implemented to transform the coordinate position of the objects. The lighting and shading techniques are implemented in this project. These techniques successfully make the output of the 3D model become more realistic and attractive. The projection is applied in this project which enables the user to see a different perspective of view by changing the angle of the camera (screen) using the keyboard button (left,right,up,down) or using a mouse to scroll the screen. The closing theme song of Spong Bob is also added into our project.

4. DISCUSSION

In this project, we have implemented some of the techniques and functions that learnt in the previous lecture topics. We have improved the 3D object modelling, hierarchical modelling and 3D transformation to create a 3D model of cartoon with the model created in Assignment 3 by adding other features.

A. 3D MODELLING

For the 3D modelling in this project, we continue the model created in previous Assignment 3. The model created is a hierarchical model as it contains 1 main character (body part of SpongeBob) and few sub characters (bottom, left hand, right hand, and legs. Some of the sub parts are created using combinations of 3D primitives provided by GLUT such as glutSolidSphere, glutSolidCube. While we are also drawing some of the sub parts with GL_POLYGON which involve a sequence of vertices since we hope to obtain a more attractive model with colour adjustment. The transformation is implemented using openGL functions such as glRotatef and glTranslatef to perform rotation of angle degrees around the vector x,y,z which is adjusted according to our output requirement.

B. LIGHTING AND SHADING

In this project,functions such as GL_AMBIENT for ambient colours. The ambient light intensity is adjusted with the ambient surface color in the function. and GL_DIFFUSE for diffuse colours, where diffuse light intensity is adjusted. GLfloat lightColor0 and lightColor1 are the variables of GL_POSITION which are required to adjust the light position. have been used in this lab exercise to control the lighting effects of the 3D object created.

With the effect of lighting, the user is able to view different effects of the output displayed when they press different input from the keyboard. The shading mode implemented in this project is Flat Shading. The glShadeModel(GL_FLAT) function is used for shading effect where flat shading selects the computed color as we entered in the function assigning to the pixel fragments generated. The effect will be displayed if the lighting is enabled and the

colour back to original colour when disabled. The output displayed becomes more funny, attractive and realistic with the implementation of lighting and shading.

C. PROJECTION AND CAMERA MODEL

For the projection, there are some variables that are required to be declared and needed for the projection effect to be displayed. It includes the angle of the rotation in the y-axis which allows the user to rotate the camera (angle), vector representing the camera position in X-Z plane (lx,lz) and the position of the camera in X-Z plane (x,z). glLoadIdentity is used to reset the transformations and the viewing transformation of the camera is defined using the gluLookAt function. The interactivity implemented in this project enables the users to see a different view of perspective according to the changes of camera angle when the users press the left key or right ke from the keyboard. The new values of x and z is changing according to the formula x = x + lx * fraction and z = z + lz * fraction where fraction is initialised as 0.

5. CONCLUSION

In conclusion, the purpose of this report is to explain how we use the OpenGL function that we learned from the class session in order to produce a lifewell component model and the attractive output scene. Throughout the project, we knew that all the function of OpenGL can liked together in one coding such as using 3D object modelling, hierarchical modelling and 3D transformation to create our main character, using lightning and shading to switch our output for the lightning or non-lightning and lastly using projection and camera model to setup and view the surrounding of our output. Besides, we need to apply the keyboard function to manipulate our output movement such as moving up and down the hand of our main character.

Throughout the process of completing this project, there are a few challenges we have met. Since there are a lot of new functions that need to be applied for the coding part, the output of the coding does not meet our expectations. It was a time-consuming task to find the problems and errors of the coding and we had to find discussions about the solution for the errors that occured, and we had to complete the project on time to avoid late submission.

Last but not least, we would like to thank our lecturer , Dr. Goh Eg Su who provides us with the material of the different functions of OpenGL that make us know which function is suitable and how to implement this project.

6. VIDEO LINK

Click to view the output video

7. Appendix (Code)

```
#include <GL/glut.h>
#include <stdlib.h>
#include <math.h>
#include <windows.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <fstream>

static int lshoulderAngle = 0, lpalm = 0, rshoulderAngle = 0, rpalm = 0;
static int window;
float G_right_upper_leg = 0.0;
float G_left_upper_leg = 0.0;
float G_left_lower_leg = 0.0f;
float G_right_lower_leg = 0.0f;
float x = 0.0f,y=0.0f, z = 5.0f;
float angleX = 0.0f;
float angleY = 0.0f;
float lx = 0.0f,ly = 0.0f,lz = -1.0f;
```

```
float deltaAngleX = 0.0f;

float deltaAngleY = 0.0f;

int deltaMoveX = 0;

int deltaMoveY = 0;

int xOrigin = -1;

int yOrigin = -1;

int width = 1200;

int height = 700;

#define PI 3.14159265f;



void displayBody() {

        glPushMatrix();

        glRotatef(-20, 0.0, 1.0, 0.0);


        //top

        glBegin(GL_POLYGON);

        glColor3f(1.0, 1.0, 0.0);

        glVertex3f(-0.5, 1.5, 0.5);

        glVertex3f(0.5, 1.5, 0.5);

        glVertex3f(0.5, 1.5, -0.5);
```

```
glVertex3f(-0.5, 1.5, -0.5);
glEnd();

// back
glBegin(GL_POLYGON);
glColor3f(1.0, 1.0, 0.0);
glVertex3f(-0.5, 0.4, -0.5);
glVertex3f(-0.5, 1.5, -0.5);
glVertex3f(0.5, 1.5, -0.5);
glVertex3f(0.5, 0.4, -0.5);
glEnd();

// right
glBegin(GL_POLYGON);
glColor3f(0.9, 0.9, 0.0);
glVertex3f(0.5, 1.5, 0.5);
glVertex3f(0.5, 0.4, 0.5);
glVertex3f(0.5, 0.4, -0.5);
glVertex3f(0.5, 1.5, -0.5);
glEnd();
```

```
        // left
        glBegin(GL_POLYGON);
        glColor3f(0.9, 0.9, 0.0);
        glVertex3f(-0.5, 1.5, 0.5);
        glVertex3f(-0.5, 0.4, 0.5);
        glVertex3f(-0.5, 0.4, -0.5);
        glVertex3f(-0.5, 1.5, -0.5);
        glEnd();

        // front
        glBegin(GL_POLYGON);
        glColor3f(1.0, 1.0, 0.0);
        glVertex3f(-0.5, 0.4, 0.5);
        glVertex3f(-0.5, 1.5, 0.5);
        glVertex3f(0.5, 1.5, 0.5);
        glVertex3f(0.5, 0.4, 0.5);
        glEnd();
}

void displayHand(GLdouble width, GLdouble height, GLdouble depth) {
        glPushMatrix();
```

```
        glScalef(width, height, depth);
        glutSolidCube(1.0);
        glPopMatrix();
}

void displayLeftHand() {

        //lefthand
        glPushMatrix();
        glColor3f(1.0, 1.0, 0.0);
        glTranslatef(-0.8, 0.8, 0.0);
        glRotatef(20.0, 0.0, 0.0, 1.0);
        glRotatef((GLfloat)lshoulderAngle, 0.0, 0.0, 1.0);

        displayHand(0.6, 0.1, 0.1);

        glTranslatef(-0.2, 0.0, 0.0);
        glRotatef((GLfloat)lpalm, 0.0, 0.0, 1.0);
        glTranslatef(-0.2, 0.0, 0.0);
        glutSolidSphere(0.1, 50, 50);
        glPopMatrix();
```

```
}

void displayRightHand() {

	//rightHand
	glColor3f(1.0, 1.0, 0.0);
	glTranslatef(0.8, 0.8, 0.0);
	glRotatef(-20.0, 0.0, 0.0, 1.0);
	glRotatef((GLfloat)rshoulderAngle, 0.0, 0.0, 1.0);
	displayHand(0.6, 0.1, 0.1);

	glTranslatef(0.2, 0.0, 0.0);
	glRotatef((GLfloat)rpalm, 0.0, 0.0, 1.0);
	glTranslatef(0.2, 0.0, 0.0);
	glutSolidSphere(0.1, 50, 50);
	glPopMatrix();

}

void displayBottom() {
```

```
glPushMatrix();

glRotatef(-20, 0.0, 1.0, 0.0);


// back

glBegin(GL_POLYGON);

glColor3f(0.6, 0.4, 0.2);

glVertex3f(-0.5, -0.5 + 0.8, -0.5);

glVertex3f(-0.5, -1.1 + 0.8, -0.5);

glVertex3f(0.5, -1.1 + 0.8, -0.5);

glVertex3f(0.5, -0.5 + 0.8, -0.5);

glEnd();


//white back

glTranslatef(0.0, 0.0, -0.003);

glBegin(GL_POLYGON);

glColor3f(0.9, 0.9, 0.9);

glVertex3f(-0.5, -0.5 + 1, -0.5);

glVertex3f(-0.5, -0.7 + 1, -0.5);

glVertex3f(0.5, -0.7 + 1, -0.5);

glVertex3f(0.5, -0.5 + 1, -0.5);
```

```
glEnd();


// right
glBegin(GL_POLYGON);
glColor3f(0.5, 0.4, 0.2);
glVertex3f(0.5, -0.5 + 0.8, 0.5);
glVertex3f(0.5, -1.1 + 0.8, 0.5);
glVertex3f(0.5, -1.1 + 0.8, -0.5);
glVertex3f(0.5, -0.5 + 0.8, -0.5);
glEnd();


//white right
glTranslatef(0.005, 0.0, 0.0);
glBegin(GL_POLYGON);
glColor3f(0.9, 0.9, 0.9);
glVertex3f(0.5, -0.5 + 1, 0.5);
glVertex3f(0.5, -0.7 + 1, 0.5);
glVertex3f(0.5, -0.7 + 1, -0.5);
glVertex3f(0.5, -0.5 + 1, -0.5);
glEnd();
```

```
// left
glBegin(GL_POLYGON);
glColor3f(0.5, 0.4, 0.2);
glVertex3f(-0.5, -0.5 + 0.8, 0.5);
glVertex3f(-0.5, -1.1 + 0.8, 0.5);
glVertex3f(-0.5, -1.1 + 0.8, -0.5);
glVertex3f(-0.5, -0.5 + 0.8, -0.5);
glEnd();

//white left
glTranslatef(-0.005, 0.0, 0.0);
glBegin(GL_POLYGON);
glColor3f(0.9, 0.9, 0.9);
glVertex3f(-0.5, -0.5 + 1, 0.5);
glVertex3f(-0.5, -0.7 + 1, 0.5);
glVertex3f(-0.5, -0.7 + 1, -0.5);
glVertex3f(-0.5, -0.5 + 1, -0.5);
glEnd();

// front
glBegin(GL_POLYGON);
```

```
glColor3f(0.6, 0.4, 0.2);
glVertex3f(-0.5, -0.5 + 0.8, 0.5);
glVertex3f(-0.5, -1.1 + 0.8, 0.5);
glVertex3f(0.5, -1.1 + 0.8, 0.5);
glVertex3f(0.5, -0.5 + 0.8, 0.5);
glEnd();

//white front
glTranslatef(0.0, 0.0, 0.005);
glBegin(GL_POLYGON);
glColor3f(1.0, 1.0, 1.0);
glVertex3f(-0.5, -0.5 + 1, 0.5);
glVertex3f(-0.5, -0.7 + 1, 0.5);
glVertex3f(0.5, -0.7 + 1, 0.5);
glVertex3f(0.5, -0.5 + 1, 0.5);
glEnd();

//top
glBegin(GL_POLYGON);
glColor3f(1.0, 1.0, 0.0);
glVertex3f(-0.5, -0.5 + 1, 0.5);
```

```
        glVertex3f(0.5, -0.5 + 1, 0.5);
        glVertex3f(0.5, -0.5 + 1, -0.5);
        glVertex3f(-0.5, -0.5 + 1, -0.5);
        glEnd();

        glPopMatrix();

}

void displayFace() {
        glPushMatrix();
        //right eye
        glRotatef(-20, 0.0, 1.0, 0.15);
        glTranslatef(0.35, 1.1, 0.4);
        glColor3f(1.0, 1.0, 1.0);
        glutSolidSphere(0.15, 20, 20);

        glTranslatef(0.07, -0.01, 0.11);
        glColor3f(0.0, 0.4, 1.0);
        glutSolidSphere(0.08, 20, 20);
```

```
glTranslatef(0.03, -0.005, 0.06);

glColor3f(0.0, 0.0, 0.0);

glutSolidSphere(0.03, 20, 20);



//left eye

glRotatef(-20, 0.0, 1.0, 0.0);

glTranslatef(-0.5, 0.0, 0.15);

glColor3f(1.0, 1.0, 1.0);

glutSolidSphere(0.15, 20, 20);


glTranslatef(0.1, -0.01, 0.07);

glColor3f(0.0, 0.4, 1.0);

glutSolidSphere(0.08, 20, 20);


glTranslatef(0.07, -0.01, 0.03);

glColor3f(0.0, 0.0, 0.0);

glutSolidSphere(0.03, 20, 20);


glRotatef(40, 0.0, 1.0, 0.0);

glTranslatef(0.1, -0.2, -0.17); //nose
```

```
glBegin(GL_LINE_STRIP);

glColor3f(0.0, 0.0, 0.0);

GLfloat angle;

for (int i = 0; i <= 200; i++) {

        angle = i * PI;

        angle = angle / 200;

        glVertex2f(cos(angle) * 0.3, sin(-angle) * 0.15);

}

glEnd();



glBegin(GL_LINE_STRIP); //mouth

glColor3f(0.0, 0.0, 0.0);

GLfloat Angle;

for (int i = 0; i <= 200; i++) {

        Angle = i * PI;

        Angle = Angle / 200;

        glVertex2f(cos(Angle) * 0.08, sin(Angle) * 0.08);

}

glEnd();
```

```
        glBegin(GL_POLYGON); //tooth
        glColor3f(1.0, 1.0, 1.0);
        glVertex3f(-0.1, -0.16, 0.0);
        glVertex3f(-0.1, -0.25, 0.0);
        glVertex3f(0.1, -0.25, 0.0);
        glVertex3f(0.1, -0.16, 0.0);
        glEnd();
        glPopMatrix();
}

void displayleg()
{

        glTranslatef(0.0, 0.9, 0.0);
        glRotatef(-20, 0.0, 1.0, 0.0);
        glPushMatrix();

        //right leg
        glColor3f(1.0, 1.0, 0.0);
        glTranslatef(0.3, -1.0, 0.0);
        glRotatef(G_left_upper_leg, 1.0, 0.0, 0.0);
```

```
glTranslatef(0.0, -0.36, 0.0);
glPushMatrix();
glScalef(0.20, 0.95, 0.3);
glutSolidCube(1.0);
glPopMatrix();

//right shoe
glColor3f(0.0f, 0.0f, 0.0f);
glTranslatef(0.0, -0.55, 0.0);
glutSolidSphere(0.13, 20, 20);

glPopMatrix();
//left leg
glPushMatrix();
glColor3f(1.0, 1.0, 0.0);
glTranslatef(-0.30, -1.03, 0.0);
glRotatef(G_right_upper_leg, 1.0, 0.0, 0.0);
glTranslatef(0.0, -0.3, 0.0);
glPushMatrix();
glScalef(0.20, 0.95, 0.3);
glutSolidCube(1.0);
```

```
        glPopMatrix();
        //left shoe
        glColor3f(0.0f, 0.0f, 0.0f);
        glTranslatef(0.0, -0.55, 0.0);
        glutSolidSphere(0.13, 20, 20);

        glPopMatrix();
        glutSwapBuffers();

}

void drawBubble() {

        glColor3f(0.8, 0.8, 1.0);
        glutSolidSphere(0.5,20,20);
}

void displayGround() {

        glBegin(GL_QUADS);
        glColor3f(0.8, 0.8, 0.5);
```

```
        glVertex3f(-100.0f, -20.0f, -100.0f);
        glVertex3f(-100.0f, -20.0f, 100.0f);
        glVertex3f(100.0f, -20.0f, 100.0f);
        glVertex3f(100.0f, -20.0f, -100.0f);
        glEnd();

        for (int i = -1; i < 3; i++)
                for (int j = -5; j < 3; j++) {

                        glPushMatrix();
                        glTranslatef(i * 20.0, i + 1, j * 10.0);
                        drawBubble();
                        glPopMatrix();
                }
}

void keyboard(unsigned char key, int x, int y)
{
        float fraction=0.1f;
        switch (key) {
        case 'a':
```

```
            (lpalm += 30) %= 360;
            glutPostRedisplay();
            break;
    case 'd':
            (lpalm -= 30) %= 360;
            glutPostRedisplay();
            break;
    case 'w':
            (lshoulderAngle -= 15) %= 30;
            glutPostRedisplay();
            break;
    case 'A':
            (rpalm += 30) %= 360;
            glutPostRedisplay();
            break;
    case 'D':
            (rpalm -= 30) %= 360;
            glutPostRedisplay();
            break;
    case 'W':
            (rshoulderAngle += 15) %= 30;
```

```
            glutPostRedisplay();
            break;
case 'o':
case 'O':
            PlaySoundA("C:\\zmisc\\Spongebob.wav", NULL, SND_ASYNC);
            break;
case 'n':
case 'N':PlaySound(NULL, 0, 0);
            break;
case 'l':glEnable(GL_LIGHTING);
            break;
case 'm':glDisable(GL_LIGHTING);
            break;

case 27:
            exit(0);
            break;
default:
            break;

}
```

```
}

void changeSize(int w, int h)

{

	if (h == 0)

		h = 1;


	float ratio = w * 1.0 / h;


	glMatrixMode(GL_PROJECTION);

	glLoadIdentity();

	glViewport(0, 0, w, h);

	gluPerspective(45.0f, ratio, 0.1, 100.0f);

	glMatrixMode(GL_MODELVIEW);

}



void processNormalKeys(unsigned char key, int x, int y) {


	if (key == 27)
```

```
		exit(0);

}


void processSpecialKeys(int key, int xx, int yy)
{

	float fraction = 0.5f;
	float fractiony = 2.0f;

	switch (key) {

	case GLUT_KEY_UP:x += lx * fraction;
		z += lz * fraction;
		break;
	case GLUT_KEY_DOWN:x -= lx * fraction;
		z -= lz * fraction;
		break;
	case GLUT_KEY_LEFT:y += ly * fractiony;
```

```
            break;
        case GLUT_KEY_RIGHT:y -= ly * fractiony;

            break;

        }
}

void releaseKey(int key, int x, int y) {

        switch (key) {
        case GLUT_KEY_UP:
        case GLUT_KEY_DOWN: deltaMoveX = 0;deltaMoveY = 0;
            break;
        }
}

void mouseMove(int x, int y) {
        if (xOrigin >= 0) {
            deltaAngleX = (x - xOrigin) * 0.001f;
            lx = sin(angleX + deltaAngleX);
```

```
            lz = -cos(angleX + deltaAngleX);
        }
        if (yOrigin >= 0) {
            deltaAngleY = (y - yOrigin) * 0.001f;
            ly = sin(angleY + deltaAngleY);

        }
}

void mouseButton(int button, int state, int x, int y) {
        if (button == GLUT_LEFT_BUTTON) {
            if (state == GLUT_UP)
            {
                angleX += deltaAngleX;
                angleY += deltaAngleY;
                yOrigin = 0;
                xOrigin = -1;
            }
            else
            {
                xOrigin = x;
```

```
                    yOrigin = y;
                }
        }

}

void init(void)
{
        glClearColor(0.1, 0.6, 0.8, 0.0);
        glMatrixMode(GLUT_SINGLE | GLUT_RGB);
        glLoadIdentity();
        glOrtho(0, width, 0, height, 0.0, 1.0);
        glShadeModel(GL_FLAT);
        glEnable(GL_DEPTH_TEST);
        glEnable(GL_COLOR_MATERIAL);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glEnable(GL_LIGHT1);
        glEnable(GL_NORMALIZE);

}
```

```
void renderScene(void) {

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        glLoadIdentity();

        gluLookAt(x, y, z,
                x + lx, y+ly, z + lz,
                0.0f, 0.3f, 0.0f);

        displayGround();
        displayBody();
        displayFace();
        displayRightHand();
        displayLeftHand();
        displayBottom();
        displayleg();

        GLfloat ambientColor[] = { 0.2f,0.2f,0.2f,1.0f };
        glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);
```

```
	GLfloat lightColor0[] = { 0.5f,0.5f,0.5f,1.0f };
	GLfloat lightPos0[] = { 4.0f,0.0f,8.0f,1.0f };
	glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
	glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);

	GLfloat lightColor1[] = { 0.5f,0.2f,0.2f,1.0f };

	GLfloat lightPos1[] = { -1.0f,0.5f,0.5f,0.0f };
	glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
	glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);

}

int main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_SINGLE | GLUT_RGBA);
	glutInitWindowSize(width,height);
	glutInitWindowPosition(100, 0);
	window = glutCreateWindow("Spongebob");
```

```
	init();

	glutDisplayFunc(renderScene);
	glutReshapeFunc(changeSize);
	glutIdleFunc(renderScene);

	glutIgnoreKeyRepeat(1);
	glutKeyboardFunc(processNormalKeys);
	glutSpecialFunc(processSpecialKeys);
	glutSpecialUpFunc(releaseKey);
	glutKeyboardFunc(keyboard);
	glutMouseFunc(mouseButton);
	glutMotionFunc(mouseMove);
	glEnable(GL_DEPTH_TEST);

	glutMainLoop();

	return 0;
}
```