#define VIEW\_TURN\_RATE 10

#include<stdio.h>

#include <GL/glut.h>

#include <stdlib.h>

static int shoulder1 = 0, elbow1 = 0, shoulder2 = 0, elbow2 = 0,leg11=0,leg12=0,leg21=0,leg22=0;

static int spin = 0, spin1=0;

static int begin;

static int turn,turn1=0;

// Mouse Function

void

movelight(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

begin = x;

}

}

//Motion Function

void

motion(int x, int y)

{

spin = (spin + (x - begin)) % 360;

begin = x;

glutPostRedisplay();

}

//Initializations

void init(void)

{

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_DEPTH\_TEST);

glClearColor (0.5, 0.2, 0.2, 0.8);

// glShadeModel (GL\_FLAT);

}

// Drilling Machine

void drilling()

{

glPushMatrix();

glScalef (0.3, 0.2, 3.0);

glutSolidCube (1.0);

glScalef (0.2, 0.1, 2.0);

glutSolidCube (1.0);

glScalef (0.2, 0.1, 1.5);

glutSolidCube (1.0);

glPopMatrix();

}

//Body of the robot

void robo()

{

glPushMatrix();

//neck

glPushMatrix();

glTranslatef (0.0, 1.0, 0.0);

glutSolidCube(0.75);

glPopMatrix();

//head

glPushMatrix();

glTranslatef (0.0, 2.0, 0.0);

glutSolidSphere (0.6, 15, 15);

glPushMatrix(); //left eye

glTranslatef(-0.25,-0.08,0.5);

glutSolidSphere(0.1,15,15);

glPopMatrix();

glPushMatrix(); // right eye

glTranslatef(0.25,-0.08,0.5);

glutSolidSphere(0.1,15,15);

glPopMatrix();

glPopMatrix();

//chest

glPushMatrix();

glTranslatef (0.0, 0.0, 0.0);

glutSolidCube (1.5);

glPopMatrix();

//left hand

glPushMatrix();

glTranslatef (-1.0, 0.5, 0.0);

glutSolidSphere (0.3, 15, 15);

glRotatef ((GLfloat) shoulder1, 0.0, 0.0, 1.0);

glRotated(-40,0.0,0.0,1.0);

glTranslatef (-0.6, 0.0, 0.0);

glPushMatrix();

glScalef (0.8, 0.5, 1.0);

glutSolidCube (1.0); //shoulder1

glPopMatrix();

glTranslatef (-0.6, 0.0, 0.0);

glutSolidSphere (0.3, 15, 15);

glRotatef ((GLfloat) elbow1, 1.0, 0.0, 0.0);

glTranslatef (-0.6, 0.0, 0.0);

glPushMatrix();

glScalef (0.8, 0.4, 1.0);

glutSolidCube (1.0); //elbow1

glPopMatrix();

glTranslatef (-0.7, 0.0, 0.0);

glutSolidSphere (0.3, 15, 15);

glTranslatef (-0.4, 0.0, 0.0);

drilling(); //drilling machine

glPopMatrix();

// right hand

glPushMatrix();

glTranslatef (1.0, 0.5, 0.0);

glutSolidSphere (0.3, 15, 15);

glRotatef ((GLfloat) shoulder2, 0.0, 0.0, 1.0);

glRotated(-60,0.0,0.0,1.0);

glTranslatef (0.6, 0.0, 0.0);

glPushMatrix();

glScalef (0.8, 0.5, 1.0);

glutSolidCube (1.0); //shoulder2

glPopMatrix();

glTranslatef (0.6, 0.0, 0.0);

glutSolidSphere (0.3, 15, 15);

glRotatef ((GLfloat) elbow2, 0.0, 0.0, 1.0);

glRotated(-40,0.0,0.0,1.0);

glTranslatef (0.6, 0.0, 0.0);

glPushMatrix();

glScalef (0.8, 0.4, 1.0);

glutSolidCube (1.0); //elbow2

glPopMatrix();

glTranslatef (0.7, 0.0, 0.0);

glutSolidSphere (0.3, 15, 15);

glPopMatrix();

// leg1

glPushMatrix();

glTranslatef (-0.5, -1.0, 0.0);

glutSolidSphere (0.4, 15, 15);

glRotatef ((GLfloat) leg11, 0.0, 0.0, 1.0);

glRotated(80,0.0,0.0,1.0);

glTranslatef (-1.0, 0.0, 0.0);

glPushMatrix();

glScalef (1.0, 0.5, 1.0);

glutSolidCube (1.0); //leg11

glPopMatrix();

glTranslatef (-1.0, 0.0, 0.0);

glutSolidSphere (0.4, 15, 15);

glRotatef ((GLfloat) leg12, 0.0, 0.0, 1.0);

glTranslatef (-1.0, 0.0, 0.0);

glPushMatrix();

glScalef (1.0, 0.5, 1.0);

glutSolidCube (1.0); //leg12

glPopMatrix();

glPushMatrix();

glTranslatef (-0.8, 0.0, 0.0); //foot

glScalef (0.5, 0.5, 1.5);

glutSolidCube (1.0);

glPopMatrix();

glPopMatrix();

// right leg

glPushMatrix();

glTranslatef (0.5, -1.0, 0.0);

glutSolidSphere (0.4, 15, 15);

glRotatef ((GLfloat) leg21, 0.0, 0.0, 1.0);

glRotated(-80,0.0,0.0,1.0);

glTranslatef (1.0, 0.0, 0.0);

glPushMatrix();

glScalef (1.0, 0.5, 1.0);

glutSolidCube (1.0); //leg21

glPopMatrix();

glTranslatef (1.0, 0.0, 0.0);

glutSolidSphere (0.4, 15, 15);

glRotatef ((GLfloat) leg22, 0.0, 0.0, 1.0);

glTranslatef (1.0, 0.0, 0.0);

glPushMatrix();

glScalef (1.0, 0.5, 1.0);

glutSolidCube (1.0); //leg22

glPopMatrix();

glPushMatrix();

glTranslatef (0.80, 0.0, 0.0);

glScalef (0.5, 0.5, 1.5);

glutSolidCube (1.0); //foot

glPopMatrix();

glPopMatrix();

glPopMatrix();

}

/\* start of rotation of robot functions \*/

void

TurnRight(void)

{

turn = (turn - VIEW\_TURN\_RATE) % 360;

}

void

TurnLeft(void)

{

turn = (turn + VIEW\_TURN\_RATE) % 360;

}

void

TurnForwards(void)

{

turn1 = (turn1 - VIEW\_TURN\_RATE) % 360;

}

void

TurnBackwards(void)

{

turn1 = (turn1 + VIEW\_TURN\_RATE) % 360;

}

//Display Function

void display(void)

{

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

GLfloat position[] = {1.0, 1.0, 1.0, 0.0};

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glTranslatef(0.0, 0.0, -5.0);

glPushMatrix();

glScalef(1.0,1.0,1.0);

glPushMatrix();

glColor3f(0.0,1.0,0.0);

glutWireCube(18.0); //wire frame

glPopMatrix();

glPushMatrix(); //robo function

glRotatef((GLfloat) turn, 0.0, 1.0, 0.0);

glRotatef((GLfloat) turn1, 1.0, 0.0, 0.0);

robo();

glPopMatrix();

glPushMatrix(); // Light properties

glRotated((GLdouble) spin, 0.0, 1.0, 0.0);

glRotated(0.0, 1.0, 0.0, 0.0);

glLightfv(GL\_LIGHT0, GL\_POSITION, position);

glTranslated(0.0, 0.0, 1.5);

glDisable(GL\_LIGHTING);

glColor3f(0.0, 1.0, 1.0);

glutWireCube(0.5); // light source

glEnable(GL\_LIGHTING);

glPopMatrix();

glPopMatrix();

glPopMatrix();

glFlush();

glutSwapBuffers();

}

//Reshape function

void reshape (int w, int h)

{

glViewport (0, 0, (GLsizei) w, (GLsizei) h);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity ();

gluPerspective(65.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef (0.0, 0.0, -5.0);

}

// keyboard function

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

{

switch (key) {

case 's':

shoulder1 = (shoulder1 + 5) % 360;

glutPostRedisplay();

break;

case 'S':

shoulder1 = (shoulder1 - 5) % 360;

glutPostRedisplay();

break;

case 'e':

elbow1 = (elbow1 + 5) % 360;

glutPostRedisplay();

break;

case 'E':

elbow1 = (elbow1 - 5) % 360;

glutPostRedisplay();

break;

case 'd':

shoulder2 = (shoulder2 + 5) % 360;

glutPostRedisplay();

break;

case 'D':

shoulder2 = (shoulder2 - 5) % 360;

glutPostRedisplay();

break;

case 'f':

elbow2 = (elbow2 + 5) % 360;

glutPostRedisplay();

break;

case 'F':

elbow2 = (elbow2 - 5) % 360;

glutPostRedisplay();

break;

case 'l':

leg11 = (leg11 + 5) % 360;

glutPostRedisplay();

break;

case 'L':

leg11 = (leg11 - 5) % 360;

glutPostRedisplay();

break;

case 'm':

leg12 = (leg12 + 5) % 360;

glutPostRedisplay();

break;

case 'M':

leg12 = (leg12 - 5) % 360;

glutPostRedisplay();

break;

case 'n':

leg21 = (leg21 + 5) % 360;

glutPostRedisplay();

break;

case 'N':

leg21 = (leg21 - 5) % 360;

glutPostRedisplay();

break;

case 'o':

leg22 = (leg22 + 5) % 360;

glutPostRedisplay();

break;

case 'O':

leg22 = (leg22 - 5) % 360;

glutPostRedisplay();

break;

case 27:

exit(0);

break;

default:

break;

}

}

// Function for using arrow keys for robot rotation

void

special(int key, int x, int y)

{

int i = 0;

switch (key) {

/\* start of view position functions \*/

case GLUT\_KEY\_RIGHT:{

TurnRight();

i++;

}

break;

case GLUT\_KEY\_LEFT:{

TurnLeft();

i++;

}

break;

case GLUT\_KEY\_DOWN:{

TurnForwards();

i++;

}

break;

case GLUT\_KEY\_UP:{

TurnBackwards();

i++;

}

break;

}

if (i)

glutPostRedisplay();

}

// menu selection function

void

menu\_select(int mode)

{

switch (mode) {

case 4:

exit(EXIT\_SUCCESS);

}

}

void

null\_select(int mode)

{

}

//menu function

void glutMenu(void)

{

int glut\_menu[10];

glut\_menu[1] = glutCreateMenu(null\_select);

glutAddMenuEntry("AT THE SHOULDERS : s,S", 0);

glutAddMenuEntry("AT THE ELBOW : e,E", 0);

glut\_menu[2] = glutCreateMenu(null\_select);

glutAddMenuEntry("AT THE SHOULDERS : d,D", 0);

glutAddMenuEntry("AT THE ELBOW : f,F", 0);

glut\_menu[4] = glutCreateMenu(null\_select);

glutAddMenuEntry("AT THE HIP : l,L", 0);

glutAddMenuEntry("AT THE KNEES : m,M", 0);

glut\_menu[5] = glutCreateMenu(null\_select);

glutAddMenuEntry("AT THE HIP : n,N", 0);

glutAddMenuEntry("AT THE KNEES : o,O", 0);

glut\_menu[6] = glutCreateMenu(null\_select);

glutAddMenuEntry("MOVE THE LEFT BUTTON OF THE MOUSE HORIZONTALLY", 0);

glut\_menu[3] = glutCreateMenu(NULL);

glutAddSubMenu("LEG 1", glut\_menu[4]);

glutAddSubMenu("LEG 2", glut\_menu[5]);

glut\_menu[0] = glutCreateMenu(null\_select);

glutAddSubMenu("HAND 1", glut\_menu[1]);

glutAddSubMenu("HAND 2", glut\_menu[2]);

glut\_menu[7] = glutCreateMenu(null\_select);

glutAddMenuEntry("USE ARROW KEYS", 0);

glutCreateMenu(menu\_select);

glutAddMenuEntry("WHAT CAN I DO??? ", 0);

glutAddSubMenu("ROBOT ROTATION", glut\_menu[7]);

glutAddSubMenu("HAND ROTATION", glut\_menu[0]);

glutAddSubMenu("LEG ROTATION", glut\_menu[3]);

glutAddSubMenu("TO MOVE THE CAMERA",glut\_menu[6] );

glutAddMenuEntry("Quit", 4);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

// main function

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

{

glutInit(&argc, argv);

glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize (600, 600);

glutInitWindowPosition (100, 100);

glutCreateWindow (argv[0]);

init ();

glutMouseFunc(movelight);

glutMotionFunc(motion);

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutSpecialFunc(special);

glutMenu();

glutMainLoop();

return 0;

}