**Lab Taks-5**

Submission Guidelines-

|  |
| --- |
| **Question-1**  Create an animation using two box that will move in the opposite direction. |
| **Graph Plot (Picture)-**  **[Not needed]** |
| **Code-**  **#include <windows.h> // for MS Windows**  **#include <GL/glut.h> // GLUT, include glu.h and gl.h**  **float \_move = 0.0f;**  **float \_move1 = 0.0f;**  **void object1()**  **{**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(\_move, 0.0f, 0.0f);**  **glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad**  **glColor3f(1.0f, 0.0f, 0.0f); // Red**  **glVertex2f(0.0f,0.0f);**  **glVertex2f(1.0f,0.0f);**  **glVertex2f(1.0f,1.0f);**  **glVertex2f(0.0f,1.0f); // x, y**  **glEnd();**  **glPopMatrix();**  **}**  **void object2()**  **{**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(\_move1, 0.0f, 0.0f);**  **glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad**  **glColor3f(1.0f, 1.0f, 0.0f); // Red**  **glVertex2f(0.0f,0.0f);**  **glVertex2f(1.0f,0.0f);**  **glVertex2f(1.0f,-1.0f);**  **glVertex2f(0.0f,-1.0f); // x, y**  **glEnd();**  **glPopMatrix();**  **}**  **/\* Handler for window-repaint event. Call back when the window first appears and**  **whenever the window needs to be re-painted. \*/**  **void display() {**  **glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque**  **glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)**  **object1();**  **object2();**  **glutSwapBuffers();**  **glFlush();**  **//glutSwapBuffers(); // Render now**  **}**  **void update(int value) {**  **\_move += .02;**  **if(\_move > 2.5)**  **{**  **\_move = -3.0;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update, 0);**  **}**  **void update1(int value) {**  **\_move1 -= .02;**  **if(\_move1 < -2.5)**  **{**  **\_move1 = 3.0;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update1, 0);**  **}**  **/\* Main function: GLUT runs as a console application starting at main() \*/**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv); // Initialize GLUT**  **glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title**  **glutInitWindowSize(320, 320);**  **gluOrtho2D(-3,3,-3,3); // Set the window's initial width & height**  **glutDisplayFunc(display); // Register display callback handler for window re-paint**  **glutTimerFunc(20, update, 0);**  **glutTimerFunc(20, update1, 0);**  **glutMainLoop(); // Enter the event-processing loop**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-2**  Design a car which will have rotating wheels. |
| **Graph Plot (Picture)-**  **[Not needed]** |
| **Code-#include <GL/glut.h>**  **#include <cmath>**  **float wheelRotationAngle = 0.0;**  **void drawWheel() {**  **glColor3f(0.0, 0.0, 1.0);**  **float wheelRadius = 0.1;**  **int numSegments = 50;**  **glBegin(GL\_POLYGON);**  **for (int j = 0; j < numSegments; j++) {**  **float theta = 2.0\*3.1416\*float(j)/float(numSegments);**  **float x = wheelRadius \* cos(theta);**  **float y = wheelRadius \* sin(theta);**  **glVertex2f(x, y);**  **}**  **glEnd();**  **glColor3f(0.0, 0.0, 0.0);**  **glLineWidth(3.0);**  **glBegin(GL\_LINES);**  **glVertex2f(-0.05, 0);**  **glVertex2f(0.05, 0);**  **glVertex2f(0, -0.05);**  **glVertex2f(0, 0.05);**  **glEnd();**  **}**  **void drawCar() {**  **glColor3f(0.5, 0.0, 0.0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-0.4, -0.2);**  **glVertex2f(0.4, -0.2);**  **glVertex2f(0.3, 0.0);**  **glVertex2f(-0.3, 0.0);**  **glEnd();**  **glColor3f(0.0, 0.5, 1.0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-0.25,0.0);**  **glVertex2f(0.25,0.0);**  **glVertex2f(0.15, 0.15);**  **glVertex2f(-0.15, 0.15);**  **glEnd();**  **for (int i = -1;i<=1;i+= 2) {**  **glPushMatrix();**  **glTranslatef(i\*0.25,-0.2, 0);**  **glRotatef(wheelRotationAngle,0,0,1);**  **drawWheel();**  **glPopMatrix();**  **}**  **}**  **void display() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glLoadIdentity();**  **drawCar();**  **glFlush();**  **}**  **void update(int value) {**  **wheelRotationAngle += 5.0;**  **if (wheelRotationAngle >= 360.0) {**  **wheelRotationAngle -= 360.0;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(100, update, 0);**  **}**  **void reshape(int w, int h) {**  **glViewport(0,0,w,h);**  **glMatrixMode(GL\_PROJECTION);**  **glLoadIdentity();**  **gluOrtho2D(-1,1,-1,1);**  **glMatrixMode(GL\_MODELVIEW);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);**  **glutInitWindowSize(840, 680);**  **glutCreateWindow("Car with rotating wheel");**  **glutDisplayFunc(display);**  **glutReshapeFunc(reshape);**  **glClearColor(0.0,0.0,0.0,0.0);**  **glutTimerFunc(100, update, 0);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-3**  Now move your car of question-2 from left to right in a loop. |
| **Graph Plot (Picture)-**  **[Not needed]** |
| **Code-**  **#include <GL/glut.h>**  **#include <cmath>**  **float wheelRotationAngle = 0.0;**  **float \_move3= 0.0f;**  **void update\_car(int value)**  **{**  **\_move3 += 0.001;**  **if(\_move3 >1)**  **{**  **\_move3 =-0.5;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update\_car, 0);**  **}**  **void drawWheel() {**  **glColor3f(0.0, 0.0, 1.0);**  **float wheelRadius = 0.1;**  **int numSegments = 50;**  **glBegin(GL\_POLYGON);**  **for (int j = 0; j < numSegments; j++) {**  **float theta = 2.0\*3.1416\*float(j)/float(numSegments);**  **float x = wheelRadius \* cos(theta);**  **float y = wheelRadius \* sin(theta);**  **glVertex2f(x, y);**  **}**  **glEnd();**  **glColor3f(0.0, 0.0, 0.0);**  **glLineWidth(3.0);**  **glBegin(GL\_LINES);**  **glVertex2f(-0.05, 0);**  **glVertex2f(0.05, 0);**  **glVertex2f(0, -0.05);**  **glVertex2f(0, 0.05);**  **glEnd();**  **}**  **void drawCar() {**    **glLoadIdentity();**  **glPushMatrix();**  **glTranslatef(0.0f,\_move3, 0.0f);**  **glColor3f(0.5, 0.0, 0.0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-0.4, -0.2);**  **glVertex2f(0.4, -0.2);**  **glVertex2f(0.3, 0.0);**  **glVertex2f(-0.3, 0.0);**  **glEnd();**  **glColor3f(0.0, 0.5, 1.0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-0.25,0.0);**  **glVertex2f(0.25,0.0);**  **glVertex2f(0.15, 0.15);**  **glVertex2f(-0.15, 0.15);**  **glEnd();**  **for (int i = -1;i<=1;i+= 2) {**  **glPushMatrix();**  **glTranslatef(i\*0.25,-0.2, 0);**  **glRotatef(wheelRotationAngle,0,0,1);**  **drawWheel();**  **glPopMatrix();**  **}**  **}**  **void display() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glLoadIdentity();**  **drawCar();**  **glFlush();**  **}**  **void update(int value) {**  **wheelRotationAngle += 5.0;**  **if (wheelRotationAngle >= 360.0) {**  **wheelRotationAngle -= 360.0;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(100, update, 0);**  **}**  **void reshape(int w, int h) {**  **glViewport(0,0,w,h);**  **glMatrixMode(GL\_PROJECTION);**  **glLoadIdentity();**  **gluOrtho2D(-1,1,-1,1);**  **glMatrixMode(GL\_MODELVIEW);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);**  **glutInitWindowSize(840, 680);**  **glutCreateWindow("Car with rotating wheel");**  **glutDisplayFunc(display);**  **glutReshapeFunc(reshape);**  **glClearColor(0.0,0.0,0.0,0.0);**  **glutTimerFunc(100, update, 0);**  **glutMainLoop();**  **gglutTimerFunc(20, update\_car, 0);**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-4**  Design a windmill with rotating blades |
| **Graph Plot (Picture)-**  **[Not needed]** |
| **Code-**  **#include <GL/glut.h>**  **#include <math.h>**  **GLfloat angle = 0.0f;**  **void display() {**  **glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);**  **glLoadIdentity();**  **glClearColor(1,1,1,1);**  **glPushMatrix();**  **glTranslatef(-1.0, -1.0, -6.0);**  **// windmill base**  **glBegin(GL\_QUADS);**  **glColor3f(0.2, 0.2, 0.2);**  **glVertex3f(-0.1, 0.0, 0.1);**  **glVertex3f(0.1, 0.0, 0.1);**  **glVertex3f(0.1, 1.0, 0.1);**  **glVertex3f(-0.1, 1.0, 0.1);**  **glEnd();**  **// windmill pole**  **glBegin(GL\_QUADS);**  **glColor3f(0.6, 0.6, 0.6);**  **glVertex3f(-0.05, 1.0, 0.05);**  **glVertex3f(0.05, 1.0, 0.05);**  **glVertex3f(0.05, 3.0, 0.05);**  **glVertex3f(-0.05, 3.0, 0.05);**  **glEnd();**  **// first windmill blade**  **glPushMatrix();**  **glTranslatef(0.0, 3.0, 0.05);**  **glRotatef(angle, 0.0, 0.0, 1.0);**  **glBegin(GL\_TRIANGLES);**  **glColor3f(0.5, 0.5, 0.5);**  **glVertex3f(0.0, 0.0, 0.0);**  **glVertex3f(0.5, 0.2, 0.0);**  **glVertex3f(0.5, -0.2, 0.0);**  **glEnd();**  **glPopMatrix();**  **//second windmill blade**  **glPushMatrix();**  **glTranslatef(0.0, 3.0, 0.05);**  **glRotatef(angle + 120, 0.0, 0.0, 1.0);**  **glBegin(GL\_TRIANGLES);**  **glColor3f(0.5, 0.5, 0.5);**  **glVertex3f(0.0, 0.0, 0.0);**  **glVertex3f(0.5, 0.2, 0.0);**  **glVertex3f(0.5, -0.2, 0.0);**  **glEnd();**  **glPopMatrix();**  **// third windmill blade**  **glPushMatrix();**  **glTranslatef(0.0, 3.0, 0.05);**  **glRotatef(angle + 240, 0.0, 0.0, 1.0);**  **glBegin(GL\_TRIANGLES);**  **glColor3f(0.5, 0.5, 0.5);**  **glVertex3f(0.0, 0.0, 0.0);**  **glVertex3f(0.5, 0.2, 0.0);**  **glVertex3f(0.5, -0.2, 0.0);**  **glEnd();**  **glPopMatrix();**  **glPopMatrix();**  **glutSwapBuffers();**  **}**  **void reshape(int width, int height) {**  **glViewport(0, 0, (GLsizei)width, (GLsizei)height);**  **glMatrixMode(GL\_PROJECTION);**  **glLoadIdentity();**  **gluPerspective(60, (GLfloat)width / (GLfloat)height, 1.0, 100.0);**  **glMatrixMode(GL\_MODELVIEW);**  **}**  **void timer(int value) {**  **angle += 1.0;**  **if (angle >= 360.0)**  **angle -= 360.0;**  **glutPostRedisplay();**  **glutTimerFunc(16, timer, 0);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);**  **glutInitWindowSize(800, 600);**  **glutCreateWindow("Windmill");**  **glEnable(GL\_DEPTH\_TEST);**  **glutDisplayFunc(display);**  **glutReshapeFunc(reshape);**  **glutTimerFunc(0, timer, 0);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |