**Lab Taks-5**

Submission Guidelines-

* Rename the file with your serial number only
* Must submit within the announced time.
* Must include resources for all the section in the table

|  |
| --- |
| **Question-1**  Create an animation using two box that will move in the opposite direction. |
| **Graph Plot (Picture)-** |
| **Code-**  **#include <iostream>**  **#include<GL/gl.h>**  **#include <GL/glut.h>**  **using namespace std;**  **float \_move1 = -2.0f;**  **float \_move2 = 2.0f;**  **void box\_1()**  **{**  **glColor3d(1,0,0);**  **glLoadIdentity(); //Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(\_move1, 0.0f, 0.0f);**  **glBegin(GL\_QUADS);**  **glVertex2f(0.1f, 0.0f);**  **glVertex2f(0.5f, 0.0f);**  **glVertex2f(0.5f, 0.2f);**  **glVertex2f(0.1f, 0.2);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_top(int value)**  **{**  **\_move1 += .02;**  **if(\_move1 > 1.5)**  **{**  **\_move1 = -1;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(10, update\_top, 0);**  **}**  **void box\_2()**  **{**  **glColor3ub(0,230,250);**  **glLoadIdentity(); //Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(\_move2, 0.0f, 0.0f);**  **glBegin(GL\_QUADS);**  **glVertex2f(0.1f, -0.2f);**  **glVertex2f(0.5f, -0.2f);**  **glVertex2f(0.5f, -0.4f);**  **glVertex2f(0.1f, -0.4);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_bottom(int value)**  **{**  **\_move2 -= .02;**  **if(\_move2 < -1.5)**  **{**  **\_move2 = +1;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(10, update\_bottom, 0);**  **}**  **void Combined()**  **{**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glClearColor(1,1,1,0);**  **box\_1();**  **box\_2();**  **glutSwapBuffers();**  **}**  **int main(int argc, char\*\* argv)**  **{**  **glutInit(&argc, argv);**  **//gluOrtho2D(-2,2,-2,2);**  **glutInitWindowSize(1040, 680);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutCreateWindow("Transformation");**  **glutDisplayFunc(Combined);**  **glutTimerFunc(20, update\_top, 0);**  **glutTimerFunc(2, update\_bottom, 0);//Add a timer**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-2**  Design a car which will have rotating wheels. |
| **Graph Plot (Picture)-** |
| **Code-**  **#include <iostream>**  **#include <GL/gl.h>**  **#include <GL/glut.h>**  **#include <math.h>**  **using namespace std;**  **float \_angle1 = 0.0f;**  **void wheel() {**  **glLoadIdentity(); // Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **// Left wheel**  **glPushMatrix();**  **glTranslatef(0.0, 0.0f, 0.0f);**  **glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);**  **glLineWidth(2.5);**  **glBegin(GL\_LINES); // Draw a circle**  **for (int i = 0; i < 200; i++) {**  **glColor3f(0.5, 0.5, 0.5); // Set wheel color to gray**  **float pi = 3.1416;**  **float A = (i \* 2 \* pi) / 200;**  **float r = 0.065;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x, y);**  **}**  **glEnd();**  **glPopMatrix();**  **// Right wheel**  **glPushMatrix();**  **glTranslatef(0.3, 0.0f, 0.0f);**  **glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);**  **glBegin(GL\_LINES); // Draw another circle**  **for (int i = 0; i < 200; i++) {**  **glColor3f(0.5, 0.5, 0.5); // Set wheel color to gray**  **float pi = 3.1416;**  **float A = (i \* 2 \* pi) / 200;**  **float r = 0.065;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x, y);**  **}**  **glEnd();**  **glPopMatrix();**  **}**  **void drawScene() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glColor3d(1, 0, 0);**  **glLoadIdentity(); // Reset the drawing perspective**  **glOrtho(-5, 5, -5, 5, -5, 5);**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **// Car body**  **glBegin(GL\_QUADS);**  **glColor3f(0.5, 0.2, 0.8); // Modified car body color (purple)**  **glVertex2f(-1.0f, 0.0f);**  **glVertex2f(-1.0f, 1.0f);**  **glVertex2f(2.3f, 1.0f);**  **glVertex2f(2.3f, .0f);**  **glEnd();**  **// Car head**  **glBegin(GL\_QUADS);**  **glColor3f(0.3, 0.7, 0.3); // Modified car head color (green)**  **glVertex2f(-0.5f, 1.0f);**  **glVertex2f(-0.0f, 1.5f);**  **glVertex2f(1.5f, 1.5f);**  **glVertex2f(2.0f, 1.0f);**  **glEnd();**  **// Left wheel**  **glBegin(GL\_POLYGON);**  **for (int i = 0; i < 200; i++) {**  **glColor3f(0.5, 0.5, 0.5); // Set wheel color to gray**  **float pi = 3.1416;**  **float A = (i \* 2 \* pi) / 200;**  **float r = 0.28;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x, y);**  **}**  **glEnd();**  **// Right wheel**  **glBegin(GL\_POLYGON);**  **for (int i = 0; i < 200; i++) {**  **glColor3f(0.5, 0.5, 0.5); // Set wheel color to gray**  **float pi = 3.1416;**  **float A = (i \* 2 \* pi) / 200;**  **float r = 0.28;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x + 1.5, y);**  **}**  **glEnd();**  **wheel();**  **glPopMatrix();**  **glutSwapBuffers();**  **}**  **void update1(int value) {**  **\_angle1 += 2.0f;**  **if (\_angle1 > 360.0) {**  **\_angle1 -= 360;**  **}**  **glutPostRedisplay(); // Notify GLUT that the display has changed**  **glutTimerFunc(20, update1, 0); // Notify GLUT to call update again in 20 milliseconds**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 800);**  **glutCreateWindow("Transformation");**  **glutDisplayFunc(drawScene);**  **gluOrtho2D(-2, 2, -2, 2);**  **glutTimerFunc(20, update1, 0); // Add a timer**  **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)-** |
| **Code-**  **#include <iostream>**  **#include <GL/gl.h>**  **#include <GL/glut.h>**  **#include <math.h>**  **using namespace std;**  **float \_move = 0.0f;**  **float \_move2 = 0.0f;**  **float \_angle1 = 0.0f;**  **void wheel() {**  **glLoadIdentity(); // Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **// glTranslatef(\_move2, 0.0f, 0.0f);**  **glTranslatef(0.0, 0, 0);**  **glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);**  **glLineWidth(2.5);**  **glBegin(GL\_LINES); // Draw a Red 1x1 Square centered at origin**  **for (int i = 0; i < 200; i++) {**  **glColor3f(0.5, 0.5, 0.5); // Gray wheel**  **float pi = 3.1416;**  **float A = (i \* 2 \* pi) / 200;**  **float r = 0.065;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x, y);**  **}**  **glEnd();**  **glPopMatrix();**  **glLoadIdentity(); // Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **// glTranslatef(\_move2, 0.0f, 0.0f);**  **glTranslatef(0.3, 0, 0);**  **glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);**  **glBegin(GL\_LINES); // Draw a Red 1x1 Square centered at origin**  **for (int i = 0; i < 200; i++) {**  **glColor3f(0.5, 0.5, 0.5); // Gray wheel**  **float pi = 3.1416;**  **float A = (i \* 2 \* pi) / 200;**  **float r = 0.065;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x, y);**  **}**  **glEnd();**  **glPopMatrix();**  **}**  **void drawScene() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glColor3d(1, 0, 0);**  **glLoadIdentity(); // Reset the drawing perspective**  **glOrtho(-5, 5, -5, 5, -5, 5);**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(\_move, 0.0f, 0.0f);**  **// Car body**  **glBegin(GL\_QUADS);**  **glColor3f(0.5, 0.2, 0.8); // Modified car body color (purple)**  **glVertex2f(-1.0f, 0.0f);**  **glVertex2f(-1.0f, 1.0f);**  **glVertex2f(2.3f, 1.0f);**  **glVertex2f(2.3f, .0f);**  **glEnd();**  **// Car head (green)**  **glBegin(GL\_QUADS);**  **glColor3f(0.3, 0.7, 0.3); // Modified car head color (green)**  **glVertex2f(-0.5f, 1.0f);**  **glVertex2f(-0.0f, 1.5f);**  **glVertex2f(1.5f, 1.5f);**  **glVertex2f(2.0f, 1.0f);**  **glEnd();**  **// Left wheel**  **glBegin(GL\_POLYGON);**  **for (int i = 0; i < 200; i++) {**  **glColor3f(0.5, 0.5, 0.5); // Gray wheel**  **float pi = 3.1416;**  **float A = (i \* 2 \* pi) / 200;**  **float r = 0.28;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x, y);**  **}**  **glEnd();**  **// Right wheel**  **glBegin(GL\_POLYGON);**  **for (int i = 0; i < 200; i++) {**  **glColor3f(0.5, 0.5, 0.5); // Gray wheel**  **float pi = 3.1416;**  **float A = (i \* 2 \* pi) / 200;**  **float r = 0.28;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x + 1.5, y);**  **}**  **glEnd();**  **glPopMatrix();**  **glutSwapBuffers();**  **}**  **void update(int value) {**  **\_move += .02;**  **if (\_move > 2.3) {**  **\_move = -2.0;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update, 0);**  **}**  **void update2(int value) {**  **\_move2 += .02;**  **if (\_move2 > 1.3) {**  **\_move = -1.0;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update2, 0);**  **}**  **void update1(int value) {**  **\_angle1 += 2.0f;**  **if (\_angle1 > 360.0) {**  **\_angle1 -= 360;**  **}**  **glutPostRedisplay(); // Notify GLUT that the display has changed**  **glutTimerFunc(20, update1, 0); // Notify GLUT to call update again in 25 milliseconds**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 800);**  **glutCreateWindow("Transformation");**  **glutDisplayFunc(drawScene);**  **gluOrtho2D(-2, 2, -2, 2);**  **glutTimerFunc(20, update, 0); // Add a timer**  **glutTimerFunc(20, update1, 0); // Add a timer**  **// glutTimerFunc(20, update2, 0); // Add a timer**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-4**  Design a windmill with rotating blades |
| **Graph Plot (Picture)-** |
| **Code-**  **#include <windows.h> // for MS Windows**  **#include <GL/glut.h> // GLUT, include glu.h and gl.h**  **#include <math.h>**  **#include <iostream>**  **/\* Handler for window-repaint event. Call back when the window first appears and**  **whenever the window needs to be re-painted. \*/**  **int \_angle1 = 0;**  **void init()**  **{**  **glLoadIdentity();**  **gluOrtho2D(0, 500, 0, 500);**  **}**  **void blades()**  **{**  **glBegin(GL\_POLYGON); //blade**  **glColor3ub(0, 0,0);**  **glVertex2f(257, 370);**  **glVertex2f(267, 430);**  **glVertex2f(284, 490);**  **glVertex2f(280, 430);**  **glVertex2f(261, 363);**  **glEnd();**  **}**  **void windmill()**  **{**  **glBegin(GL\_QUADS); //body**  **glColor3ub(55, 25, 55);**  **glVertex2f(250,50);**  **glVertex2f(260, 50);**  **glVertex2f(257, 350);**  **glVertex2f(253, 350);**  **glEnd();**  **glPushMatrix();**  **glTranslatef(255, 360, 0);**  **glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);**  **glTranslatef(-255, -360, 0);**  **blades();//1st**  **glPushMatrix();**  **glTranslatef(255, 360, 0);**  **glRotatef(120, 0.0f, 0.0f, 1.0f);**  **glTranslatef(-255, -360, 0);**  **blades();//2nd**  **glPopMatrix();**  **glPushMatrix();**  **glTranslatef(255, 360, 0);**  **glRotatef(240, 0.0f, 0.0f, 1.0f);**  **glTranslatef(-255, -360, 0);**  **blades(); //3rd blade**  **glPopMatrix();**  **glPopMatrix();**  **}**  **void update(int value)**  **{**  **// std::cout << "updated called";**  **\_angle1 -= 1.0f;**  **glutPostRedisplay(); //Notify GLUT that the display has changed**  **glutTimerFunc(20, update, 0); //Notify GLUT to call update again in 25 milliseconds**  **}**  **void circle(int xx, int yy, float r)**  **{**  **glBegin(GL\_POLYGON); // Draw a Red 1x1 Square centered at origin**  **for (int i = 0; i < 200; i++)**  **{**  **// glColor3ub(255, 215, 0);**  **float pi = 3.1416;**  **float A = (i \* 2 \* pi) / 200;**  **// float r = 4.0f;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x + xx, y + yy);**  **}**  **glEnd();**  **}**  **void display()**  **{**  **glClearColor(0.2f, 0.4f, 0.93f, 1.0f); // Set background color to black and opaque**  **glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)**  **windmill();**  **// CarForward();**  **circle(255, 360, 10);**  **glFlush(); // Render now**  **}**  **/\* Main function: GLUT runs as a console application starting at main() \*/**  **int main(int argc, char\*\* argv)**  **{**  **glutInit(&argc, argv); // Initialize GLUT**  **glutInitWindowSize(1040, 800); // Set the window's initial width & height**  **glutCreateWindow("Windmill Animation"); // Create a window with the given title**  **gluOrtho2D(0, 500, 0, 500);**  **glutDisplayFunc(display); // Register display callback handler for window re-paint**  **glutTimerFunc(20, update, 0); //Add a timer**  **glutMainLoop(); // Enter the event-processing loop**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |