**Lab Taks-5**

Submission Guidelines-

* Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.
* 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)-**  **[Not needed]** |
| **Code-**  **#include <iostream>**  **#include<GL/gl.h>**  **#include <GL/glut.h>**  **using namespace std;**  **float \_move = 0.0f;**  **float move\_ = 0.0f;**  **void drawScene() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glColor3d(1,0,0);**  **glLoadIdentity(); //Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(\_move, 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();**  **glPushMatrix();**  **glTranslatef(move\_, 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();**  **glutSwapBuffers();**  **}**  **void update(int value) {**  **\_move += .02;**  **move\_ -= .02;**  **if(\_move > 1.5)**  **{**  **\_move = -1.5;**  **}**  **if (move\_ < -1.5)**  **{**  **move\_ = 1.5;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update, 0);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 800);**  **glutCreateWindow("BOX");**  **glutDisplayFunc(drawScene);**  **gluOrtho2D(-2,2,-2,2);**  **glutTimerFunc(20, update, 0); //Add a timer**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-2**  Design a car which will have rotating wheels. |
| **Graph Plot (Picture)-**  **[Not needed]** |
| **Code-**  **#include <iostream>**  **#include<GL/gl.h>**  **#include <GL/glut.h>**  **#include <math.h>**  **using namespace std;**  **float \_move = 0.0f;**  **float \_angle1=0.0f;**  **void wheel()**  **{**  **glLoadIdentity(); //Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **//glTranslatef(\_move, 0.0f, 0.0f);**  **glTranslatef(-0.2,0,0);**  **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(1.0,1.0,0.0);**  **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(\_move, 0.0f, 0.0f);**  **glTranslatef(0.2,0,0);**  **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(1.0,1.0,0.0);**  **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**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **//glTranslatef(\_move, 0.0f, 0.0f);**  **glBegin(GL\_QUADS);**  **glVertex2f(-0.3f, 0.0f);**  **glVertex2f(0.3f, 0.0f);**  **glVertex2f(0.3f, 0.2f);**  **glVertex2f(-0.3f, 0.2);**  **glEnd();**  **glBegin(GL\_QUADS);**  **glVertex2f(-0.2f, 0.2f);**  **glVertex2f(0.2f, 0.2f);**  **glVertex2f(0.1f, 0.3f);**  **glVertex2f(-0.1f, 0.3);**  **glEnd();**  **wheel();**  **glPopMatrix();**  **glutSwapBuffers();**  **}**  **void update(int value) {**  **\_move += .02;**  **if(\_move > 1.3)**  **{**  **\_move = -1.0;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update, 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("Moving wheel");**  **glutDisplayFunc(drawScene);**  **gluOrtho2D(-2,2,-2,2);**  **glutTimerFunc(20, update, 0); //Add a timer**  **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)-**  **[Not needed]** |
| **Code-**  **#include <iostream>**  **#include<GL/gl.h>**  **#include <GL/glut.h>**  **#include <math.h>**  **using namespace std;**  **float \_move = 0.0f;**  **float \_angle1=0.0f;**  **void wheel()**  **{**  **glLoadIdentity(); //Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(\_move, 0.0f, 0.0f);**  **glTranslatef(-0.2,0,0);**  **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(1.0,1.0,0.0);**  **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(\_move, 0.0f, 0.0f);**  **glTranslatef(0.2,0,0);**  **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(1.0,1.0,0.0);**  **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**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(\_move, 0.0f, 0.0f);**  **glBegin(GL\_QUADS);**  **glVertex2f(-0.3f, 0.0f);**  **glVertex2f(0.3f, 0.0f);**  **glVertex2f(0.3f, 0.2f);**  **glVertex2f(-0.3f, 0.2);**  **glEnd();**  **glBegin(GL\_QUADS);**  **glVertex2f(-0.2f, 0.2f);**  **glVertex2f(0.2f, 0.2f);**  **glVertex2f(0.1f, 0.3f);**  **glVertex2f(-0.1f, 0.3);**  **glEnd();**  **wheel();**  **glPopMatrix();**  **glutSwapBuffers();**  **}**  **void update(int value) {**  **\_move += .02;**  **if(\_move > 1.3)**  **{**  **\_move = -1.0;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update, 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("Moving car");**  **glutDisplayFunc(drawScene);**  **gluOrtho2D(-2,2,-2,2);**  **glutTimerFunc(20, update, 0); //Add a timer**  **glutTimerFunc(20, update1, 0); //Add a timer**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-4**  Design a windmill with rotating blades |
| **Graph Plot (Picture)-**  **[Not needed]** |
| **Code-**  **#include <iostream>**  **#include<GL/gl.h>**  **#include <GL/glut.h>**  **#include <math.h>**  **using namespace std;**  **float \_move = 0.0f;**  **float \_angle1=0.0f;**  **float \_angle2=0.0f;**  **void wheel()**  **{**  **glLoadIdentity();**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(0.0,0.2,0);**  **glRotatef(\_angle1, 0.0f, 0.0f,1.0f);**  **glBegin(GL\_POLYGON);**  **for(int i=0;i<200;i++)**  **{**  **glColor3f(0.0,0.0,0.0);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=0.025;**  **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);**  **glBegin(GL\_POLYGON);**  **glVertex2f(0.05f, 0.1f);**  **glVertex2f(0.05f,-0.55f);**  **glVertex2f(-0.05f,-0.55f);**  **glVertex2f(-0.05f, 0.1f);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glVertex2f(-0.05f, 0.1f);**  **glVertex2f(0.05f, 0.1f);**  **glVertex2f(0.0f, 0.2f);**  **glEnd();**  **glColor3f(0,255,120);**  **glLoadIdentity();**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(\_move, 0.0f, 0.0f);**  **glTranslatef(0.0,0.2,0);**  **glRotatef(\_angle2, 0.0f, 0.0f,1.0f);**  **glBegin(GL\_POLYGON);**  **glVertex2f(0.0f, 0.05f);**  **glVertex2f(0.35f, 0.05f);**  **glVertex2f(0.35f, -0.05f);**  **glVertex2f(0.0f, -0.05f);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glVertex2f(0.0f, 0.05f);**  **glVertex2f(-0.35f,0.05f);**  **glVertex2f(-0.35f,-0.05f);**  **glVertex2f(0.0f, -0.05f);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glVertex2f(0.05f, 0.0f);**  **glVertex2f(0.05f, 0.35f);**  **glVertex2f(-0.05f, 0.35f);**  **glVertex2f(-0.05f, 0.0f);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glVertex2f(0.05f, 0.0f);**  **glVertex2f(0.05f,-0.35f);**  **glVertex2f(-0.05f,-0.35f);**  **glVertex2f(-0.05f, 0.0f);**  **glEnd();**  **wheel();**  **glPopMatrix();**  **glutSwapBuffers();**  **}**  **void update1(int value) {**  **\_angle1-=2.0f;**  **if(\_angle1 > -360.0)**  **{**  **\_angle1+=-360;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update1, 0);**  **}**  **void update2(int value) {**  **\_angle2-=2.0f;**  **if(\_angle2 > -360.0)**  **{**  **\_angle2+=-360;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update2, 0);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 800);**  **glutCreateWindow("WINDMILL");**  **glutDisplayFunc(drawScene);**  **glutTimerFunc(20, update1, 0);**  **glutTimerFunc(20, update2, 0);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |