**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\_1 = 0.0f;**  **void top\_box()**  **{**  **glColor3d(1,1,0);**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(move\_1, 0.0f, 0.0f);**  **glBegin(GL\_QUADS);**  **glVertex2f(0.0f, 20.0f);**  **glVertex2f(10.0f, 20.0f);**  **glVertex2f(10.0f, 30.0f);**  **glVertex2f(0.0f, 30.0);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_1(int value) {**  **move\_1 += 0.49999; //assume as moving speed**  **if(move\_1>50) //boundary for the positive x axis for last part of the box**  **{**  **move\_1 = -61;//reappear the box**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update\_1, 0);**  **}**  **float move\_2 = 0.0f;**  **void bottom\_box()**  **{**  **glColor3d(1,1,0);**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(move\_2, 0.0f, 0.0f);**  **glBegin(GL\_QUADS);**  **glVertex2f(0.0f, 0.0f);**  **glVertex2f(-10.0f, 0.0f);**  **glVertex2f(-10.0f, 10.0f);**  **glVertex2f(0.0f, 10.0);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_2(int value) {**  **move\_2 -= 0.49999; //assume as moving speed**  **if(move\_2 < -50) //boundary for the negative x axis for last part of the box**  **{**  **move\_2 = 61;//reappear the box**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update\_2, 0);**  **}**  **void identity\_axis()**  **{**  **glLoadIdentity(); //Reset the drawing perspective**  **gluOrtho2D(-50,50,-50,50);**  **}**  **void display() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **top\_box();**  **bottom\_box();**  **glutSwapBuffers();**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 800);**  **glutCreateWindow("Box Move Animation");**  **glutDisplayFunc(display);**  **identity\_axis();**  **glutTimerFunc(20, update\_1, 0); //Add a timer**  **glutTimerFunc(20, update\_2, 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 <windows.h>  #include <math.h>  using namespace std;  void car()  {  //car middle body  glColor3ub(255,0,0);  glBegin(GL\_POLYGON);  glVertex2f(28.0f,81.0f);  glVertex2f(63.0f,81.0f);  glVertex2f(61.0f,101.0f);  glVertex2f(28.0f,101.0f);  glEnd();  //upper part of car  glColor3ub(255,0,0);  glBegin(GL\_POLYGON);  glVertex2f(35.0f,101.0f);  glVertex2f(55.0f,101.0f);  glVertex2f(52.0f,116.0f);  glVertex2f(38.0f,116.0f);  glEnd();  //car rear window glass  glColor3ub(0,0,0);  glBegin(GL\_POLYGON);  glVertex2f(37.0f,103.0f);  glVertex2f(44.5f,103.0f);  glVertex2f(44.5f,114.0f);  glVertex2f(39.0f,114.0f);  glEnd();  //car front window glass  glColor3ub(0,0,0);  glBegin(GL\_POLYGON);  glVertex2f(45.5f,103.0f);  glVertex2f(53.0f,103.0f);  glVertex2f(51.0f,114.0f);  glVertex2f(45.5f,114.0f);  glEnd();  //rear wheel  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=4.1f;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+38,y+83 );  }  glEnd();  //front wheel  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=4.1f;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+52,y+83 );  }  glEnd();  glPopMatrix();  }  //rear first blade  float angle\_rear\_blade\_1 = 0.0f;  void rear\_blade\_1()  {  glColor3d(1,1,0);  //glLoadIdentity(); //Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(38.0f,83.0f,0.0f);  glRotatef(angle\_rear\_blade\_1 , 0.0f, 0.0f,1.0f);  glTranslatef(-38.0f,-83.0f,0.0f);  glBegin(GL\_QUADS);  glVertex2f(37.5f, 80.0f);  glVertex2f(38.5f, 80.0f);  glVertex2f(38.5f, 86.0f);  glVertex2f(37.5f, 86.0f);  glEnd();  glPopMatrix();  }  void update\_rear\_blade\_1 (int value) {  angle\_rear\_blade\_1 -=4.0f;  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(10, update\_rear\_blade\_1, 0); //Notify GLUT to call update again in 25 milliseconds  }  //rear second blade  float angle\_rear\_blade\_2 = 0.0f;  void rear\_blade\_2()  {  glColor3d(1,1,0);  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(38.0f,83.0f,0.0f);  glRotatef(angle\_rear\_blade\_2 , 0.0f, 0.0f,1.0f);  glTranslatef(-38.0f,-83.0f,0.0f);  glBegin(GL\_QUADS);  glVertex2f(35.0f, 82.5f);  glVertex2f(41.0f, 82.5f);  glVertex2f(41.0f, 83.5f);  glVertex2f(35.0f, 83.5f);  glEnd();  glPopMatrix();  }  void update\_rear\_blade\_2 (int value) {  angle\_rear\_blade\_2 -=4.0f;  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(10, update\_rear\_blade\_2, 0); //Notify GLUT to call update again in 25 milliseconds  }  //front first blade  float angle\_front\_blade\_1 = 0.0f;  void front\_blade\_1()  {  glColor3d(1,1,0);  //glLoadIdentity(); //Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(52.0f,83.0f,0.0f);  glRotatef(angle\_front\_blade\_1 , 0.0f, 0.0f,1.0f);  glTranslatef(-52.0f,-83.0f,0.0f);  glBegin(GL\_QUADS);  glVertex2f(51.5f, 80.0f);  glVertex2f(52.5f, 80.0f);  glVertex2f(52.5f, 86.0f);  glVertex2f(51.5f, 86.0f);  glEnd();  glPopMatrix();  }  void update\_front\_blade\_1 (int value) {  angle\_front\_blade\_1 -=4.0f;  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(10, update\_front\_blade\_1, 0); //Notify GLUT to call update again in 25 milliseconds  }  //front second blade  float angle\_front\_blade\_2 = 0.0f;  void front\_blade\_2()  {  glColor3d(1,1,0);  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(52.0f,83.0f,0.0f);  glRotatef(angle\_front\_blade\_2 , 0.0f, 0.0f,1.0f);  glTranslatef(-52.0f,-83.0f,0.0f);  glBegin(GL\_QUADS);  glVertex2f(49.0f, 82.5f);  glVertex2f(55.0f, 82.5f);  glVertex2f(55.0f, 83.5f);  glVertex2f(49.0f, 83.5f);  glEnd();  glPopMatrix();  }  void update\_front\_blade\_2 (int value) {  angle\_front\_blade\_2 -=4.0f;  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(10, update\_front\_blade\_2, 0); //Notify GLUT to call update again in 25 milliseconds  }  void identity\_axis()  {  glLoadIdentity(); //Reset the drawing perspective  gluOrtho2D(0,120,0,220);  }  void display() {  glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT);  car();  rear\_blade\_1();  rear\_blade\_2();  front\_blade\_1();  front\_blade\_2();  glutSwapBuffers();  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Car with moving Wheels");  glutDisplayFunc(display);  identity\_axis();  glutTimerFunc(10, update\_rear\_blade\_1, 0); //Add a timer  glutTimerFunc(10, update\_rear\_blade\_2, 0); //Add a timer  glutTimerFunc(10, update\_front\_blade\_1, 0); //Add a timer  glutTimerFunc(10, update\_front\_blade\_2, 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 <windows.h>**  **#include <math.h>**  **using namespace std;**  **float move\_all =0.0f;**  **void car()**  **{**  **//car middle body**  **glColor3ub(255,0,0);**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(move\_all, 0.0f, 0.0f);//for move**  **glBegin(GL\_POLYGON);**  **glVertex2f(28.0f,81.0f);**  **glVertex2f(63.0f,81.0f);**  **glVertex2f(61.0f,101.0f);**  **glVertex2f(28.0f,101.0f);**  **glEnd();**  **//upper part of car**  **glColor3ub(255,0,0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(35.0f,101.0f);**  **glVertex2f(55.0f,101.0f);**  **glVertex2f(52.0f,116.0f);**  **glVertex2f(38.0f,116.0f);**  **glEnd();**  **//car rear window glass**  **glColor3ub(0,0,0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(37.0f,103.0f);**  **glVertex2f(44.5f,103.0f);**  **glVertex2f(44.5f,114.0f);**  **glVertex2f(39.0f,114.0f);**  **glEnd();**  **//car front window glass**  **glColor3ub(0,0,0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(45.5f,103.0f);**  **glVertex2f(53.0f,103.0f);**  **glVertex2f(51.0f,114.0f);**  **glVertex2f(45.5f,114.0f);**  **glEnd();**  **//rear wheel**  **glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin**  **for(int i=0;i<200;i++)**  **{**  **glColor3ub(0,0,0);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=4.1f;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x+38,y+83 );**  **}**  **glEnd();**  **//front wheel**  **glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin**  **for(int i=0;i<200;i++)**  **{**  **glColor3ub(0,0,0);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=4.1f;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x+52,y+83 );**  **}**  **glEnd();**  **glPopMatrix();**  **}**  **//all object movement**  **void update\_all(int value) {**  **move\_all += 1.0; //assume as moving speed**  **if(move\_all>120) //boundary for the positive x axis for last part of the box**  **{**  **move\_all = -63;//reappear the box**  **}**  **glutPostRedisplay();**  **glutTimerFunc(10, update\_all, 0);**  **}**  **//rear first blade**  **float angle\_rear\_blade\_1 = 0.0f;**  **void rear\_blade\_1()**  **{**  **glColor3d(1,1,0);**  **//glLoadIdentity(); //Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(move\_all, 0.0f, 0.0f);//for move**  **glTranslatef(38.0f,83.0f,0.0f);//for center rotation**  **glRotatef(angle\_rear\_blade\_1 , 0.0f, 0.0f,1.0f);**  **glTranslatef(-38.0f,-83.0f,0.0f);//for center rotation**  **glBegin(GL\_QUADS);**  **glVertex2f(37.5f, 80.0f);**  **glVertex2f(38.5f, 80.0f);**  **glVertex2f(38.5f, 86.0f);**  **glVertex2f(37.5f, 86.0f);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_rear\_blade\_1 (int value) {**  **angle\_rear\_blade\_1 -=4.0f;**  **glutPostRedisplay(); //Notify GLUT that the display has changed**  **glutTimerFunc(10, update\_rear\_blade\_1, 0); //Notify GLUT to call update again in 25 milliseconds**  **}**  **//rear second blade**  **float angle\_rear\_blade\_2 = 0.0f;**  **void rear\_blade\_2()**  **{**  **glColor3d(1,1,0);**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(move\_all, 0.0f, 0.0f);//for move**  **glTranslatef(38.0f,83.0f,0.0f);//for center rotation**  **glRotatef(angle\_rear\_blade\_2 , 0.0f, 0.0f,1.0f);**  **glTranslatef(-38.0f,-83.0f,0.0f);//for center rotation**  **glBegin(GL\_QUADS);**  **glVertex2f(35.0f, 82.5f);**  **glVertex2f(41.0f, 82.5f);**  **glVertex2f(41.0f, 83.5f);**  **glVertex2f(35.0f, 83.5f);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_rear\_blade\_2 (int value) {**  **angle\_rear\_blade\_2 -=4.0f;**  **glutPostRedisplay(); //Notify GLUT that the display has changed**  **glutTimerFunc(10, update\_rear\_blade\_2, 0); //Notify GLUT to call update again in 25 milliseconds**  **}**  **//front first blade**  **float angle\_front\_blade\_1 = 0.0f;**  **void front\_blade\_1()**  **{**  **glColor3d(1,1,0);**  **//glLoadIdentity(); //Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(move\_all, 0.0f, 0.0f);//for move**  **glTranslatef(52.0f,83.0f,0.0f);//for center rotation**  **glRotatef(angle\_front\_blade\_1 , 0.0f, 0.0f,1.0f);**  **glTranslatef(-52.0f,-83.0f,0.0f);//for center rotation**  **glBegin(GL\_QUADS);**  **glVertex2f(51.5f, 80.0f);**  **glVertex2f(52.5f, 80.0f);**  **glVertex2f(52.5f, 86.0f);**  **glVertex2f(51.5f, 86.0f);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_front\_blade\_1 (int value) {**  **angle\_front\_blade\_1 -=4.0f;**  **glutPostRedisplay(); //Notify GLUT that the display has changed**  **glutTimerFunc(10, update\_front\_blade\_1, 0); //Notify GLUT to call update again in 25 milliseconds**  **}**  **//front second blade**  **float angle\_front\_blade\_2 = 0.0f;**  **void front\_blade\_2()**  **{**  **glColor3d(1,1,0);**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(move\_all, 0.0f, 0.0f);//for move**  **glTranslatef(52.0f,83.0f,0.0f);//for center rotation**  **glRotatef(angle\_front\_blade\_2 , 0.0f, 0.0f,1.0f);**  **glTranslatef(-52.0f,-83.0f,0.0f);//for center rotation**  **glBegin(GL\_QUADS);**  **glVertex2f(49.0f, 82.5f);**  **glVertex2f(55.0f, 82.5f);**  **glVertex2f(55.0f, 83.5f);**  **glVertex2f(49.0f, 83.5f);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_front\_blade\_2 (int value) {**  **angle\_front\_blade\_2 -=4.0f;**  **glutPostRedisplay(); //Notify GLUT that the display has changed**  **glutTimerFunc(10, update\_front\_blade\_2, 0); //Notify GLUT to call update again in 25 milliseconds**  **}**  **void identity\_axis()**  **{**  **glLoadIdentity(); //Reset the drawing perspective**  **gluOrtho2D(0,120,0,220);**  **}**  **void display() {**  **glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **car();**  **rear\_blade\_1();**  **rear\_blade\_2();**  **front\_blade\_1();**  **front\_blade\_2();**  **glutSwapBuffers();**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 800);**  **glutCreateWindow("Move Whole Car");**  **glutDisplayFunc(display);**  **identity\_axis();**  **glutTimerFunc(10, update\_all, 0);**  **glutTimerFunc(10, update\_rear\_blade\_1, 0); //Add a timer**  **glutTimerFunc(10, update\_rear\_blade\_2, 0); //Add a timer**  **glutTimerFunc(10, update\_front\_blade\_1, 0); //Add a timer**  **glutTimerFunc(10, update\_front\_blade\_2, 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 <windows.h>**  **#include <math.h>**  **using namespace std;**  **//first blade**  **float angle\_1 = 0.0f;**  **void first\_blade()**  **{**  **glColor3d(1,1,0);**  **//glLoadIdentity(); //Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(31.0f,37.5f,0.0f);**  **glRotatef(angle\_1, 0.0f, 0.0f,1.0f);**  **glTranslatef(-31.0f,-37.5f,0.0f);**  **glBegin(GL\_QUADS);**  **glVertex2f(30.0f, 30.0f);**  **glVertex2f(32.0f, 30.0f);**  **glVertex2f(32.0f, 45.0f);**  **glVertex2f(30.0f, 45.0);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_1 (int value) {**  **angle\_1+=2.0f;**  **glutPostRedisplay(); //Notify GLUT that the display has changed**  **glutTimerFunc(10, update\_1, 0); //Notify GLUT to call update again in 25 milliseconds**  **}**  **//second blade**  **float angle\_2 = 0.0f;**  **void second\_blade()**  **{**  **glColor3d(1,1,0);**  **//glLoadIdentity(); //Reset the drawing perspective**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glTranslatef(31.0f,37.5f,0.0f);**  **glRotatef(angle\_2, 0.0f, 0.0f,1.0f);**  **glTranslatef(-31.0f,-37.5f,0.0f);**  **glBegin(GL\_QUADS);**  **glVertex2f(23.0f, 36.5f);**  **glVertex2f(39.0f, 36.5f);**  **glVertex2f(39.0f, 38.5f);**  **glVertex2f(23.0f, 38.5f);**  **glEnd();**  **glPopMatrix();**  **}**  **void update\_2 (int value) {**  **angle\_2+=2.0f;**  **glutPostRedisplay(); //Notify GLUT that the display has changed**  **glutTimerFunc(10, update\_2, 0); //Notify GLUT to call update again in 25 milliseconds**  **}**  **void windmill\_stand()**  **{**  **glColor3d(1,1,0);**  **glBegin(GL\_QUADS);**  **glVertex2f(30.0f, 5.0f);**  **glVertex2f(32.0f, 5.0f);**  **glVertex2f(32.0f, 37.5f);**  **glVertex2f(30.0f, 37.5f);**  **glEnd();**  **glColor3d(1,1,0);**  **glLineWidth(10.0f);**  **glBegin(GL\_LINES);**  **glVertex2f(28.0f, 5.0f);**  **glVertex2f(34.0f, 5.0f);**  **glEnd();**  **}**  **void circle()**  **{**  **glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin**  **for(int i=0;i<200;i++)**  **{**  **glColor3ub(255,255,0);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=1.8f;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x+31,y+37.5 );**  **}**  **glEnd();**  **}**  **void identity\_axis()**  **{**  **glLoadIdentity(); //Reset the drawing perspective**  **gluOrtho2D(0,50,0,50);**  **}**  **void display() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **first\_blade();**  **second\_blade();**  **windmill\_stand();**  **circle();**  **glutSwapBuffers();**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 800);**  **glutCreateWindow("Windmill");**  **glutDisplayFunc(display);**  **identity\_axis();**  **glutTimerFunc(20, update\_1, 0); //Add a timer**  **glutTimerFunc(20, update\_2, 0); //Add a timer**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |