**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)-** |
| **Code-**  #include <iostream>  #include <GL/gl.h>  #include <GL/glut.h>  using namespace std;  float \_move = 0.0f;  float \_move1 = 0.0f;  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  // Draw red object  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();  // Draw blue object  glColor3d(0, 0, 1);  glLoadIdentity(); // Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-\_move1, 0.0f, 0.0f);  glBegin(GL\_QUADS);  glVertex2f(0.1, 0.0 + 0.3);  glVertex2f(0.5f, 0.0 + 0.3);  glVertex2f(0.5f, 0.2 + 0.3);  glVertex2f(0.1f, 0.2 + 0.3);  glEnd();  glPopMatrix();  glutSwapBuffers();  }  void update(int value) {  \_move += .02;  if (\_move > 1.3) {  \_move = -1.0;  }  \_move1 += .02;  if (\_move1 > 1.3) {  \_move1 = -1.0;  }  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  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  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-2**  Design a car which will have rotating wheels. |
| **Graph Plot (Picture)-** |
| **Code-**  #include <windows.h>  #include <GL/glut.h>  #include <math.h>  #include <vector>  using namespace std;  float \_angle = 0;  struct Color  {  int r;  int g;  int b;  };  void \_polygon(vector<pair<float, float>> coord, Color color = {0, 0, 0}, float Tx = 0, float Ty = 0, float s = 1)  {  glColor3ub(color.r, color.g, color.b);  glBegin(GL\_POLYGON);  for (int i = 0; i < coord.size(); i++) {  glVertex2f(Tx + s \* coord[i].first, Ty + s \* coord[i].second);  }  glEnd();  }  void \_car(Color car = {56,143,188})  {  \_polygon({{152, 223},{621, 223},{629, 306},{586, 346},{586, 356},{362, 356},{283, 306},{166, 290},{152, 253}}, car);  }  void \_wheel(float radius)  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  // here the code will reposition the circle  glTranslatef(243, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3ub(0,0,0);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  // here the code will reposition the circle  glTranslatef(243, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3ub(255,255,255);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius-10;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  // here the code will reposition the circle  glTranslatef(243, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_LINES);  for(int i=0; i<200; i++)  {  glColor3ub(255,255,255);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(537, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3ub(0,0,0);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  // here the code will reposition the circle  glTranslatef(537, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3ub(255,255,255);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius-10;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(537, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_LINES);  for(int i=0; i<200; i++)  {  glColor3ub(255,255,255);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  }  void \_window(Color window = {28,30,71}, Color lights = {255,193,92})  {  \_polygon({{300, 301},{403, 301},{403, 341},{365, 341}}, window);  \_polygon({{300, 301},{353, 301},{353, 341},{300, 341}}, window, 120, 0);  \_polygon({{300, 301},{423, 301},{380, 341},{300, 341}}, window, 190, 0);  \_polygon({{162, 264},{182, 264},{198, 284},{173, 284}}, lights, 0, 0);  }  void \_rotateFunc(int val)  {  \_angle += 2.0f;  if (\_angle > 360.0)  {  \_angle -= 360;  }  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(20, \_rotateFunc, 0); //Notify GLUT to call update again in 25 milliseconds  }  void display() {  glClearColor(0, 0, 0, 1.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  \_car();  \_wheel(40);  \_window();  glFlush(); // Render now  glutSwapBuffers();  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitWindowSize(800, 600);  //glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH)-1100)/2,(glutGet(GLUT\_SCREEN\_HEIGHT)-600)/2);  glutInitWindowPosition(200, 50);  glutCreateWindow("Lab Task - 5");  //this line must be below of glutCreateWindow();  gluOrtho2D(0, 800, 0, 600);  glutDisplayFunc(display);  glutTimerFunc(20, \_rotateFunc, 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)-** |
| Code-  #include <windows.h>  #include <GL/glut.h>  #include <math.h>  #include <vector>  using namespace std;  float \_move = 0;  float \_angle = 0;  struct Color  {  int r;  int g;  int b;  };  void \_polygon(vector<pair<float, float>> coord, Color color = {0, 0, 0}, float Tx = 0, float Ty = 0, float s = 1)  {  glColor3ub(color.r, color.g, color.b);  glBegin(GL\_POLYGON);  for (int i = 0; i < coord.size(); i++) {  glVertex2f(Tx + s \* coord[i].first, Ty + s \* coord[i].second);  }  glEnd();  }  void \_car(Color car = {56,143,188})  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-\_move, 0, 0);  \_polygon({{152, 223},{621, 223},{629, 306},{586, 346},{586, 356},{362, 356},{283, 306},{166, 290},{152, 253}}, car);  glPopMatrix();  }  void \_wheel(float radius)  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-\_move, 0, 0);  // here the code will reposition the circle  glTranslatef(243, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3ub(0,0,0);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-\_move, 0, 0);  // here the code will reposition the circle  glTranslatef(243, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3ub(255,255,255);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius-10;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-\_move, 0, 0);  // here the code will reposition the circle  glTranslatef(243, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_LINES);  for(int i=0; i<200; i++)  {  glColor3ub(255,255,255);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-\_move, 0, 0);  glTranslatef(537, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3ub(0,0,0);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-\_move, 0, 0);  // here the code will reposition the circle  glTranslatef(537, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3ub(255,255,255);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius-10;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-\_move, 0, 0);  glTranslatef(537, 220, 0);  glRotatef(\_angle, 0, 0, 1);  glLineWidth(10);  glBegin(GL\_LINES);  for(int i=0; i<200; i++)  {  glColor3ub(255,255,255);  float pi = 3.1416;  float A = (i\*2\*pi)/200;  float r = radius;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x, y);  }  glEnd();  glPopMatrix();  }  void \_window(Color window = {28,30,71}, Color lights = {255,193,92})  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-\_move, 0, 0);  \_polygon({{300, 301},{403, 301},{403, 341},{365, 341}}, window);  \_polygon({{300, 301},{353, 301},{353, 341},{300, 341}}, window, 120, 0);  \_polygon({{300, 301},{423, 301},{380, 341},{300, 341}}, window, 190, 0);  \_polygon({{162, 264},{182, 264},{198, 284},{173, 284}}, lights, 0, 0);  glPopMatrix();  }  void \_moveFunc(int val)  {  \_move += 4;  if (\_move > 700)  {  \_move = -600;  }  glutPostRedisplay();  glutTimerFunc(20, \_moveFunc, 0);  }  void \_rotateFunc(int val)  {  \_angle += 2.0f;  if (\_angle > 360.0)  {  \_angle -= 360;  }  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(20, \_rotateFunc, 0); //Notify GLUT to call update again in 25 milliseconds  }  void display() {  glClearColor(0, 0, 0, 1.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  \_car();  \_wheel(40);  \_window();  glFlush(); // Render now  glutSwapBuffers();  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitWindowSize(800, 600);  glutInitWindowPosition(200, 50);  glutCreateWindow("Lab Task - 5");  //this line must be below of glutCreateWindow();  gluOrtho2D(0, 800, 0, 600);  glutDisplayFunc(display);  glutTimerFunc(20, \_moveFunc, 0);  glutTimerFunc(20, \_rotateFunc, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-4**  Design a windmill with rotating blades |
| **Graph Plot (Picture)-** |
| **Code-**  #include <windows.h>  #include <GL/glut.h>  #include <math.h>  #include <vector>  using namespace std;  float \_angle = 0;  struct Color  {  int r;  int g;  int b;  };  void \_polygon(vector<pair<float, float>> coord, Color color = {0, 0, 0}, float Tx = 0, float Ty = 0, float s = 1)  {  glColor3ub(color.r, color.g, color.b);  glBegin(GL\_POLYGON);  for (int i = 0; i < coord.size(); i++) {  glVertex2f(Tx + s \* coord[i].first, Ty + s \* coord[i].second);  }  glEnd();  }  void \_body (Color steel = {110, 110, 110}, Color layer = {144, 54, 47})  {  \_polygon({{400, 366},{347, 304},{356, 299},{444, 299},{454, 304}}, layer);  \_polygon({{356, 299},{328, 133},{472, 133},{444, 299}}, steel);  \_polygon({{347, 221},{321, 204},{366, 193},{436, 193},{479, 204},{454, 221},{433, 217},{369, 217}}, layer);  }  void \_propeller1(Color prop = {53, 46, 46})  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(398, 320, 0);  glRotatef(\_angle, 0,0,1);  glTranslatef(-398, -320, 0);  \_polygon({{371, 446},{371, 194},{425, 194},{425, 446}}, prop);  glPopMatrix();  }  void \_propeller2(Color prop = {53, 46, 46})  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(398, 320, 0);  glRotatef(90, 0,0,1);  glTranslatef(-398, -320, 0);  \_polygon({{371, 446},{371, 194},{425, 194},{425, 446}}, prop);  glPopMatrix();  }  void \_rotateFunc(int val)  {  \_angle += 2.0f;  if (\_angle > 360.0)  {  \_angle -= 360;  }  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(20, \_rotateFunc, 0); //Notify GLUT to call update again in 25 milliseconds  }  void display() {  glClearColor(1, 1, 1, 1.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  \_body();  \_propeller1();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(398, 320, 0);  glRotatef(\_angle, 0,0,1);  glTranslatef(-398, -320, 0);  \_propeller2();  glPopMatrix();  glFlush(); // Render now  glutSwapBuffers();  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitWindowSize(800, 600);  glutInitWindowPosition(200, 50);  glutCreateWindow("Lab Task - 5");  //this line must be below of glutCreateWindow();  gluOrtho2D(0, 800, 0, 600);  glutDisplayFunc(display);  glutTimerFunc(20, \_rotateFunc, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |