**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 \_move1 = 0.0f;  float \_move2 = 0.0f;  float speed = 0.02; // Adjust the speed as needed  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1, 0, 0);  glLoadIdentity(); // Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  // Draw the first box  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();  // Draw the second box  glPushMatrix();  glTranslatef(\_move2, 0.0f, 0.0f);  glBegin(GL\_QUADS);  glVertex2f(0.2f, 0.0f);  glVertex2f(0.7f, 0.0f);  glVertex2f(0.7f, 0.2f);  glVertex2f(0.2f, 0.2);  glEnd();  glPopMatrix();  glutSwapBuffers();  }  void update(int value) {  \_move1 -= speed;  \_move2 += speed;  if (\_move1 < -1.3) {  \_move1 = 1.0;  }  if (\_move2 > 1.3) {  \_move2 = -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 <iostream>  #include <GL/gl.h>  #include <GL/glut.h>  #include <windows.h>  #include <math.h>  float \_angle1 = 0.0f;  //float \_move = 0.0f;  using namespace std;  void circle()  {  glBegin(GL\_LINES);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3f(0,0,0.0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=0.1;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x,y );  }  glEnd();  }  void drawCar() {  glClearColor(0.38, 0.81, 0.85, 0.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  glLineWidth(39.5);  glBegin(GL\_QUADS);  glColor3f(0.38,0.43,0.45);  glVertex2f(-60.0, 0.0);  glVertex2f(60.0, 0.0);  glVertex2f(60.0, -60.0);  glVertex2f(-60.0, -60.0);  glEnd();  // Body  glBegin(GL\_QUADS);  glColor3f(0.3, 0.7, 1.0); // Light Blue color for the car body  glVertex2f(-0.5, -0.2);  glVertex2f(0.5, -0.2);  glVertex2f(0.5, 0.2);  glVertex2f(-0.5, 0.2);  glEnd();  // Roof  glBegin(GL\_TRIANGLES);  glColor3f(0.0, 0.5, 1.0);  glVertex2f(-0.5, 0.3);  glVertex2f(0.5, 0.3);  glVertex2f(0.0, 0.5);  glEnd();  // Windows  glBegin(GL\_QUADS);  glColor3f(0.8, 0.8, 0.8); // Light Gray color for windows  glVertex2f(-0.5, 0.1);  glVertex2f(0.5, 0.1);  glVertex2f(0.5, 0.3);  glVertex2f(-0.5, 0.3);  glEnd();  // Left wheel  glPushMatrix();  glTranslatef(-0.3, -0.2, 0.0);  glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);  circle();  //glColor3f(1.0, 1.0, 1.0);  glPopMatrix();  // Right wheel  glPushMatrix();  glTranslatef(0.3, -0.2, 0.0);  glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);  circle();  //glColor3f(1.0, 1.0, 1.0);  glPopMatrix();  }  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1, 0, 0);  glLoadIdentity(); // Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  // glTranslatef(\_move, 0.0, 0.0);  drawCar();  glPopMatrix();  glutSwapBuffers();  }  void update(int value) {  //\* \_move += 0.01;  //if (\_move > 1.3) {  // \_move = -1;  //}  \_angle1 += 2.0f;  if (\_angle1 > 360.0) {  \_angle1 -= 360;  }  glutPostRedisplay(); // Notify GLUT that the display has changed  glutTimerFunc(20, update, 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("Enhanced Car Animation");  glutDisplayFunc(drawScene);  glutTimerFunc(20, update, 0); // Add a timer  gluOrtho2D(-60, 60, -60, 60);  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 <windows.h>  #include <math.h>  float \_angle1 = 0.0f;  float \_move = 0.0f;  void circle(float x, float y, float radius) {  glBegin(GL\_POLYGON);  for (int i = 0; i < 360; i++) {  float angle = i \* 3.14159 / 180;  glVertex2f(x + cos(angle) \* radius, y + sin(angle) \* radius);  }  glEnd();  }  void drawCar() {  // Body  glBegin(GL\_QUADS);  glColor3f(0.0, 0.5, 1.0); // Light Blue color for the car body  glVertex2f(-0.7, -0.3);  glVertex2f(0.7, -0.3);  glVertex2f(0.7, 0.3);  glVertex2f(-0.7, 0.3);  glEnd();  // Roof  glBegin(GL\_TRIANGLES);  glColor3f(0.0, 0.5, 1.0);  glVertex2f(-0.7, 0.3);  glVertex2f(0.7, 0.3);  glVertex2f(0.0, 0.7);  glEnd();  // Windows  glBegin(GL\_QUADS);  glColor3f(0.8, 0.8, 0.8); // Light Gray color for windows  glVertex2f(-0.5, 0.1);  glVertex2f(0.5, 0.1);  glVertex2f(0.5, 0.3);  glVertex2f(-0.5, 0.3);  glEnd();  // Left wheel  glPushMatrix();  glTranslatef(-0.5, -0.3, 0.0);  glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);  circle(0.0, 0.0, 0.1);  glColor3f(1.0, 1.0, 1.0);  glPopMatrix();  // Right wheel  glPushMatrix();  glTranslatef(0.5, -0.3, 0.0);  glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);  circle(0.0, 0.0, 0.1);  glPopMatrix();  }  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1, 0, 0);  glLoadIdentity(); // Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(\_move, 0.0, 0.0);  drawCar();  glPopMatrix();  glutSwapBuffers();  }  void update(int value) {  \_move += 0.01;  if (\_move > 1.3) {  \_move = -1;  }  \_angle1 += 2.0f;  if (\_angle1 > 360.0) {  \_angle1 -= 360;  }  glutPostRedisplay(); // Notify GLUT that the display has changed  glutTimerFunc(20, update, 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("Enhanced Car Animation");  glutDisplayFunc(drawScene);  glutTimerFunc(20, update, 0); // Add a timer  gluOrtho2D(-2, 2, -2, 2);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-4**  Design a windmill with rotating blades |
| **Graph Plot (Picture)-** |
| **Code-**  #include <iostream>  #include <GL/gl.h>  #include <GL/glut.h>  #include <windows.h>  using namespace std;  float \_angle1 = 0.0f;  float \_angleBlade = 0.0f;  void drawBlade() {  glPushMatrix();  glRotatef(\_angleBlade, 0.0f, 0.0f, 1.0f);  glBegin(GL\_TRIANGLES);  glColor3f(0.1, 0.4, 0.3); // Green color for blades  glVertex2f(0.0f, 0.0f);  glVertex2f(0.2f, 0.1f);  glVertex2f(0.2f, -0.1f);  glEnd();  glPopMatrix();  }  void drawWindmill() {  // Windmill body  glBegin(GL\_QUADS);  glColor3f(0.5, 0.5, 0.5); // Gray color for the windmill body  glVertex2f(-0.05f, -0.5f);  glVertex2f(0.05f, -0.5f);  glVertex2f(0.05f, 0.5f);  glVertex2f(-0.05f, 0.5f);  glEnd();  // Draw three blades evenly spaced  glPushMatrix();  glTranslatef(0.0f, 0.5f, 0.0f);  drawBlade();  glRotatef(120.0f, 0.0f, 0.0f, 1.0f);  drawBlade();  glRotatef(120.0f, 0.0f, 0.0f, 1.0f);  drawBlade();  glPopMatrix();  }  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1, 0, 0);  glLoadIdentity(); // Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  // Draw windmill at the center  glPushMatrix();  glRotatef(\_angle1, 0.0f, 0.0f, 1.0f);  drawWindmill();  glPopMatrix();  glutSwapBuffers();  }  void update(int value) {  \_angleBlade += 5.0f;  if (\_angleBlade > 360.0) {  \_angleBlade -= 360;  }  glutPostRedisplay(); // Notify GLUT that the display has changed  glutTimerFunc(20, update, 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("Windmill Animation");  glutDisplayFunc(drawScene);  glutTimerFunc(20, update, 0); // Add a timer  gluOrtho2D(-1, 1, -1, 1); // Set the orthographic view  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |