**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 <windows.h>  #include <GL/glut.h>  float moveBlue = 0.0f;  float moveGreen = 0.0f;  void objectBlue()  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef( 0.0f, moveBlue, 0.0f);  glBegin(GL\_POLYGON);  glColor3f(0.0f, 0.0f, 1.0f);  glVertex2f(0.0f,0.0f);  glVertex2f(2.5f,0.0f);  glVertex2f(2.5f,2.5f);  glVertex2f(0.0f,2.5f);  glEnd();  glPopMatrix();  }  void objectGreen()  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef( 0.0f, moveGreen, 0.0f);  glBegin(GL\_POLYGON);  glColor3f(0.0f, 1.0f, 0.0f);  glVertex2f(0.0f,0.0f);  glVertex2f(-2.5f,0.0f);  glVertex2f(-2.5f,2.5f);  glVertex2f(0.0f,2.5f);  glEnd();  glPopMatrix();  }  void display()  {  glClearColor(0.0f, 0.0f, 0.0f, 1.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  objectBlue();  objectGreen();  glutSwapBuffers();  glFlush();  }  void updateBlue(int value)  {  moveBlue += .04;  if(moveBlue > 9.5)  {  moveBlue = -10.0;  }  glutPostRedisplay();  glutTimerFunc(20, updateBlue, 0);  }  void updateGreen(int value)  {  moveGreen -= .04;  if(moveGreen < -9.5)  {  moveGreen = 10.0;  }  glutPostRedisplay();  glutTimerFunc(20, updateGreen, 0);  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutCreateWindow("OpenGL Setup Test");  glutInitWindowSize(470, 470);  gluOrtho2D(-10,10,-10,10);  glutDisplayFunc(display);  glutTimerFunc(20, updateBlue, 0);  glutTimerFunc(20, updateGreen, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-2**  Design a car which will have rotating wheels. |
| **Graph Plot (Picture)-**  **[Not needed]** |
| Code-  #include <GL/glut.h>  #include <iostream>  #include <cmath>  float carX = 0.0;  float carY = -0.7;  float wheelAngle = 0.0;  float scaleValue = 15.0;  void Car()  {  glPushMatrix();  glTranslatef(carX, carY, 0.0);  glScalef(scaleValue, scaleValue, 1.0);  glScalef(-1.0, 1.0, 1.0);  glColor3f(0.0, 0.0, 1.0);  // Roof  glBegin(GL\_POLYGON);  glVertex2f(-0.4, 0.0);  glVertex2f(-0.3, 0.2);  glVertex2f(0.4, 0.2);  glVertex2f(0.4, 0.0);  glEnd();  // Body  glBegin(GL\_POLYGON);  glVertex2f(-0.4, -0.2);  glVertex2f(-0.4, 0.0);  glVertex2f(0.4, 0.0);  glVertex2f(0.4, -0.2);  glEnd();  // Windows  glColor3f(0.8, 0.8, 1.0);  glBegin(GL\_POLYGON);  glVertex2f(-0.30, 0.05);  glVertex2f(-0.30, 0.15);  glVertex2f(-0.10, 0.15);  glVertex2f(-0.10, 0.05);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(0.15, 0.05);  glVertex2f(0.15, 0.15);  glVertex2f(0.35, 0.15);  glVertex2f(0.35, 0.05);  glEnd();  //door  glBegin(GL\_LINES);  glColor3f(0.0, 0.0, 0.0);  glVertex2f(-0.05, -0.2);  glVertex2f(-0.05, 0.2);  glEnd();  glPopMatrix();  }  void Wheels()  {  const float wheelRadius = 0.1 \* scaleValue;  const float wheelY = -0.28 \* scaleValue;  // Left wheel  glPushMatrix();  glColor3f(0.0, 0.0, 0.0);  glPushMatrix();  glTranslatef(-0.25 \* scaleValue, wheelY, 0.0);  glRotatef(wheelAngle, 0.0, 0.0, 1.0);  glBegin(GL\_POLYGON);  for (int i = 0; i < 360; i++)  {  float angle = i \* 3.14159 / 180.0;  float x = wheelRadius \* cos(angle);  float y = wheelRadius \* sin(angle);  glVertex2f(x, y);  }  glEnd();  glColor3f(1.0, 1.0, 1.0);  glBegin(GL\_LINES);  glVertex2f(1.5, 0.0);  glVertex2f(-1.5, 0.0);  glEnd();  glBegin(GL\_LINES);  glVertex2f(0.0, 1.5);  glVertex2f(0.0, -1.5);  glEnd();  glBegin(GL\_LINES);  glVertex2f(1.2, 0.9);  glVertex2f(-1.2, -0.9);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-0.9, 1.2);  glVertex2f(0.9, -1.2);  glEnd();  glPopMatrix();  // Right wheel  glPushMatrix();  glColor3f(0.0, 0.0, 0.0);  glTranslatef(0.25 \* scaleValue, wheelY, 0.0);  glRotatef(wheelAngle, 0.0, 0.0, 1.0);  glBegin(GL\_POLYGON);  for (int i = 0; i < 360; i++)  {  float angle = i \* 3.14159 / 180.0;  float x = wheelRadius \* cos(angle);  float y = wheelRadius \* sin(angle);  glVertex2f(x, y);  }  glEnd();  glColor3f(1.0, 1.0, 1.0);  glBegin(GL\_LINES);  glVertex2f(1.5, 0.0);  glVertex2f(-1.5, 0.0);  glEnd();  glBegin(GL\_LINES);  glVertex2f(0.0, 1.5);  glVertex2f(0.0, -1.5);  glEnd();  glBegin(GL\_LINES);  glVertex2f(1.2, 0.9);  glVertex2f(-1.2, -0.9);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-0.9, 1.2);  glVertex2f(0.9, -1.2);  glEnd();  glPopMatrix();  }  void Rotate(int value)  {  wheelAngle -= 2.5;  glutPostRedisplay();  glutTimerFunc(16, Rotate, 0);  }  void Display()  {  glClear(GL\_COLOR\_BUFFER\_BIT);  Car();  Wheels();  glutSwapBuffers();  glutPostRedisplay();  }  void Reshape(int w, int h)  {  glViewport(0, 0, w, h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluOrtho2D(-5.0, 5.0, -5.0, 5.0);  glMatrixMode(GL\_MODELVIEW);  }  void Init()  {  glClearColor(1.0, 1.0, 1.0, 1.0);  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutCreateWindow("Car");  glutReshapeWindow(580, 520);  gluOrtho2D(-5.0, 5.0, -5.0, 5.0);  glutDisplayFunc(Display);  glutReshapeFunc(Reshape);  Init();  glutTimerFunc(16, Rotate, 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)-**  **[Not needed]** |
| **Code-**  #include <GL/glut.h>  #include <iostream>  #include <cmath>  float carX = 0.0;  float carY = -0.7;  float wheelAngle = 0.0;  float scaleValue = 15.0;  void Car()  {  glPushMatrix();  glTranslatef(carX, carY, 0.0);  glScalef(scaleValue, scaleValue, 1.0);  glScalef(-1.0, 1.0, 1.0);  glColor3f(0.0, 0.0, 1.0);  // Roof  glBegin(GL\_POLYGON);  glVertex2f(-0.4, 0.0);  glVertex2f(-0.3, 0.2);  glVertex2f(0.4, 0.2);  glVertex2f(0.4, 0.0);  glEnd();  // Body  glBegin(GL\_POLYGON);  glVertex2f(-0.4, -0.2);  glVertex2f(-0.4, 0.0);  glVertex2f(0.4, 0.0);  glVertex2f(0.4, -0.2);  glEnd();  // Windows  glColor3f(0.8, 0.8, 1.0);  glBegin(GL\_POLYGON);  glVertex2f(-0.30, 0.05);  glVertex2f(-0.30, 0.15);  glVertex2f(-0.10, 0.15);  glVertex2f(-0.10, 0.05);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(0.15, 0.05);  glVertex2f(0.15, 0.15);  glVertex2f(0.35, 0.15);  glVertex2f(0.35, 0.05);  glEnd();  //door  glBegin(GL\_LINES);  glColor3f(0.0, 0.0, 0.0);  glVertex2f(-0.05, -0.19);  glVertex2f(-0.05, 0.2);  glEnd();  glPopMatrix();  }  void Wheels()  {  const float wheelRadius = 0.1 \* scaleValue;  const float wheelY = -0.28 \* scaleValue;  // Left wheel  glPushMatrix();  glColor3f(0.0, 0.0, 0.0);  glTranslatef((-0.25 \* scaleValue) + carX, wheelY, 0.0);  glRotatef(wheelAngle, 0.0, 0.0, 1.0);  glBegin(GL\_POLYGON);  for (int i = 0; i < 360; i++)  {  float angle = i \* 3.14159 / 180.0;  float x = wheelRadius \* cos(angle);  float y = wheelRadius \* sin(angle);  glVertex2f(x, y);  }  glEnd();  glColor3f(1.0, 1.0, 1.0);  glBegin(GL\_LINES);  glVertex2f(1.5, 0.0);  glVertex2f(-1.5, 0.0);  glEnd();  glBegin(GL\_LINES);  glVertex2f(0.0, 1.5);  glVertex2f(0.0, -1.5);  glEnd();  glBegin(GL\_LINES);  glVertex2f(1.2, 0.9);  glVertex2f(-1.2, -0.9);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-0.9, 1.2);  glVertex2f(0.9, -1.2);  glEnd();  glPopMatrix();  // Right wheel  glPushMatrix();  glColor3f(0.0, 0.0, 0.0);  glTranslatef((0.25 \* scaleValue) + carX, wheelY, 0.0);  glRotatef(wheelAngle, 0.0, 0.0, 1.0);  glBegin(GL\_POLYGON);  for (int i = 0; i < 360; i++)  {  float angle = i \* 3.14159 / 180.0;  float x = wheelRadius \* cos(angle);  float y = wheelRadius \* sin(angle);  glVertex2f(x, y);  }  glEnd();  glColor3f(1.0, 1.0, 1.0);  glBegin(GL\_LINES);  glVertex2f(1.5, 0.0);  glVertex2f(-1.5, 0.0);  glEnd();  glBegin(GL\_LINES);  glVertex2f(0.0, 1.5);  glVertex2f(0.0, -1.5);  glEnd();  glBegin(GL\_LINES);  glVertex2f(1.2, 0.9);  glVertex2f(-1.2, -0.9);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-0.9, 1.2);  glVertex2f(0.9, -1.2);  glEnd();  glPopMatrix();  }  void Rotate(int value)  {  wheelAngle -= 2.5;  glutPostRedisplay();  glutTimerFunc(16, Rotate, 0);  }  void Translate(int value)  {  carX += 0.1;  if(carX>28.0)  {  carX=-28.0;  }  glutPostRedisplay();  glutTimerFunc(16, Translate, 0);  }  void Display()  {  glClear(GL\_COLOR\_BUFFER\_BIT);  Car();  Wheels();  glutSwapBuffers();  glutPostRedisplay();  }  void Reshape(int w, int h)  {  glViewport(0, 0, w, h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluOrtho2D(-5.0, 5.0, -5.0, 5.0);  glMatrixMode(GL\_MODELVIEW);  }  void Init()  {  glClearColor(1.0, 1.0, 1.0, 1.0);  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutCreateWindow("Car");  glutReshapeWindow(580, 520);  gluOrtho2D(-5.0, 5.0, -5.0, 5.0);  glutDisplayFunc(Display);  glutReshapeFunc(Reshape);  Init();  glutTimerFunc(16, Rotate, 0);  glutTimerFunc(16, Translate, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-4**  Design a windmill with rotating blades |
| **Graph Plot (Picture)-**  **[Not needed]** |
| **Code-**  #include<windows.h>  #include <GL/gl.h>  #include <GL/glut.h>  #include <math.h>  int frameNumber = 0;  void drawWindmill()  {  int i;  glColor3f(90.0/255, 90.0/255, 90.0/255);  glBegin(GL\_POLYGON);  glVertex2f(-0.05f, 0);  glVertex2f(-0.05f, 3);  glVertex2f(0.05f, 3);  glVertex2f(0.05f, 0);  glEnd();  glTranslatef(0, 3, 0);  glColor3f(0.0, 0.0, 0.0);  glRotated(frameNumber \* (180.0 / 45), 0, 0, 1);  for (i = 0; i < 4; i++)  {  glRotated(90, 0, 0, 1);  glBegin(GL\_POLYGON);  glVertex2f(0, 0);  glVertex2f(1.0f, 0.2f);  glVertex2f(1.0f, -0.2f);  glEnd();  }  }  void display()  {  glClear(GL\_COLOR\_BUFFER\_BIT);  glLoadIdentity();  glPushMatrix();  glTranslated(3.7, 0.8, 0);  glScaled(0.7, 0.7, 1);  drawWindmill();  glPopMatrix();  glutSwapBuffers();  }  void Frame(int v)  {  frameNumber++;  glutPostRedisplay();  glutTimerFunc(10, Frame, 0);  }  void init()  {  glClearColor(1, 1, 1, 0);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  glOrtho(0, 7, -1, 4, -1, 1);  glMatrixMode(GL\_MODELVIEW);  }  int main(int argc, char \*\*argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE);  glutInitWindowSize(700, 470);  glutInitWindowPosition(100, 100);  glutCreateWindow("windmill");  init();  glutDisplayFunc(display);  glutTimerFunc(200, Frame, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |