**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 obj()  {  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(-\_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();  glutSwapBuffers();  }  void update(int value)  {  \_move += .02;  \_move1 += .02;  if(\_move > 1.6)  {  \_move = -1.0;  }  if(\_move1 > 1.6)  {  \_move1 = -1.0;  }  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  void display()  {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(0,0,1);  glLoadIdentity();  obj();  glFlush();  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInit(&argc, argv); // Initialize GLUT  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(720, 720); // Set the window's initial width & height  glutCreateWindow("Transformation");  glutDisplayFunc(display); // Register callback handler for window re-paint event  gluOrtho2D(5,5,5,5);  glutTimerFunc(20, update, 0); //Add a timer  glutMainLoop(); // Enter the event-processing loop  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>  using namespace std;  float \_angle1 = 0.0f;  void circle(float radius, float xc, float yc, float r, float g, float b)  {  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3f(r,g,b);  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+xc,y+yc);  }  glEnd();  }  void sky()  {  glBegin(GL\_POLYGON);  glColor3f(0.008,0.102,0.204);  glVertex2f(-60.0f, -5.0f);  glVertex2f(-60.0, 60.0f);  glVertex2f(95.0f, 60.0f);  glVertex2f(95.0f, -5.0f);  glEnd();  }  void road()  {  glBegin(GL\_POLYGON);  glColor3f(0.35f,0.35f,0.35f);  glVertex2f(-60.0f, -30.0f);  glVertex2f(-60.0, -5.0f);  glVertex2f(95.0f, -5.0f);  glVertex2f(95.0f, -30.0f);  glEnd();  }  void mountain()  {  glBegin(GL\_POLYGON);  glColor3f(0.659f,0.627,0.584f);  glVertex2f(-60.0f, -5.0f);  glVertex2f(-60.0, 20.0f);  glVertex2f(-50.0f, 30.0f);  glVertex2f(-20.0f, -5.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.659f,0.627,0.584f);  glVertex2f(-50.0f, -5.0f);  glVertex2f(-20.0, 40.0f);  glVertex2f(80.0f, -5.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.659f,0.627,0.584f);  glVertex2f(-20.0f, -5.0f);  glVertex2f(30.0, 40.0f);  glVertex2f(95.0f, -5.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.659f,0.627,0.584f);  glVertex2f(35.0f, -5.0f);  glVertex2f(95.0, 40.0f);  glVertex2f(95.0f, -5.0f);  glEnd();  }  void car()  {  glMatrixMode(GL\_MODELVIEW);  //  //Car  //  glBegin(GL\_POLYGON);  glColor3f(0.125f,0.561,0.075f);  glVertex2f(50.0f, -10.0f);  glVertex2f(0.0, -10.0f);  glVertex2f(-8.0f, -8.0f);  glVertex2f(-8.0f, -2.0f);  glVertex2f(6.0f, 0.0f);  glVertex2f(12.0f, 8.0f);  glVertex2f(34.0f, 8.0f);  glVertex2f(38.0f, 0.0f);  glVertex2f(50.0f, 0.0f);  glVertex2f(50.0f, -10.0f);  glEnd();  //  // 1st Window  //  glBegin(GL\_POLYGON);  glColor3f(0.5f,1.0,1.0f);  glVertex2f(21.3f, 0.0f);  glVertex2f(7.4f, 0.0f);  glVertex2f(12.8f, 7.0f);  glVertex2f(21.3f, 7.0f);  glEnd();  //  // 2nd Window  //  glBegin(GL\_POLYGON);  glColor3f(0.5f,1.0,1.0f);  glVertex2f(36.5f, 0.0f);  glVertex2f(22.3f, 0.0f);  glVertex2f(22.3f, 7.0f);  glVertex2f(32.2f, 7.0f);  glEnd();  glPushMatrix();  glTranslatef(8.0,-10.0,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(-8.0,10.0,0.0);  circle(4.24264, 8.0, -10.0, 0.35,0.35,0.35);  circle(3.62243, 8.0, -10.0, 0.0,0.0,0.0);  circle(0.55, 8.0, -8.0, 1.0,0.0,0.0);  circle(0.55, 8.0, -12.0, 1.0,0.0,0.0);  glPopMatrix();  glPushMatrix();  glTranslatef(38.0,-10.0,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(-38.0,10.0,0.0);  circle(4.24264, 38.0, -10.0, 0.35,0.35,0.35);  circle(3.62243, 38.0, -10.0, 0.0,0.0,0.0);  circle(0.55, 38.0, -8.0, 1.0,0.0,0.0);  circle(0.55, 38.0, -12.0, 1.0,0.0,0.0);  glPopMatrix();  glutSwapBuffers();  }  void update(int value)  {  \_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  }  void display()  {  glClear(GL\_COLOR\_BUFFER\_BIT);  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);  glColor3d(1,0,0);  //glLoadIdentity();  sky();  mountain();  road();  car();  glFlush();  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv); // Initialize GLUT  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(720, 720); // Set the window's initial width & height  glutCreateWindow("Moving Wheel");  glutDisplayFunc(display); // Register callback handler for window re-paint event  gluOrtho2D(-60,80,-30,60);  glutTimerFunc(30, update, 0); //Add a timer  glutMainLoop(); // Enter the event-processing loop  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>  using namespace std;  float \_angle1 = 0.0f;  float \_move = 0.0f;  void circle(float radius, float xc, float yc, float r, float g, float b)  {  glBegin(GL\_POLYGON);  for(int i=0; i<200; i++)  {  glColor3f(r,g,b);  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+xc,y+yc);  }  glEnd();  }  void sky()  {  glBegin(GL\_POLYGON);  glColor3f(0.008,0.102,0.204);  glVertex2f(-60.0f, -5.0f);  glVertex2f(-60.0, 60.0f);  glVertex2f(95.0f, 60.0f);  glVertex2f(95.0f, -5.0f);  glEnd();  }  void road()  {  glBegin(GL\_POLYGON);  glColor3f(0.35f,0.35f,0.35f);  glVertex2f(-60.0f, -30.0f);  glVertex2f(-60.0, -5.0f);  glVertex2f(95.0f, -5.0f);  glVertex2f(95.0f, -30.0f);  glEnd();  }  void mountain()  {  glBegin(GL\_POLYGON);  glColor3f(0.659f,0.627,0.584f);  glVertex2f(-60.0f, -5.0f);  glVertex2f(-60.0, 20.0f);  glVertex2f(-50.0f, 30.0f);  glVertex2f(-20.0f, -5.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.659f,0.627,0.584f);  glVertex2f(-50.0f, -5.0f);  glVertex2f(-20.0, 40.0f);  glVertex2f(80.0f, -5.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.659f,0.627,0.584f);  glVertex2f(-20.0f, -5.0f);  glVertex2f(30.0, 40.0f);  glVertex2f(95.0f, -5.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.659f,0.627,0.584f);  glVertex2f(35.0f, -5.0f);  glVertex2f(95.0, 40.0f);  glVertex2f(95.0f, -5.0f);  glEnd();  }  void car()  {  glMatrixMode(GL\_MODELVIEW);  //  //Car  //  glPushMatrix();  glTranslatef(\_move, 0.0f, 0.0f);  glBegin(GL\_POLYGON);  glColor3f(0.125f,0.561,0.075f);  glVertex2f(50.0f, -10.0f);  glVertex2f(0.0, -10.0f);  glVertex2f(-8.0f, -8.0f);  glVertex2f(-8.0f, -2.0f);  glVertex2f(6.0f, 0.0f);  glVertex2f(12.0f, 8.0f);  glVertex2f(34.0f, 8.0f);  glVertex2f(38.0f, 0.0f);  glVertex2f(50.0f, 0.0f);  glVertex2f(50.0f, -10.0f);  glEnd();  //  // 1st Window  //  glBegin(GL\_POLYGON);  glColor3f(0.5f,1.0,1.0f);  glVertex2f(21.3f, 0.0f);  glVertex2f(7.4f, 0.0f);  glVertex2f(12.8f, 7.0f);  glVertex2f(21.3f, 7.0f);  glEnd();  //  // 2nd Window  //  glBegin(GL\_POLYGON);  glColor3f(0.5f,1.0,1.0f);  glVertex2f(36.5f, 0.0f);  glVertex2f(22.3f, 0.0f);  glVertex2f(22.3f, 7.0f);  glVertex2f(32.2f, 7.0f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(\_move, 0.0f, 0.0f);  glTranslatef(8.0,-10.0,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(-8.0,10.0,0.0);  circle(4.24264, 8.0, -10.0, 0.35,0.35,0.35);  circle(3.62243, 8.0, -10.0, 0.0,0.0,0.0);  circle(0.55, 8.0, -8.0, 1.0,0.0,0.0);  circle(0.55, 8.0, -12.0, 1.0,0.0,0.0);  glPopMatrix();  glPushMatrix();  glTranslatef(\_move, 0.0f, 0.0f);  glTranslatef(38.0,-10.0,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(-38.0,10.0,0.0);  circle(4.24264, 38.0, -10.0, 0.35,0.35,0.35);  circle(3.62243, 38.0, -10.0, 0.0,0.0,0.0);  circle(0.55, 38.0, -8.0, 1.0,0.0,0.0);  circle(0.55, 38.0, -12.0, 1.0,0.0,0.0);  glPopMatrix();  glutSwapBuffers();  }  void update(int value)  {  \_move += 1.0;  if(\_move > 95.0)  {  \_move = -90.0;  }  \_angle1-=2.0f;  if(\_angle1 > 360.0)  {  \_angle1-=360;  }  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  void display()  {  glClear(GL\_COLOR\_BUFFER\_BIT);  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);  glColor3d(1,0,0);  sky();  mountain();  road();  car();  glFlush();  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv); // Initialize GLUT  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(720, 720); // Set the window's initial width & height  glutCreateWindow("Moving Wheel");  glutDisplayFunc(display); // Register callback handler for window re-paint event  gluOrtho2D(-60,80,-30,60);  glutTimerFunc(30, update, 0); //Add a timer  glutMainLoop(); // Enter the event-processing loop  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>  #include <math.h>  using namespace std;  float \_angle1 = 0.0f;  void circle(float radius, float xc, float yc, float r, float g, float b)  {  glBegin(GL\_POLYGON);  for(int i=0;i<200;i++)  {  glColor3f(r,g,b);  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+xc,y+yc);  }  glEnd();  }  void stand()  {  glMatrixMode(GL\_MODELVIEW);  glBegin(GL\_POLYGON);  glColor3f(0.8f,0.51,0.102f);  glVertex2f(8.8f, 9.0f);  glVertex2f(9.2f, 9.0f);  glVertex2f(10.0f, 1.0f);  glVertex2f(8.0f, 1.0f);  glEnd();  circle(0.72111, 9.0, 9.0, 0.65,0.65,0.65);  glPushMatrix();  glTranslatef(9.0,9.0,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(-9.0,-9.0,0.0);  glBegin(GL\_POLYGON);  glColor3f(0.75f, 0.75f, 0.75f);  glVertex2f(9.8, 9.8f);  glVertex2f(12.0f, 12.0f);  glVertex2f(11.6f, 12.4f);  glVertex2f(9.4f, 10.2f);  glVertex2f(9.4f, 9.8f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.75f, 0.75f, 0.75f);  glVertex2f(9.8, 9.8f);  glVertex2f(9.4f, 9.8f);  glVertex2f(9.0f, 9.0f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(9.0,9.0,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(-9.0,-9.0,0.0);  glBegin(GL\_POLYGON);  glColor3f(0.75f, 0.75f, 0.75f);  glVertex2f(5.6f, 11.6f);  glVertex2f(6.0f, 12.0f);  glVertex2f(8.2f, 9.8f);  glVertex2f(8.2f, 9.4f);  glVertex2f(7.8f, 9.4f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.75f, 0.75f, 0.75f);  glVertex2f(8.2f, 9.8f);  glVertex2f(9.0f, 9.0f);  glVertex2f(8.2f, 9.4f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(9.0,9.0,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(-9.0,-9.0,0.0);  glBegin(GL\_POLYGON);  glColor3f(0.75f, 0.75f, 0.75f);  glVertex2f(6.4f, 5.6f);  glVertex2f(6.0f, 6.0f);  glVertex2f(8.4f, 8.4f);  glVertex2f(8.4f, 8.4f);  glVertex2f(8.6f, 8.2f);  glVertex2f(8.6f, 7.8f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.75f, 0.75f, 0.75f);  glVertex2f(8.4f, 8.4f);  glVertex2f(9.0f, 9.0f);  glVertex2f(8.6f, 8.2f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(9.0,9.0,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(-9.0,-9.0,0.0);  glBegin(GL\_POLYGON);  glColor3f(0.75f, 0.75f, 0.75f);  glVertex2f(9.6f, 8.4f);  glVertex2f(12.0f, 6.0f);  glVertex2f(12.4f, 6.4f);  glVertex2f(10.2f, 8.6f);  glVertex2f(9.8f, 8.6f);  glBegin(GL\_POLYGON);  glColor3f(0.75f, 0.75f, 0.75f);  glVertex2f(9.6f, 8.4f);  glVertex2f(9.0f, 9.0f);  glVertex2f(9.8f, 8.6f);  glEnd();  glPopMatrix();  circle(0.5, 9.0, 9.0, 0.35,0.35,0.35);  glutSwapBuffers();  }  void update(int value){  \_angle1-=1.0f;  if(\_angle1 > 360.0)  {  \_angle1-=360;  }  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);  glColor3d(1,0,0);  stand();  glFlush();  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv); // Initialize GLUT  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(720, 720); // Set the window's initial width & height  glutCreateWindow("Windmill");  glutDisplayFunc(display); // Register callback handler for window re-paint event  gluOrtho2D(-5,25,-5,15);  glutTimerFunc(30, update, 0); //Add a timer  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-** |