**Lab Taks-5**

|  |
| --- |
| **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;  int state1 = 1;  int state2 = 2;  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1,0,0);  glLoadIdentity(); //Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(0.0f,\_move, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(0.6f, 1.0f);  glVertex2f(0.6f, 0.8f);  glVertex2f(0.8f, 0.8f);  glVertex2f(0.8f, 1.0f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(0.0f,\_move1, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(-0.8f, -0.8f);  glVertex2f(-0.8f, -1.0f);  glVertex2f(-0.6f, -1.0f);  glVertex2f(-0.6f, -0.8f);  glEnd();  glPopMatrix();  glutSwapBuffers();  }  //for object 1  void update(int value) {  switch(state1){  case 1:  if (\_move < 0.0){  \_move += 0.03;  }  else{  state1 = -1;  }  break;  case -1:  if (\_move > -1.8 ){  \_move -= 0.03;  }  else {  state1 = 1;  }  break;  }  /\* \_move -= .02;  if(\_move < -2)  {  \_move = 0.0f;  }\*/  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  //for animation bottom to up  void update1(int value) {  switch(state2){  case 2:  if (\_move1 > 0.0){  \_move1 -= 0.03;  }  else{  state2 = -2;  }  break;  case -2:  if (\_move1 < 1.8 ){  \_move1 += 0.03;  }  else {  state2 = 2;  }  break;  }  /\*\_move1 += .02;  if(\_move1 > 2)  {  \_move1 = 0.0f;  }\*/  glutPostRedisplay();  glutTimerFunc(20, update1, 0);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Transformation");  //gluOrtho2D(-2,2,-2,2);  glutDisplayFunc(drawScene);  glutTimerFunc(20, update, 0); //Add a timer  glutTimerFunc(20, update1, 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> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  #include <math.h>  /\* Handler for window-repaint event. Call back when the window first appears and  whenever the window needs to be re-painted. \*/  float \_move = 0.0f;  float \_angle1 = 0.0f;  void back\_jungles(){  //extreme back jungles  glBegin(GL\_POLYGON);  glColor3f(0.1, 0.6, 0);  glVertex2f(-40, -16);  glVertex2f(-40, 2.2);  glVertex2f(-38, 4);  glVertex2f(-32.5, 5.98);  glVertex2f(-17.6, 11.2);  glVertex2f(4, 15);  glVertex2f(29, 11.5);  glVertex2f(45, -1);  glVertex2f(45, -14);  glEnd();  // back small jungles  glBegin(GL\_POLYGON);  glColor3f(0, 0.9, 0);  glVertex2f(-40, -10);  glVertex2f(-38, -8);  glVertex2f(-36, -6.706921);  glVertex2f(-32.9358216, -6.706921);  glVertex2f(-31, -8);  glVertex2f(-29, -10);  glVertex2f(-27, -6);  glVertex2f(-25, -3);  glVertex2f(-21, -2);  glVertex2f(-19, -3);  glVertex2f(-17, -6);  glVertex2f(-14, -10);  glVertex2f(-12, -11);  glVertex2f(-11, -9);  glVertex2f(-8, -9);  glVertex2f(-7, -10);  glVertex2f(-6, -8);  glVertex2f(-5, -5);  glVertex2f(-2,-4);  glVertex2f(0,-4);  glVertex2f(3,-7);  glVertex2f(4.5,-7.5);  glVertex2f(5,-8.5);  glVertex2f(10,-5);  glVertex2f(14,-4.5);  glVertex2f(16,-5.5);  glVertex2f(17.5,-7.5);  glVertex2f(19,-10.5);  glVertex2f(21.5,-8.5);  glVertex2f(25,-7);  glVertex2f(28.5,-6.5);  glVertex2f(32.5,-8);  glVertex2f(34.5,-11.5);  glVertex2f(36.5,-13);  glVertex2f(38.5,-10);  glVertex2f(41,-8.5);  glVertex2f(45,-8);  glVertex2f(45, -16);  glVertex2f(-40, -16);  glEnd();  }  void traffic\_poll(){  //black area under traffic light poll  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(-37, -14);  glVertex2f(-37, -16);  glVertex2f(45, -16);  glVertex2f(45, -14);  glEnd();  // light poll -- black area  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(-36, 2);  glVertex2f(-36, -14);  glVertex2f(-34.5, -14);  glVertex2f(-34.5, 2);  glEnd();  // light gray area of poll  glBegin(GL\_POLYGON);  glColor3f(0.3, 0.3, 0.3);  glVertex2f(-38, 12);  glVertex2f(-38, 2);  glVertex2f(-32.5, 2);  glVertex2f(-32.5, 12);  glEnd();  }  void three\_circle(float radius, float xc, float yc, float r, float g, float b)  {  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  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 road(){  //road  glBegin(GL\_POLYGON);  glColor3f(0.3, 0.3, 0.3);  glVertex2f(-40, -16);  glVertex2f(-40, -30);  glVertex2f(45, -30);  glVertex2f(45, -16);  glEnd();  //white zebra crossing signs  //road  glColor3f(1, 1, 1);  glBegin(GL\_POLYGON);  glVertex2f(-36, -16.7);  glVertex2f(-36.5, -18.2);  glVertex2f(-25.5, -18.2);  glVertex2f(-25, -16.7);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-36.5, -19);  glVertex2f(-37, -20.5);  glVertex2f(-26, -20.5);  glVertex2f(-25.5, -19);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-37, -22);  glVertex2f(-37.5, -23.5);  glVertex2f(-26.5, -23.5);  glVertex2f(-26, -22);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-37.5, -25);  glVertex2f(-38, -26.5);  glVertex2f(-27, -26.5);  glVertex2f(-26.5, -25);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-38, -28);  glVertex2f(-38.5, -29.5);  glVertex2f(-27.5, -29.5);  glVertex2f(-27, -28);  glEnd();  //two under the car  glBegin(GL\_POLYGON);  glVertex2f(-2, -22);  glVertex2f(-3, -23);  glVertex2f(5, -23);  glVertex2f(6, -22);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(16, -22);  glVertex2f(15, -23);  glVertex2f(24, -23);  glVertex2f(24.8, -22);  glEnd();  }  void car(){  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-30, 0.0f, 0.0f);  glRotatef(180,0.0f, 1.0f, 0.0f);  glTranslatef(\_move, 0.0f, 0.0f);  //red body  glBegin(GL\_POLYGON);  glColor3f(1, 0, 0);  glVertex2f(-8.2, -19.8);  glVertex2f(-8.2, -19.8);  glVertex2f(-13, -19.2);  glVertex2f(-13.2, -17.4);  glVertex2f(-13.1, -16.8);  glVertex2f(-12.8, -15.8);  glVertex2f(-4, -14);  glVertex2f(-2, -10);  glVertex2f(10, -10);  glVertex2f(16.75, -15.95);  glVertex2f(16, -20);  glVertex2f(14.414494, -20.0239526);  glEnd();  //black sqaure  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(15, -16);  glVertex2f(16, -16);  glVertex2f(16, -17);  glVertex2f(15, -17);  glEnd();  //window 1  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(0, -11);  glVertex2f(-2, -15);  glVertex2f(4, -15);  glVertex2f(4, -11);  glEnd();  //window 2 square  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(5, -11);  glVertex2f(5, -15);  glVertex2f(10, -15);  glVertex2f(10, -11);  glEnd();  //wheel 1 in between  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0.3,0.3,0.3);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.88844;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-5.3779922367174, -19.2035973808484,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(5.3779922367174, 19.2035973808484,0.0);  //wheel\_1  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<360;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.3809527;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484 );  }  //glVertex2f(0.3f,0.4f);  //glVertex2f(0.1f,0.4f);  glEnd();  //wheel 1 outline  glBegin(GL\_POINTS);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.88844;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  //wheel 1 white circle  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,1,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.6118059056;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  // wheel 1 five white small circles  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 6.3,y - 18.3);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,1,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 6.6816,y - 19.76441);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,0,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.13779,y - 20.4809);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,1,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 4,y - 19.5);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,1,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 4.53203,y - 18.1760);  }  glEnd();  // wheel 1 white lines  glLineWidth(1);  glColor3f(1, 1, 1);  glBegin(GL\_LINES);  glVertex2f(-5.377995, -18.59179);  glVertex2f(-5.259289, -16.8256057);  glVertex2f(-5.988550077, -19.1645385906856);  glVertex2f(-7.75837164792479, -19.1515827274553);  glVertex2f(-5.48297359, -19.806328979);  glVertex2f(-6.144128, -21.443123);  glVertex2f(-4.990972387, -19.6774350135);  glVertex2f(-3.9308820937866, -21.0943133);  glVertex2f(-4.773882291142, -19.106828738443);  glVertex2f(-3.0554403012639, -18.67949131);  glEnd();  glPopMatrix();  //glutSwapBuffers();  /\*  //wheel 2 in between  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0.3,0.3,0.3);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.730296;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x + 11.7782074159104,y - 19.3136665772639);  }  glEnd();  //wheel 2  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<360;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.184420;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x + 11.7782074159104,y - 19.3136665772639);  }  //glVertex2f(0.3f,0.4f);  //glVertex2f(0.1f,0.4f);  glEnd();  //wheel 2 outline  glBegin(GL\_POINTS);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.730296;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x + 11.7782074159104,y - 19.3136665772639);  }  glEnd();  \*/  //wheel\_2  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(17, 0.0f, 0.0f);  //wheel 2 in between  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0.3,0.3,0.3);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.88844;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  glTranslatef(-5.3779922367174, -19.2035973808484,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(5.3779922367174, 19.2035973808484,0.0);  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<360;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.3809527;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484 );  }  //glVertex2f(0.3f,0.4f);  //glVertex2f(0.1f,0.4f);  glEnd();  //wheel 1 outline  glBegin(GL\_POINTS);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.88844;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  //wheel 1 white circle  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,1,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.6118059056;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  // wheel 1 five white small circles  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 6.3,y - 18.3);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,1,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 6.6816,y - 19.76441);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,0,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.13779,y - 20.4809);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,1,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 4,y - 19.5);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,1,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 4.53203,y - 18.1760);  }  glEnd();  // wheel 1 white lines  glLineWidth(1);  glColor3f(1, 1, 1);  glBegin(GL\_LINES);  glVertex2f(-5.377995, -18.59179);  glVertex2f(-5.259289, -16.8256057);  glVertex2f(-5.988550077, -19.1645385906856);  glVertex2f(-7.75837164792479, -19.1515827274553);  glVertex2f(-5.48297359, -19.806328979);  glVertex2f(-6.144128, -21.443123);  glVertex2f(-4.990972387, -19.6774350135);  glVertex2f(-3.9308820937866, -21.0943133);  glVertex2f(-4.773882291142, -19.106828738443);  glVertex2f(-3.0554403012639, -18.67949131);  glEnd();  glPopMatrix();  glPopMatrix();  glutSwapBuffers();  }  void update(int value) {  \_move -= .5;  if(\_move < -86.75)  {  \_move = 0.0;  }  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  void update1(int value) {  \_angle1+=5.0f;  if(\_angle1 > 360.0)  {  \_angle1-=360;  }  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(20, update1, 0); //Notify GLUT to call update again in 25 milliseconds  }  void sky()  {  //cloud 2  glBegin(GL\_POLYGON);  glColor3f(1, 1, 1);  glVertex2f(33.8, 19.6);  glVertex2f(33.8, 16.8);  glVertex2f(38.2, 16.8);  glVertex2f(40.6, 18);  glVertex2f(40.6, 20);  glVertex2f(39.4, 21.6);  glVertex2f(36.4, 21.8);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(20, 20);  glVertex2f(20, 16.8);  glVertex2f(33.8, 16.8);  glVertex2f(33.8, 19.6);  glVertex2f(31.2, 21.3);  glVertex2f(28.2, 22.8);  glVertex2f(23.4, 22.8);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(20, 20);  glVertex2f(18.4, 21.4);  glVertex2f(15, 22.8);  glVertex2f(12.8, 21.6);  glVertex2f(12.8, 18.8);  glVertex2f(15.4, 16.8);  glVertex2f(20.6, 16.8);  glEnd();  //cloud 1  glBegin(GL\_POLYGON);  glVertex2f(-23, 20.5);  glVertex2f(-25, 22);  glVertex2f(-30.5, 23);  glVertex2f(-35, 21);  glVertex2f(-34, 18);  glVertex2f(-30, 17);  glVertex2f(-23, 16.5);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-23, 20.5);  glVertex2f(-23, 16.5);  glVertex2f(-18, 17);  glVertex2f(-14, 19);  glVertex2f(-14.5, 21.5);  glVertex2f(-16.5, 23);  glVertex2f(-20.5, 22);  glVertex2f(-20.5, 22);  glEnd();  //birds  glLineWidth(4);  glColor3f(0,0,0);  glBegin(GL\_LINES);  glVertex2f(-6, 22.6);  glVertex2f(-5.2, 23);  glVertex2f(-5.2, 23);  glVertex2f(-4.4, 22.6);  glVertex2f(-4.4, 22.6);  glVertex2f(-3.6, 21.8);  glVertex2f(-3.6, 21.8);  glVertex2f(-2.9, 20.8);  glVertex2f(-2.9, 20.8);  glVertex2f(-2.4, 21.8);  glVertex2f(-2.4, 21.8);  glVertex2f(-1.6, 22.6);  glVertex2f(-1.6, 22.6);  glVertex2f(-0.8, 23);  glVertex2f(-0.8, 23);  glVertex2f(0 ,22.6);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-2, 20.6);  glVertex2f(-1.2, 21);  glVertex2f(-1.2, 21);  glVertex2f(-0.4, 20.6);  glVertex2f(-0.4, 20.6);  glVertex2f(0.4, 19.8);  glVertex2f(0.4, 19.8);  glVertex2f(1.1, 18.8);  glVertex2f(1.1, 18.8);  glVertex2f(1.6, 19.8);  glVertex2f(1.6, 19.8);  glVertex2f(2.4, 20.6);  glVertex2f(2.4, 20.6);  glVertex2f(3.2, 21);  glVertex2f(3.2, 21);  glVertex2f(4, 20.6);  glEnd();  glBegin(GL\_LINES);  glVertex2f(1, 23.6);  glVertex2f(1.8, 24);  glVertex2f(1.8, 24);  glVertex2f(2.6, 23.6);  glVertex2f(2.6, 23.6);  glVertex2f(3.4, 22.8);  glVertex2f(3.4, 22.8);  glVertex2f(4.1, 21.8);  glVertex2f(4.1, 21.8);  glVertex2f(4.6, 22.8);  glVertex2f(4.6, 22.8);  glVertex2f(5.4, 23.6);  glVertex2f(5.4, 23.6);  glVertex2f(6.2, 24);  glVertex2f(6.2, 24);  glVertex2f(7, 23.6);  glEnd();  }  void tree(){  glColor3f(0,0,0);  glBegin(GL\_POLYGON);  glVertex2f(24, -4);  glVertex2f(24, -14.037031);  glVertex2f(26, -14.037031);  glVertex2f(26, -4);  glEnd();  glColor3f(0,0.4,0);  glBegin(GL\_POLYGON);  glVertex2f(25, 3);  glVertex2f(16, -4);  glVertex2f(34, -4);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(18, 0);  glVertex2f(32, 0);  glVertex2f(25,5.5);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(20, 3);  glVertex2f(30, 3);  glVertex2f(25,8);  glEnd();  }  void display() {  glClearColor(0.0f, 0.6f,0.90f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)  back\_jungles();  traffic\_poll();  three\_circle( 1.18558802,-35.2276891476581, 10.3308657487617,1, 0, 0);  three\_circle( 1.18558802,-35.2276891476581, 7.3308657,1, 1, 0);  three\_circle( 1.18558802,-35.2276891476581, 4.0308657,0, 1, 0);  road();  tree();  car();  sky();  glFlush(); // Render now  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInitWindowSize(1320, 820); // Set the window's initial width & height  glutInit(&argc, argv); // Initialize GLUT  glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title  glutDisplayFunc(display); // Register display callback handler for window re-paint  gluOrtho2D(-40, +45, -30, 25);  glutTimerFunc(20, update, 0); //Add a timer  glutTimerFunc(20, update1, 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)-**  \*\* used rotatef function in the code to rotate the car and move from left to right |
| **Code-**  #include <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  #include <math.h>  /\* Handler for window-repaint event. Call back when the window first appears and  whenever the window needs to be re-painted. \*/  float \_move = 0.0f;  float \_angle1 = 0.0f;  void back\_jungles(){  //extreme back jungles  glBegin(GL\_POLYGON);  glColor3f(0.1, 0.6, 0);  glVertex2f(-40, -16);  glVertex2f(-40, 2.2);  glVertex2f(-38, 4);  glVertex2f(-32.5, 5.98);  glVertex2f(-17.6, 11.2);  glVertex2f(4, 15);  glVertex2f(29, 11.5);  glVertex2f(45, -1);  glVertex2f(45, -14);  glEnd();  // back small jungles  glBegin(GL\_POLYGON);  glColor3f(0, 0.9, 0);  glVertex2f(-40, -10);  glVertex2f(-38, -8);  glVertex2f(-36, -6.706921);  glVertex2f(-32.9358216, -6.706921);  glVertex2f(-31, -8);  glVertex2f(-29, -10);  glVertex2f(-27, -6);  glVertex2f(-25, -3);  glVertex2f(-21, -2);  glVertex2f(-19, -3);  glVertex2f(-17, -6);  glVertex2f(-14, -10);  glVertex2f(-12, -11);  glVertex2f(-11, -9);  glVertex2f(-8, -9);  glVertex2f(-7, -10);  glVertex2f(-6, -8);  glVertex2f(-5, -5);  glVertex2f(-2,-4);  glVertex2f(0,-4);  glVertex2f(3,-7);  glVertex2f(4.5,-7.5);  glVertex2f(5,-8.5);  glVertex2f(10,-5);  glVertex2f(14,-4.5);  glVertex2f(16,-5.5);  glVertex2f(17.5,-7.5);  glVertex2f(19,-10.5);  glVertex2f(21.5,-8.5);  glVertex2f(25,-7);  glVertex2f(28.5,-6.5);  glVertex2f(32.5,-8);  glVertex2f(34.5,-11.5);  glVertex2f(36.5,-13);  glVertex2f(38.5,-10);  glVertex2f(41,-8.5);  glVertex2f(45,-8);  glVertex2f(45, -16);  glVertex2f(-40, -16);  glEnd();  }  void traffic\_poll(){  //black area under traffic light poll  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(-37, -14);  glVertex2f(-37, -16);  glVertex2f(45, -16);  glVertex2f(45, -14);  glEnd();  // light poll -- black area  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(-36, 2);  glVertex2f(-36, -14);  glVertex2f(-34.5, -14);  glVertex2f(-34.5, 2);  glEnd();  // light gray area of poll  glBegin(GL\_POLYGON);  glColor3f(0.3, 0.3, 0.3);  glVertex2f(-38, 12);  glVertex2f(-38, 2);  glVertex2f(-32.5, 2);  glVertex2f(-32.5, 12);  glEnd();  }  void three\_circle(float radius, float xc, float yc, float r, float g, float b)  {  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  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 road(){  //road  glBegin(GL\_POLYGON);  glColor3f(0.3, 0.3, 0.3);  glVertex2f(-40, -16);  glVertex2f(-40, -30);  glVertex2f(45, -30);  glVertex2f(45, -16);  glEnd();  //white zebra crossing signs  //road  glColor3f(1, 1, 1);  glBegin(GL\_POLYGON);  glVertex2f(-36, -16.7);  glVertex2f(-36.5, -18.2);  glVertex2f(-25.5, -18.2);  glVertex2f(-25, -16.7);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-36.5, -19);  glVertex2f(-37, -20.5);  glVertex2f(-26, -20.5);  glVertex2f(-25.5, -19);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-37, -22);  glVertex2f(-37.5, -23.5);  glVertex2f(-26.5, -23.5);  glVertex2f(-26, -22);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-37.5, -25);  glVertex2f(-38, -26.5);  glVertex2f(-27, -26.5);  glVertex2f(-26.5, -25);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-38, -28);  glVertex2f(-38.5, -29.5);  glVertex2f(-27.5, -29.5);  glVertex2f(-27, -28);  glEnd();  //two under the car  glBegin(GL\_POLYGON);  glVertex2f(-2, -22);  glVertex2f(-3, -23);  glVertex2f(5, -23);  glVertex2f(6, -22);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(16, -22);  glVertex2f(15, -23);  glVertex2f(24, -23);  glVertex2f(24.8, -22);  glEnd();  }  void car(){  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-30, 0.0f, 0.0f);  glRotatef(180,0.0f, 1.0f, 0.0f);  glTranslatef(\_move, 0.0f, 0.0f);  //red body  glBegin(GL\_POLYGON);  glColor3f(1, 0, 0);  glVertex2f(-8.2, -19.8);  glVertex2f(-8.2, -19.8);  glVertex2f(-13, -19.2);  glVertex2f(-13.2, -17.4);  glVertex2f(-13.1, -16.8);  glVertex2f(-12.8, -15.8);  glVertex2f(-4, -14);  glVertex2f(-2, -10);  glVertex2f(10, -10);  glVertex2f(16.75, -15.95);  glVertex2f(16, -20);  glVertex2f(14.414494, -20.0239526);  glEnd();  //black sqaure  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(15, -16);  glVertex2f(16, -16);  glVertex2f(16, -17);  glVertex2f(15, -17);  glEnd();  //window 1  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(0, -11);  glVertex2f(-2, -15);  glVertex2f(4, -15);  glVertex2f(4, -11);  glEnd();  //window 2 square  glBegin(GL\_POLYGON);  glColor3f(0, 0, 0);  glVertex2f(5, -11);  glVertex2f(5, -15);  glVertex2f(10, -15);  glVertex2f(10, -11);  glEnd();  //wheel 1 in between  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0.3,0.3,0.3);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.88844;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-5.3779922367174, -19.2035973808484,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(5.3779922367174, 19.2035973808484,0.0);  //wheel\_1  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<360;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.3809527;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484 );  }  //glVertex2f(0.3f,0.4f);  //glVertex2f(0.1f,0.4f);  glEnd();  //wheel 1 outline  glBegin(GL\_POINTS);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.88844;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  //wheel 1 white circle  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,1,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.6118059056;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  // wheel 1 five white small circles  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 6.3,y - 18.3);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,1,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 6.6816,y - 19.76441);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,0,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.13779,y - 20.4809);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,1,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 4,y - 19.5);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,1,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 4.53203,y - 18.1760);  }  glEnd();  // wheel 1 white lines  glLineWidth(1);  glColor3f(1, 1, 1);  glBegin(GL\_LINES);  glVertex2f(-5.377995, -18.59179);  glVertex2f(-5.259289, -16.8256057);  glVertex2f(-5.988550077, -19.1645385906856);  glVertex2f(-7.75837164792479, -19.1515827274553);  glVertex2f(-5.48297359, -19.806328979);  glVertex2f(-6.144128, -21.443123);  glVertex2f(-4.990972387, -19.6774350135);  glVertex2f(-3.9308820937866, -21.0943133);  glVertex2f(-4.773882291142, -19.106828738443);  glVertex2f(-3.0554403012639, -18.67949131);  glEnd();  glPopMatrix();  //glutSwapBuffers();  /\*  //wheel 2 in between  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0.3,0.3,0.3);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.730296;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x + 11.7782074159104,y - 19.3136665772639);  }  glEnd();  //wheel 2  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<360;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.184420;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x + 11.7782074159104,y - 19.3136665772639);  }  //glVertex2f(0.3f,0.4f);  //glVertex2f(0.1f,0.4f);  glEnd();  //wheel 2 outline  glBegin(GL\_POINTS);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.730296;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x + 11.7782074159104,y - 19.3136665772639);  }  glEnd();  \*/  //wheel\_2  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(17, 0.0f, 0.0f);  //wheel 2 in between  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0.3,0.3,0.3);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.88844;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  glTranslatef(-5.3779922367174, -19.2035973808484,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(5.3779922367174, 19.2035973808484,0.0);  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<360;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.3809527;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484 );  }  //glVertex2f(0.3f,0.4f);  //glVertex2f(0.1f,0.4f);  glEnd();  //wheel 1 outline  glBegin(GL\_POINTS);  glLineWidth(2.5);  for(int i=0;i<560;i++)  {  glColor3f(0,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 2.88844;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  //wheel 1 white circle  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,1,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.6118059056;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.3779922367174,y - 19.2035973808484);  }  glEnd();  // wheel 1 five white small circles  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,0,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 6.3,y - 18.3);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,1,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 6.6816,y - 19.76441);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,0,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 5.13779,y - 20.4809);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(0,1,1);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 4,y - 19.5);  }  glEnd();  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,1,0);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.330;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 4.53203,y - 18.1760);  }  glEnd();  // wheel 1 white lines  glLineWidth(1);  glColor3f(1, 1, 1);  glBegin(GL\_LINES);  glVertex2f(-5.377995, -18.59179);  glVertex2f(-5.259289, -16.8256057);  glVertex2f(-5.988550077, -19.1645385906856);  glVertex2f(-7.75837164792479, -19.1515827274553);  glVertex2f(-5.48297359, -19.806328979);  glVertex2f(-6.144128, -21.443123);  glVertex2f(-4.990972387, -19.6774350135);  glVertex2f(-3.9308820937866, -21.0943133);  glVertex2f(-4.773882291142, -19.106828738443);  glVertex2f(-3.0554403012639, -18.67949131);  glEnd();  glPopMatrix();  glPopMatrix();  glutSwapBuffers();  }  void update(int value) {  \_move -= .5;  if(\_move < -86.75)  {  \_move = 0.0;  }  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  void update1(int value) {  \_angle1+=5.0f;  if(\_angle1 > 360.0)  {  \_angle1-=360;  }  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(20, update1, 0); //Notify GLUT to call update again in 25 milliseconds  }  void sky()  {  //cloud 2  glBegin(GL\_POLYGON);  glColor3f(1, 1, 1);  glVertex2f(33.8, 19.6);  glVertex2f(33.8, 16.8);  glVertex2f(38.2, 16.8);  glVertex2f(40.6, 18);  glVertex2f(40.6, 20);  glVertex2f(39.4, 21.6);  glVertex2f(36.4, 21.8);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(20, 20);  glVertex2f(20, 16.8);  glVertex2f(33.8, 16.8);  glVertex2f(33.8, 19.6);  glVertex2f(31.2, 21.3);  glVertex2f(28.2, 22.8);  glVertex2f(23.4, 22.8);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(20, 20);  glVertex2f(18.4, 21.4);  glVertex2f(15, 22.8);  glVertex2f(12.8, 21.6);  glVertex2f(12.8, 18.8);  glVertex2f(15.4, 16.8);  glVertex2f(20.6, 16.8);  glEnd();  //cloud 1  glBegin(GL\_POLYGON);  glVertex2f(-23, 20.5);  glVertex2f(-25, 22);  glVertex2f(-30.5, 23);  glVertex2f(-35, 21);  glVertex2f(-34, 18);  glVertex2f(-30, 17);  glVertex2f(-23, 16.5);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-23, 20.5);  glVertex2f(-23, 16.5);  glVertex2f(-18, 17);  glVertex2f(-14, 19);  glVertex2f(-14.5, 21.5);  glVertex2f(-16.5, 23);  glVertex2f(-20.5, 22);  glVertex2f(-20.5, 22);  glEnd();  //birds  glLineWidth(4);  glColor3f(0,0,0);  glBegin(GL\_LINES);  glVertex2f(-6, 22.6);  glVertex2f(-5.2, 23);  glVertex2f(-5.2, 23);  glVertex2f(-4.4, 22.6);  glVertex2f(-4.4, 22.6);  glVertex2f(-3.6, 21.8);  glVertex2f(-3.6, 21.8);  glVertex2f(-2.9, 20.8);  glVertex2f(-2.9, 20.8);  glVertex2f(-2.4, 21.8);  glVertex2f(-2.4, 21.8);  glVertex2f(-1.6, 22.6);  glVertex2f(-1.6, 22.6);  glVertex2f(-0.8, 23);  glVertex2f(-0.8, 23);  glVertex2f(0 ,22.6);  glEnd();  glBegin(GL\_LINES);  glVertex2f(-2, 20.6);  glVertex2f(-1.2, 21);  glVertex2f(-1.2, 21);  glVertex2f(-0.4, 20.6);  glVertex2f(-0.4, 20.6);  glVertex2f(0.4, 19.8);  glVertex2f(0.4, 19.8);  glVertex2f(1.1, 18.8);  glVertex2f(1.1, 18.8);  glVertex2f(1.6, 19.8);  glVertex2f(1.6, 19.8);  glVertex2f(2.4, 20.6);  glVertex2f(2.4, 20.6);  glVertex2f(3.2, 21);  glVertex2f(3.2, 21);  glVertex2f(4, 20.6);  glEnd();  glBegin(GL\_LINES);  glVertex2f(1, 23.6);  glVertex2f(1.8, 24);  glVertex2f(1.8, 24);  glVertex2f(2.6, 23.6);  glVertex2f(2.6, 23.6);  glVertex2f(3.4, 22.8);  glVertex2f(3.4, 22.8);  glVertex2f(4.1, 21.8);  glVertex2f(4.1, 21.8);  glVertex2f(4.6, 22.8);  glVertex2f(4.6, 22.8);  glVertex2f(5.4, 23.6);  glVertex2f(5.4, 23.6);  glVertex2f(6.2, 24);  glVertex2f(6.2, 24);  glVertex2f(7, 23.6);  glEnd();  }  void tree(){  glColor3f(0,0,0);  glBegin(GL\_POLYGON);  glVertex2f(24, -4);  glVertex2f(24, -14.037031);  glVertex2f(26, -14.037031);  glVertex2f(26, -4);  glEnd();  glColor3f(0,0.4,0);  glBegin(GL\_POLYGON);  glVertex2f(25, 3);  glVertex2f(16, -4);  glVertex2f(34, -4);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(18, 0);  glVertex2f(32, 0);  glVertex2f(25,5.5);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(20, 3);  glVertex2f(30, 3);  glVertex2f(25,8);  glEnd();  }  void display() {  glClearColor(0.0f, 0.6f,0.90f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)  back\_jungles();  traffic\_poll();  three\_circle( 1.18558802,-35.2276891476581, 10.3308657487617,1, 0, 0);  three\_circle( 1.18558802,-35.2276891476581, 7.3308657,1, 1, 0);  three\_circle( 1.18558802,-35.2276891476581, 4.0308657,0, 1, 0);  road();  tree();  car();  sky();  glFlush(); // Render now  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInitWindowSize(1320, 820); // Set the window's initial width & height  glutInit(&argc, argv); // Initialize GLUT  glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title  glutDisplayFunc(display); // Register display callback handler for window re-paint  gluOrtho2D(-40, +45, -30, 25);  glutTimerFunc(20, update, 0); //Add a timer  glutTimerFunc(20, update1, 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 <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  #include <math.h>  /\* Handler for window-repaint event. Call back when the window first appears and  whenever the window needs to be re-painted. \*/  float \_angle1 = 0.0f;  void windmill(){  // triangle  glBegin(GL\_POLYGON);  glColor3f(1,0,0);  glVertex2f(0, 7);  glVertex2f(-4, 1);  glVertex2f(4, 1);  glEnd();  // lower portion part  glBegin(GL\_POLYGON);  glColor3f(1,1,0);  glVertex2f(-2.9710578, 1);  glVertex2f(-6.264816429, -10.01450359);  glVertex2f(6, -10);  glVertex2f(2.80555117, 1);  glEnd();  // circle  glBegin(GL\_POLYGON);  glLineWidth(2.5);  for(int i=0;i<360;i++)  {  glColor3f(1,1,0.99);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r= 0.464578;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x - 0.0739689526722,y + 1.7006424408778);  }  glEnd();  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(-0.0739689526722, +1.7006424408778,0.0);  glRotatef(\_angle1, 0.0f, 0.0f,1.0f);  glTranslatef(0.0739689526722, -1.7006424408778,0.0);  // blades  glBegin(GL\_POLYGON);  glColor3f(1,1,1);  glVertex2f(0.46, 2.64);  glVertex2f(0.089986, 2.13532866);  glVertex2f(0.18615530, 2.085569);  glVertex2f(0.56, 2.56);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(1,1,1);  glVertex2f(0.6216717902193, 2.5048083088078);  glVertex2f(4.2765387485632, 6.5467810761083);  glVertex2f(3.2909042069108, 7.4843358840215);  glVertex2f(0.3906217052614, 2.7057214261625);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(1,1,1);  glVertex2f(-0.98, 2.26);  glVertex2f(-0.4905033, 1.906391);  glVertex2f(-0.44368376, 1.9819684);  glVertex2f(-0.9, 2.34);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(1,1,1);  glVertex2f(-0.8084914619834, 2.4370793666386);  glVertex2f(-4.8419376762044, 6.4907295343387);  glVertex2f(-5.7912310893285, 5.5414361212145);  glVertex2f(-1.1142996212191, 2.1253331848934);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(1,1,1);  glVertex2f(-0.78, 0.86);  glVertex2f(-0.3260878, 1.310425);  glVertex2f(-0.40942731, 1.3792376);  glVertex2f(-0.88, 0.94);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(1,1,1);  glVertex2f(-0.9939943291765, 1.0271021476365);  glVertex2f(-5.5, -2.5);  glVertex2f(-4.5, -3.5);  glVertex2f(-0.6660745445086, 0.7702524303681);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(1,1,1);  glVertex2f(0.74, 0.87);  glVertex2f(0.82, 0.98);  glVertex2f(0.28616292, 1.40714899);  glVertex2f(0.201014167, 1.3261859655);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(1,1,1);  glVertex2f(0.55, 0.6);  glVertex2f(5.4, -3.2);  glVertex2f(6.2, -2);  glVertex2f(1, 1.2);  glEnd();  glFlush;  glPopMatrix();  }  void update1(int value) {  \_angle1+=3.0f;  if(\_angle1 > 360.0)  {  \_angle1-=360;  }  glutPostRedisplay(); //Notify GLUT that the display has changed  glutTimerFunc(20, update1, 0); //Notify GLUT to call update again in 25 milliseconds  }  void display() {  glClearColor(0.0f, 0.0f,0.0f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)  windmill();  glFlush(); // Render now  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInitWindowSize(800, 800); // Set the window's initial width & height  glutInit(&argc, argv); // Initialize GLUT  glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title  glutDisplayFunc(display); // Register display callback handler for window re-paint  gluOrtho2D(-8, +8, -12, +9);  glutTimerFunc(20, update1, 0); //Add a timer  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-** |