**Lab Taks-4**

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 time that will be discussed in class VUES to the section named Lab Tak-4
* Must include resources for all the section in the table

|  |
| --- |
| **Question- 1**  Draw the scenario of a traffic signal using function to represent each object |
| **Graph Plot (Picture)-** |
| **Code-**  #include <windows.h>  #include <GL/glut.h>  #include <math.h>  void circle(float radius, float cX, float cY, 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+cX,y+cY);  }  glEnd();  }  void Road()  {  //  //Road  //  glBegin(GL\_POLYGON);  glColor3f(0.33f, 0.33f, 0.33f);  glVertex2f(-24.0f, 8.0f);  glVertex2f(-24.0f, -8.0f);  glVertex2f(24.0f, -8.0f);  glVertex2f(24.0f, 8.0f);  glEnd();  glBegin(GL\_LINES);  glColor3f(1.0f, 1.0f, 1.0f);  glVertex2f(-22.0f, 0.0f);  glVertex2f(-17.0f, 0.0f);  glEnd();  glBegin(GL\_LINES);  glColor3f(1.0f, 1.0f, 1.0f);  glVertex2f(-15.0f, 0.0f);  glVertex2f(-10.0f, 0.0f);  glEnd();  glBegin(GL\_LINES);  glColor3f(1.0f, 1.0f, 1.0f);  glVertex2f(-8.0f, 0.0f);  glVertex2f(-3.0f, 0.0f);  glEnd();  glBegin(GL\_LINES);  glColor3f(1.0f, 1.0f, 1.0f);  glVertex2f(-1.0f, 0.0f);  glVertex2f(4.0f, 0.0f);  glEnd();  glBegin(GL\_LINES);  glColor3f(1.0f, 1.0f, 1.0f);  glVertex2f(6.0f, 0.0f);  glVertex2f(11.0f, 0.0f);  glEnd();  glBegin(GL\_LINES);  glColor3f(1.0f, 1.0f, 1.0f);  glVertex2f(13.0f, 0.0f);  glVertex2f(18.0f, 0.0f);  glEnd();  glBegin(GL\_LINES);  glColor3f(1.0f, 1.0f, 1.0f);  glVertex2f(20.0f, 0.0f);  glVertex2f(24.0f, 0.0f);  glEnd();  //  //background  //  glBegin(GL\_POLYGON);  glColor3f(0.122f, 0.647f, 0.871f);  glVertex2f(-24.0f, 8.0f);  glVertex2f(24.0f, 8.0f);  glVertex2f(24.0f, 24.0f);  glVertex2f(-24.0f, 24.0f);  glEnd();  }  void TrafficLight()  {  //  //Traffic light stand  //  glBegin(GL\_POLYGON);  glColor3f(0.72f, 0.71f, 0.70f);  glVertex2f(18.0f, 8.0f);  glVertex2f(20.0f, 8.0f);  glVertex2f(20.0f, 16.0f);  glVertex2f(18.0f, 16.0f);  glEnd();  //  //Traffic lightbox  //  glBegin(GL\_POLYGON);  glColor3f(0.72f, 0.71f, 0.70f);  glVertex2f(17.0f, 16.0f);  glVertex2f(21.0f, 16.0f);  glVertex2f(21.0f, 23.0f);  glVertex2f(17.0f, 23.0f);  glEnd();  circle(0.7,19,18,0.0,1.0,0.0);  circle(0.7,19,20,1.0,1.0,0.0);  circle(0.7,19,22,1.0,0.0,0.0);  }  void Car()  {  //  //Car  //  glBegin(GL\_POLYGON);  glColor3f(0.796f, 0.8f, 0.216f);  glVertex2f(-22.0f, 8.0f);  glVertex2f(-22.0f, 4.0f);  glVertex2f(-2.0f, 4.0f);  glVertex2f(-2.0f, 8.0f);  glVertex2f(-8.0f, 8.0f);  glVertex2f(-12.0f, 11.0f);  glVertex2f(-19.0f, 11.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.0f, 0.0f, 0.0f);  glVertex2f(-16.0f, 10.0f);  glVertex2f(-19.0f, 10.0f);  glVertex2f(-21.0f, 8.0f);  glVertex2f(-16.0f, 8.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.0f, 0.0f, 0.0f);  glVertex2f(-15.0f, 10.0f);  glVertex2f(-12.0f, 10.0f);  glVertex2f(-9.0f, 8.0f);  glVertex2f(-15.0f, 8.0f);  glEnd();  circle(1.5,-18,4,0.0,0.0,0.0);  circle(0.7,-18,4,0.639,0.659,0.659);  circle(1.5,-6,4,0.0,0.0,0.0);  circle(0.7,-6,4,0.639,0.659,0.659);  }  void Bus()  {  //  //Bus  //  glBegin(GL\_POLYGON);  glColor3f(0.169f, 0.872f, 0.084f);  glVertex2f(1.0f, 13.0f);  glVertex2f(1.0f, 8.0f);  glVertex2f(17.0f, 8.0f);  glVertex2f(15.0f, 13.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.169f, 0.872f, 0.084f);  glVertex2f(1.0f, 8.0f);  glVertex2f(1.0f, 4.0f);  glVertex2f(17.0f, 4.0f);  glVertex2f(17.0f, 8.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.0f, 0.0f, 0.0f);  glVertex2f(2.0f, 10.0f);  glVertex2f(4.0f, 10.0f);  glVertex2f(4.0f, 12.0f);  glVertex2f(2.0f, 12.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.0f, 0.0f, 0.0f);  glVertex2f(5.0f, 10.0f);  glVertex2f(7.0f, 10.0f);  glVertex2f(7.0f, 12.0f);  glVertex2f(5.0f, 12.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.0f, 0.0f, 0.0f);  glVertex2f(8.0f, 10.0f);  glVertex2f(10.0f, 10.0f);  glVertex2f(10.0f, 12.0f);  glVertex2f(8.0f, 12.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.0f, 0.0f, 0.0f);  glVertex2f(11.0f, 10.0f);  glVertex2f(13.0f, 10.0f);  glVertex2f(13.0f, 12.0f);  glVertex2f(11.0f, 12.0f);  glEnd();  circle(1.5,4,4,0.0,0.0,0.0);  circle(0.7,4,4,0.639,0.659,0.659);  circle(1.5,12,4,0.0,0.0,0.0);  circle(0.7,12,4,0.639,0.659,0.659);  glBegin(GL\_POLYGON);  glColor3f(0.988f, 0.988f, 0.8f);  glVertex2f(1.0f, 7.0f);  glVertex2f(17.0f, 7.0f);  glVertex2f(17.0f, 8.0f);  glVertex2f(1.0f, 8.0f);  glEnd();  }  void display() {  glClearColor(0.0f, 0.0f, 0.0f, 0.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  glLineWidth(3.5);  Road();  TrafficLight();  Car();  Bus();  glFlush();  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInit(&argc, argv); // Initialize GLUT  glutCreateWindow("Vertex, Primitive & Color"); // Create window with the given title  glutInitWindowSize(720, 720); // Set the window's initial width & height  glutInitWindowPosition(20, 20); // Position the window's initial top-left corner  glutDisplayFunc(display); // Register callback handler for window re-paint event  gluOrtho2D(-30,30,-10,30);  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question- 2**  Draw two village scenarios for day and night using function to represent each object |
| **Graph Plot (Picture)-** |
| **Code-**  #include <windows.h>  #include <GL/glut.h>  #include <math.h>  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 display() {  glClearColor(0.0f, 0.0f, 0.0f, 0.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  glLineWidth(3.5);  //  //sky  //  glBegin(GL\_POLYGON);  glColor3f(0.0f, 0.0f, 0.15f);  glVertex2f(-25.0f, 15.0f);  glVertex2f(10.0, 15.0f);  glVertex2f(10.0f, 0.0f);  glVertex2f(-25.0f, 0.0f);  glEnd();  //  //moon  //  circle(1.7592, 4.0, 12.0, 1.0,1.0,1.0);  //  //background trees  //  circle(1.0, -24.0, 0.0, 0.01,0.25,0.13);  circle(1.0, -22.0, 0.0, 0.01,0.25,0.13);  circle(1.0, -20.0, 0.0, 0.01,0.25,0.13);  circle(1.0, -18.0, 0.0, 0.01,0.25,0.13);  circle(1.0, -16.0, 0.0, 0.01,0.25,0.13);  circle(1.0, -14.0, 0.0, 0.01,0.25,0.13);  circle(1.0, -12.0, 0.0, 0.01,0.25,0.13);  circle(1.0, -10.0, 0.0, 0.01,0.25,0.13);  circle(1.0, 0.0, 0.0, 0.01,0.25,0.13);  circle(1.0, 2.0, 0.0, 0.01,0.25,0.13);  circle(1.0, 4.0, 0.0, 0.01,0.25,0.13);  circle(1.0, 6.0, 0.0, 0.01,0.25,0.13);  circle(1.0, 8.0, 0.0, 0.01,0.25,0.13);  circle(1.0, 10.0, 0.0, 0.01,0.25,0.13);  circle(1.51329, 7.0, 1.0, 0.01,0.25,0.13);  circle(1.51329, 3.0, 1.0, 0.01,0.25,0.13);  circle(1.51329, -1.5, 1.0, 0.01,0.25,0.13);  circle(1.51329, -20.0, 1.0, 0.01,0.25,0.13);  circle(1.51329, -23.5, 1.0, 0.01,0.25,0.13);  //  //grass  //  glBegin(GL\_POLYGON);  glColor3f(0.2f, 0.4, 0.18f);  glVertex2f(-25.0f, -8.0f);  glVertex2f(10.0, -8.0f);  glVertex2f(10.0f, 0.0f);  glVertex2f(-25.0f, 0.0f);  glEnd();  //  //tree  //  glBegin(GL\_POLYGON);  glColor3f(0.678f, 0.460f, 0.082f);  glVertex2f(-14.0f, -3.0f);  glVertex2f(-12.0, -3.0f);  glVertex2f(-12.0f, 3.0f);  glVertex2f(-14.0f, 3.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.678f, 0.460f, 0.082f);  glVertex2f(-14.0f, -3.0f);  glVertex2f(-15.0, -4.0f);  glVertex2f(-13.6f, -3.3f);  glVertex2f(-13.0f, -4.0f);  glVertex2f(-12.4f, -3.3f);  glVertex2f(-11.0f, -4.0f);  glVertex2f(-12.0f, -3.0f);  glEnd();  circle(3.3801, -16.0, 6.0, 0.0,0.5,0.0);  circle(4.1227, -13.0, 7.5, 0.0,0.5,0.0);  circle(3.3844, -10.0, 6.0, 0.0,0.5,0.0);  circle(3.5737, -12.0, 4.0, 0.0,0.5,0.0);  circle(3.5737, -14.0, 4.0, 0.0,0.5,0.0);  circle(1.4889, -13.0, 1.0, 0.0,0.5,0.0);  //  //straw  //  glBegin(GL\_POLYGON);  glColor3f(0.75f, 0.75, 0.1f);  glVertex2f(3.0f, -3.0f);  glVertex2f(0.0, -3.0f);  glVertex2f(0.0f, -1.0f);  glVertex2f(0.4f, 0.4f);  glVertex2f(1.4f, 1.0f);  glVertex2f(1.6f, 1.0f);  glVertex2f(2.6f, 0.4f);  glVertex2f(3.0f, -1.0f);  glEnd();  //  //house  //  glBegin(GL\_POLYGON);  glColor3f(0.849f, 0.478f, 0.254f);  glVertex2f(-6.0f, -3.0f);  glVertex2f(-1.0f, -3.0f);  glVertex2f(-1.0f, 1.0f);  glVertex2f(-6.0f, 3.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.849f, 0.478f, 0.254f);  glVertex2f(-6.0f, -3.0f);  glVertex2f(-8.0f, -2.5f);  glVertex2f(-8.0f, 1.0f);  glVertex2f(-6.0f, 1.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.350f, 0.35, 0.344f);  glVertex2f(-6.0f, 1.0f);  glVertex2f(-8.2f, 1.0f);  glVertex2f(-7.2f, 3.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.35f, 0.35f, 0.35f);  glVertex2f(-8.0f, 1.0f);  glVertex2f(-8.2f, 1.0f);  glVertex2f(-7.2f, 3.0f);  glVertex2f(-7.0f, 3.0f);  glEnd();  glBegin(GL\_POLYGON);  glColor3f(0.45f, 0.45f, 0.45f);  glVertex2f(-6.0f, 1.0f);  glVertex2f(-1.0f, 1.0f);  glVertex2f(-2.0f, 3.0f);  glVertex2f(-7.2f, 3.0f);  glEnd();  //  //Door1  //  glBegin(GL\_POLYGON);  glColor3f(0.9f, 0.6f, 0.0f);  glVertex2f(-3.0f, -3.0f);  glVertex2f(-4.0f, -3.0f);  glVertex2f(-4.0f, -1.0f);  glVertex2f(-3.0f, -1.0f);  glEnd();  //  //Win1  //  glBegin(GL\_POLYGON);  glColor3f(0.404f, 0.663f, 0.765f);  glVertex2f(-2.5f, -1.0f);  glVertex2f(-1.5f, -1.0f);  glVertex2f(-1.5f, 0.0f);  glVertex2f(-2.5f, 0.0f);  glEnd();  //  //Win2  //  glBegin(GL\_POLYGON);  glColor3f(0.404f, 0.663f, 0.765f);  glVertex2f(-5.5f, -1.0f);  glVertex2f(-5.5f, 0.0f);  glVertex2f(-4.5f, 0.0f);  glVertex2f(-4.5f, -1.0f);  glEnd();  //  //Door2  //  glBegin(GL\_POLYGON);  glColor3f(0.9f, 0.6f, 0.0f);  glVertex2f(-6.8f, -2.8f);  glVertex2f(-6.8f, -1.1f);  glVertex2f(-7.2f, -1.0f);  glVertex2f(-7.2f, -2.7f);  glEnd();  glFlush();  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInit(&argc, argv); // Initialize GLUT  glutCreateWindow("Vertex, Primitive & Color"); // Create window with the given title  glutInitWindowSize(720, 720); // Set the window's initial width & height  glutInitWindowPosition(20, 20); // Position the window's initial top-left corner  glutDisplayFunc(display); // Register callback handler for window re-paint event  gluOrtho2D(-30,30,-10,30);  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-**  **DAY** |

**Night**

