**Lab Taks-3**

Submission Guidelines-

* Rename the file with your serial number only
* Must submit within time that will be discussed in class VUES
* Must include resources for all the section in the table

|  |
| --- |
| **Question- 1**  Draw five storied building with windows and a front door |
| **Graph Plot (Picture)-**  **A grid with blue dots and letters  AI-generated content may be incorrect.** |
| **Code-**  #include <GL/glut.h>  // Display function  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  // Building structure  glColor3f(0.6, 0.6, 0.6); // Light gray  glBegin(GL\_POLYGON);  glVertex2f(200, 100);  glVertex2f(200, 400);  glVertex2f(400, 400);  glVertex2f(400, 100);  glEnd();  // Door  glColor3f(0.3, 0.2, 0.1); // Dark brown  glBegin(GL\_POLYGON);  glVertex2f(275, 100);  glVertex2f(275, 180);  glVertex2f(325, 180);  glVertex2f(325, 100);  glEnd();  // Windows - 3 rows, 2 columns  glColor3f(0.2, 0.5, 0.9); // Blue windows  int startX = 220, startY = 220;  for (int row = 0; row < 3; row++) {  for (int col = 0; col < 2; col++) {  int x = startX + col \* 80;  int y = startY + row \* 60;  glBegin(GL\_POLYGON);  glVertex2f(x, y);  glVertex2f(x, y + 40);  glVertex2f(x + 40, y + 40);  glVertex2f(x + 40, y);  glEnd();  }  }  glFlush();  }  // Initialization  void init() {  glClearColor(1.0, 1.0, 1.0, 1.0); // White background  gluOrtho2D(0, 600, 0, 500); // 2D orthographic view  }  // Main function  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);  glutInitWindowSize(600, 500);  glutInitWindowPosition(100, 100);  glutCreateWindow("Simple Building - GLUT");  init();  glutDisplayFunc(display);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-**  A computer screen shot of a computer screen  AI-generated content may be incorrect. |

|  |
| --- |
| **Question- 2**  Draw a tree |
| **Graph Plot (Picture)-** |
| **Code-**  #include <GL/glut.h>  #include <math.h>  void drawShiftedCircle(float cx, float cy, float r, float red, float green, float blue) {  float pi = 3.1416;  glBegin(GL\_POLYGON);  glColor3f(red, green, blue);  for (int i = 0; i < 200; i++) {  float A = (i \* 2 \* pi) / 200;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x + cx, y + cy);  }  glEnd();  }  void drawTree() {  drawShiftedCircle(0.0f, 0.47f, 0.3f, 0.0f, 0.6f, 0.0f);  drawShiftedCircle(-0.19f, 0.29f, 0.25f, 0.0f, 0.6f, 0.0f);  drawShiftedCircle(0.2f, 0.29f, 0.25f, 0.0f, 0.6f, 0.0f);  glBegin(GL\_QUADS);  glColor3f(0.4f, 0.2f, 0.0f);  glVertex2f(-0.08f, -0.48f);  glVertex2f(0.04f, -0.48f);  glVertex2f(0.04f, 0.15f);  glVertex2f(-0.08f, 0.15f);  glEnd();  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  drawTree();  glFlush();  }  void init() {  glClearColor(0.0f, 0.0f, 0.0f, 1.0f);  glMatrixMode(GL\_PROJECTION);  gluOrtho2D(-1, 1, -1, 1);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutCreateWindow("Tree");  glutInitWindowSize(500, 500);  glutInitWindowPosition(100, 100);  init();  glutDisplayFunc(display);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-**  **A computer screen with a black screen  AI-generated content may be incorrect.** |

|  |
| --- |
| **Question- 3**  Draw a lamppost with black background |
| **Graph Plot (Picture)-** |
| **Code-**  #include <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  #include <math.h>  void stand1(){  glColor3ub(0,0,0);  glBegin(GL\_POLYGON);  glVertex2f(-0.1,1.5);  glVertex2f(0.2,1.5);  glVertex2f(0.2,-0.4);  glVertex2f(-0.1,-0.4);  glEnd();  glColor3ub(0, 0, 0);  glLineWidth(1);  glBegin(GL\_LINE\_LOOP);  glVertex2f(-0.1,1.5);  glVertex2f(0.2,1.5);  glVertex2f(0.2,-0.4);  glVertex2f(-0.1,-0.4);  glEnd();  }  void stand2(){  glColor3ub(0,0,0);  glBegin(GL\_POLYGON);  glVertex2f(-0.1,1.5);  glVertex2f(-0.1,1.3);  glVertex2f(-1.5,1.3);  glVertex2f(-1.5,1.5);  glEnd();  glColor3ub(0, 0, 0);  glLineWidth(1);  glBegin(GL\_LINE\_LOOP);  glVertex2f(-0.1,1.5);  glVertex2f(-0.1,1.3);  glVertex2f(-1.5,1.3);  glVertex2f(-1.5,1.5);  glEnd();  }  void triangle(){  glColor3ub(0,0,0);  glBegin(GL\_POLYGON);  glVertex2f(-1.5,1.3);  glVertex2f(-2,1);  glVertex2f(-1,1);  glEnd();  glColor3ub(0, 0, 0);  glLineWidth(1);  glBegin(GL\_LINE\_LOOP);  glVertex2f(-1.5,1.3);  glVertex2f(-2,1);  glVertex2f(-1,1);  glEnd();  }  void circle(float cx,float cy)  {glBegin(GL\_POLYGON);  for(int i=0;i<200;i++)  {  glColor3ub(243,156,18);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=0.2;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+cx,y+cy );  }  glEnd();  }  void lamppost(float r, float g, float b)  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  //glScalef(0.3,0.3,0);  glTranslatef(0,0,0);  stand1();  stand2();  triangle();  circle(-1.49,0.83);  glPopMatrix();  }  void display() {  glClearColor(0.69f, 0.88f, 0.90f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT);  lamppost(237,187,153);  glFlush();  // Render now  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitWindowSize(300, 300); // Initialize GLUT  glutCreateWindow("Lamp-post"); // Create a window with the given title  // Set the window's initial width & height  glutDisplayFunc(display);  gluOrtho2D(-2,2,-2,2);// Register display callback handler for window re-paint  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-**  **A computer screen with a black screen  AI-generated content may be incorrect.** |

|  |
| --- |
| **Question- 4**  Draw a car |
| **Graph Plot (Picture)-**  **A drawing of a car on a graph paper  AI-generated content may be incorrect.** |
| **Code-**  #include <GL/glut.h>  #include <math.h>  void drawCircle(float cx, float cy, float r) {  int num\_segments = 100;  glBegin(GL\_POLYGON);  for (int i = 0; i < num\_segments; i++) {  float theta = 2.0f \* 3.1416f \* i / num\_segments;  float x = r \* cosf(theta);  float y = r \* sinf(theta);  glVertex2f(x + cx, y + cy);  }  glEnd();  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  // Car Body  glColor3f(1.0, 0.0, 0.0); // Red  glBegin(GL\_POLYGON);  glVertex2f(100, 100);  glVertex2f(100, 150);  glVertex2f(200, 150);  glVertex2f(200, 100);  glEnd();  // Car Top  glBegin(GL\_POLYGON);  glVertex2f(120, 150);  glVertex2f(140, 180);  glVertex2f(160, 180);  glVertex2f(180, 150);  glEnd();  // Wheels  glColor3f(0.0, 0.0, 0.0); // Black  drawCircle(120, 90, 10); // Left wheel  drawCircle(180, 90, 10); // Right wheel  glFlush();  }  void init() {  glClearColor(1.0, 1.0, 1.0, 1.0); // White background  gluOrtho2D(0, 300, 0, 250); // 2D projection area  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);  glutInitWindowSize(600, 400);  glutInitWindowPosition(100, 100);  glutCreateWindow("Car");  init();  glutDisplayFunc(display);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-**  **A computer screen shot of a red truck  AI-generated content may be incorrect.** |

|  |
| --- |
| **Question- 5**  Use the building, tree, lamppost and car to create a scenario |
|  |
| **Code-**  #include <GL/glut.h>  #include <math.h>  // ---------- Utility ----------  void drawCircle(float cx, float cy, float r) {  int segments = 100;  glBegin(GL\_POLYGON);  for (int i = 0; i < segments; i++) {  float theta = 2.0f \* 3.1416f \* i / segments;  float x = r \* cosf(theta);  float y = r \* sinf(theta);  glVertex2f(cx + x, cy + y);  }  glEnd();  }  // ---------- Building ----------  void drawBuilding() {  // Main structure  glColor3f(0.6, 0.6, 0.6);  glBegin(GL\_POLYGON);  glVertex2f(0, 0);  glVertex2f(0, 300);  glVertex2f(150, 300);  glVertex2f(150, 0);  glEnd();  // Door  glColor3f(0.3, 0.2, 0.1);  glBegin(GL\_POLYGON);  glVertex2f(60, 0);  glVertex2f(60, 60);  glVertex2f(90, 60);  glVertex2f(90, 0);  glEnd();  // Windows  glColor3f(0.2, 0.5, 0.9);  for (int i = 0; i < 3; i++) {  for (int j = 0; j < 2; j++) {  float x = 20 + j \* 70;  float y = 180 + i \* 40;  glBegin(GL\_POLYGON);  glVertex2f(x, y);  glVertex2f(x + 30, y);  glVertex2f(x + 30, y + 30);  glVertex2f(x, y + 30);  glEnd();  }  }  }  // ---------- Tree ----------  void drawTree() {  // Leaves  glColor3f(0.0, 0.6, 0.0);  drawCircle(50, 120, 30);  drawCircle(30, 90, 25);  drawCircle(70, 90, 25);  // Trunk  glColor3f(0.4, 0.2, 0.0);  glBegin(GL\_QUADS);  glVertex2f(40, 40);  glVertex2f(55, 40);  glVertex2f(55, 90);  glVertex2f(40, 90);  glEnd();  }  // ---------- Lamp-post ----------  void drawLamp() {  // Pole  glColor3f(0.0, 0.0, 0.0);  glBegin(GL\_POLYGON);  glVertex2f(10, 0);  glVertex2f(20, 0);  glVertex2f(20, 120);  glVertex2f(10, 120);  glEnd();  // Top bar  glBegin(GL\_POLYGON);  glVertex2f(20, 120);  glVertex2f(20, 130);  glVertex2f(80, 130);  glVertex2f(80, 120);  glEnd();  // Triangle arm  glBegin(GL\_POLYGON);  glVertex2f(80, 120);  glVertex2f(100, 100);  glVertex2f(60, 100);  glEnd();  // Light circle  glColor3ub(243, 156, 18);  drawCircle(78, 100, 10);  }  // ---------- Car ----------  void drawCar() {  // Body  glColor3f(1.0, 0.0, 0.0);  glBegin(GL\_POLYGON);  glVertex2f(0, 0);  glVertex2f(0, 40);  glVertex2f(100, 40);  glVertex2f(100, 0);  glEnd();  // Top  glBegin(GL\_POLYGON);  glVertex2f(20, 40);  glVertex2f(35, 60);  glVertex2f(65, 60);  glVertex2f(80, 40);  glEnd();  // Wheels  glColor3f(0.0, 0.0, 0.0);  drawCircle(25, -10, 10);  drawCircle(75, -10, 10);  }  // ---------- Display ----------  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  // Draw building  glPushMatrix();  glTranslatef(50, 100, 0);  drawBuilding();  glPopMatrix();  // Draw tree  glPushMatrix();  glTranslatef(250, 100, 0);  drawTree();  glPopMatrix();  // Draw lamp  glPushMatrix();  glTranslatef(400, 100, 0);  drawLamp();  glPopMatrix();  // Draw car  glPushMatrix();  glTranslatef(200, 30, 0);  drawCar();  glPopMatrix();  glFlush();  }  // ---------- Init ----------  void init() {  glClearColor(1.0, 1.0, 1.0, 1.0);  gluOrtho2D(0, 600, 0, 400);  }  // ---------- Main ----------  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitWindowSize(600, 400);  glutInitWindowPosition(100, 100);  glutCreateWindow("Scenario");  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);  init();  glutDisplayFunc(display);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-**  A computer screen shot of a computer screen  AI-generated content may be incorrect. |