**Detailed Explanation of the Code:**

This code is a simple OpenGL program that draws a **quadratic Bézier curve** based on 3 control points. Let's break down each section of the code in detail.

**1. Including Header Files:**

#include <GL/glut.h>

#include <math.h>

* **#include <GL/glut.h>**: This header file includes the OpenGL Utility Toolkit (GLUT), which provides functions to create windows, handle user inputs, and manage rendering in OpenGL applications. GLUT is widely used for 2D and 3D graphics programs.
* **#include <math.h>**: This includes the math library in C, which provides various mathematical functions. It can be used for trigonometric, exponential, logarithmic, and other operations. In this code, we aren't using many functions from the math library, but it might be useful for future expansions, especially when dealing with calculations in 3D graphics or more complex math.

**2. Control Points Definition:**

float ctrlPoints[3][2] = {

{-0.9, -0.9},

{ 0.0, 0.9},

{ 0.9, -0.9}

};

* **Explanation**:
  + This is a 2D array where each pair of values represents the (x, y) coordinates of a control point in the Cartesian plane. These 3 points will define the shape of the quadratic Bézier curve.
  + The first control point is at (-0.9, -0.9), the second control point is at (0.0, 0.9), and the third one is at (0.9, -0.9).
  + The points are hard-coded, but in an interactive application, they could be modified by the user.

**3. Drawing the Bézier Curve:**

void drawBezierCurve() {

glBegin(GL\_LINE\_STRIP);

for (float t = 0.0; t <= 1.0; t += 0.001) {

float u = 1.0f - t;

float x = u \* u \* ctrlPoints[0][0] +

2 \* u \* t \* ctrlPoints[1][0] +

t \* t \* ctrlPoints[2][0];

float y = u \* u \* ctrlPoints[0][1] +

2 \* u \* t \* ctrlPoints[1][1] +

t \* t \* ctrlPoints[2][1];

glVertex2f(x, y);

}

glEnd();

}

* **Explanation**:
  + The function drawBezierCurve is responsible for calculating and drawing the quadratic Bézier curve using the control points.
  + **Bézier Curve Formula**:  
    A quadratic Bézier curve is calculated by the formula:

B(t)=(1−t)2P0+2(1−t)tP1+t2P2B(t) = (1-t)^2 P\_0 + 2(1-t)t P\_1 + t^2 P\_2

Where:

* + - P0,P1,P2P\_0, P\_1, P\_2 are the control points.
    - tt is the parameter that varies from 0 to 1.
    - The curve is generated by computing the (x, y) coordinates at each value of tt, from 0 to 1.
  + **glBegin(GL\_LINE\_STRIP)**: This OpenGL function begins the drawing of a connected series of line segments.
  + **glVertex2f(x, y)**: For each value of tt, the calculated coordinates (x, y) are passed to glVertex2f, which tells OpenGL to plot that point on the screen.
  + **Loop**: The loop runs from t=0t = 0 to t=1t = 1 with a step size of 0.001, which ensures a smooth curve by calculating many intermediate points.

**4. Display Function:**

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0f, 1.0f, 0.0f); // Yellow curve

drawBezierCurve();

glColor3f(1.0f, 0.0f, 0.0f); // Red control points

glPointSize(5.0f);

glBegin(GL\_POINTS);

for (int i = 0; i < 3; i++)

glVertex2f(ctrlPoints[i][0], ctrlPoints[i][1]);

glEnd();

glFlush();

}

* **Explanation**:
  + glClear(GL\_COLOR\_BUFFER\_BIT);: Clears the screen to prepare for a new frame.
  + **Drawing the Bézier Curve**:
    - glColor3f(1.0f, 1.0f, 0.0f);: Sets the color of the Bézier curve to yellow.
    - drawBezierCurve();: Calls the function to draw the curve based on the control points.
  + **Drawing the Control Points**:
    - glColor3f(1.0f, 0.0f, 0.0f);: Sets the color of the control points to red.
    - glPointSize(5.0f);: Sets the size of the control points so they are visible.
    - The glBegin(GL\_POINTS) and glEnd() functions are used to draw the points at the positions of the control points.
  + glFlush();: Forces OpenGL to execute the rendering commands immediately.

**5. Initialization Function:**

void init() {

glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Background black

glMatrixMode(GL\_PROJECTION); // Set matrix mode

glLoadIdentity(); // Load identity matrix

gluOrtho2D(-1.0, 1.0, -1.0, 1.0); // Set 2D orthographic projection

}

* **Explanation**:
  + glClearColor(0.0f, 0.0f, 0.0f, 1.0f);: Sets the background color to black.
  + **Setting Up the Projection**:
    - glMatrixMode(GL\_PROJECTION);: Switches to the projection matrix mode to control the viewing area.
    - glLoadIdentity();: Resets the current matrix to the identity matrix.
    - gluOrtho2D(-1.0, 1.0, -1.0, 1.0);: Sets up a 2D orthographic projection with the visible area ranging from -1 to 1 in both x and y directions.

**6. Main Function:**

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(800, 600);

glutCreateWindow("Simple Bézier Curve");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

* **Explanation**:
  + glutInit(&argc, argv);: Initializes GLUT and processes any command-line arguments.
  + glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);: Specifies that the program will use single buffering and RGB color mode.
  + glutInitWindowSize(800, 600);: Sets the window size to 800x600 pixels.
  + glutCreateWindow("Simple Bézier Curve");: Creates a window with the title "Simple Bézier Curve".
  + init();: Calls the initialization function.
  + glutDisplayFunc(display);: Registers the display function to render the scene.
  + glutMainLoop();: Starts the GLUT main event loop to handle rendering and events.

**Potential Examiner Questions and Detailed Answers:**

**1. Question**: *What is the purpose of the ctrlPoints array, and how does it affect the Bézier curve?*

* **Answer**: The ctrlPoints array holds the 3 control points that define the shape of the quadratic Bézier curve. The coordinates of each control point are used in the Bézier formula to calculate the curve's path. Changing these points will change the direction and shape of the curve. For example, moving the second control point will pull the curve toward or away from that point, altering the curve’s curvature.

**2. Question**: *Explain the Bézier curve formula used in the code. What happens if you change the value of t?*

* **Answer**: The Bézier curve formula used in the code is:

B(t)=(1−t)2P0+2(1−t)tP1+t2P2B(t) = (1-t)^2 P\_0 + 2(1-t)t P\_1 + t^2 P\_2

* + P0,P1,P2P\_0, P\_1, P\_2 are the control points.
  + tt is the parameter that ranges from 0 to 1.

The value of t determines the position along the curve. As t increases from 0 to 1, the curve moves from the first control point (P0P\_0) to the second (P1P\_1) and finally to the third (P2P\_2). The function calculates intermediate points on the curve by varying t.

**3. Question**: *Why is the glBegin(GL\_LINE\_STRIP) used in the drawBezierCurve function?*

* **Answer**: glBegin(GL\_LINE\_STRIP) tells OpenGL to draw a continuous series of line segments between the points specified by glVertex2f. This is used to create the smooth Bézier curve by connecting the intermediate points calculated for each value of t.

**4. Question**: *What are the possible ways to interactively change the control points and modify the curve?*

* **Answer**:
  + One way to interactively change the control points is by allowing the user to click and drag the control points using the mouse.
  + This can be achieved using GLUT's event handling functions like glutMouseFunc() to track mouse clicks and glutMotionFunc() to track mouse movements.
  + After modifying the control points, the display function should be called to re-render the curve with the new control points.

**5. Question**: *What would happen if you changed the color of the curve?*

* **Answer**: Changing the color of the curve can be done using glColor3f(). The current code sets the curve color to yellow (glColor3f(1.0f, 1.0f, 0.0f)). You can modify the RGB values in glColor3f() to change the color. For example, setting it to red would be glColor3f(1.0f, 0.0f, 0.0f).

**6. Question**: *Can the code handle 3D Bézier curves? If not, how would you modify it?*

* **Answer**: The code currently handles only 2D Bézier curves. To handle 3D Bézier curves, you would need to modify the control points to include 3 coordinates (x, y, z) and use the 3D Bézier formula. Additionally, you would need to use glVertex3f(x, y, z) instead of glVertex2f(x, y) to plot the points in 3D space.

These are detailed explanations of the code, along with possible questions the examiner may ask and their respective answers.