Here’s a **detailed explanation** of your Koch Curve OpenGL program in simple terms, ideal for viva or understanding how it works step-by-step.

**🔍 Program Overview:**

This C++ program uses **OpenGL** and **recursive logic** to draw a **Koch Curve**, a famous fractal (a pattern that repeats itself). The curve is drawn between two points, and at each recursive level, the line segment is split and altered to create the "snowflake" shape.

**🧱 Code Breakdown**

**1. Header Files**

#include <GL/glut.h>

#include <math.h>

* GL/glut.h: Required to use OpenGL functions (via GLUT).
* math.h: Used to calculate cosine and sine values for angles (needed for triangle peak calculation).

**2. drawKoch Function**

void drawKoch(float x1, float y1, float x2, float y2, int depth)

* **Purpose**: Draws the Koch Curve recursively between (x1, y1) and (x2, y2).

**🔁 If depth == 0:**

if (depth == 0) {

glVertex2f(x1, y1);

glVertex2f(x2, y2);

return;

}

* If recursion is complete, draw a straight line between the two points.

**🔄 Else:**

float dx = (x2 - x1) / 3.0f;

float dy = (y2 - y1) / 3.0f;

* Divide the segment into 3 parts.

float xA = x1 + dx;

float yA = y1 + dy;

float xB = x1 + 2 \* dx;

float yB = y1 + 2 \* dy;

* Points at 1/3 and 2/3 positions.

**🔺 Calculate the peak of equilateral triangle:**

float px = xA + (dx \* cos(M\_PI / 3.0f)) - (dy \* sin(M\_PI / 3.0f));

float py = yA + (dx \* sin(M\_PI / 3.0f)) + (dy \* cos(M\_PI / 3.0f));

* Uses 60° angle (π/3) to find the triangle's peak.

**📐 Recursive Calls:**

drawKoch(x1, y1, xA, yA, depth - 1);

drawKoch(xA, yA, px, py, depth - 1);

drawKoch(px, py, xB, yB, depth - 1);

drawKoch(xB, yB, x2, y2, depth - 1);

* Repeats for 4 smaller segments.

**3. display Function**

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.2f, 0.7f, 1.0f); // Light blue

glBegin(GL\_LINES);

drawKoch(-0.8f, 0.0f, 0.8f, 0.0f, 4); // Start and end point, with depth 4

glEnd();

glFlush();

}

* Clears screen, sets line color, starts line drawing mode.
* Calls drawKoch() from left to right of the screen with a recursion **depth of 4**.
* Ends drawing and flushes to screen.

**4. init Function**

void init() {

glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black background

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1, 1, -1, 1); // 2D projection from -1 to 1 in both axes

}

* Sets black background.
* Sets 2D coordinate system ranging from -1 to 1 (OpenGL's normalized space).

**5. main Function**

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(800, 600);

glutCreateWindow("Koch Curve");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

* Initializes GLUT.
* Sets window size and mode.
* Creates a window titled "Koch Curve".
* Registers display() as the function to render graphics.
* Enters the GLUT event loop.

**🎓 Viva Questions and Answers:**

**Q1. What is a Koch Curve?**  
**A:** It’s a fractal curve built by recursively dividing a line into 4 parts and forming a triangle on the middle third. It’s an example of a geometric fractal.

**Q2. Why do we use recursion in the Koch curve?**  
**A:** Because the Koch curve is built by repeating the same pattern at smaller scales. Recursion simplifies this repetition.

**Q3. What does the depth parameter represent?**  
**A:** It controls how many times the line is divided. Higher depth = more detailed curve.

**Q4. Why is cos(π/3) and sin(π/3) used?**  
**A:** To find the coordinates of the triangle's peak point since we are forming a 60° (π/3 rad) equilateral triangle.

**Q5. What happens if depth is increased?**  
**A:** The curve becomes more complex with more divisions. Execution time and line segments increase exponentially.

**Q6. Why is glBegin(GL\_LINES) used?**  
**A:** Because each segment of the Koch curve is a line. GL\_LINES tells OpenGL to draw using line segments.

Would you like a **diagram** or **animated visualization** to better understand how the recursion builds the Koch Curve?