```python
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math

# Number of sides for the polygon (e.g., 8 for an octagon)
n = 9  # Change this to any desired number of sides

# Display callback function
def display():
    # Clear the screen
    glClear(GL_COLOR_BUFFER_BIT)

    # Draw the polygon
    draw_polygon()

    # Swap buffers
    glutSwapBuffers()

# Function to draw the polygon using GL_TRIANGLE_FAN
def draw_polygon():
    radius = 0.8  # Radius of the polygon
    angle_step = 2 * math.pi / n  # Angle between vertices

    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)  # Wireframe mode
    glColor3f(1.0, 0, 0)  # Set polygon color (red)

    glBegin(GL_TRIANGLE_FAN)

    # Center vertex of the polygon
    glVertex2f(0.0, 0.0)

    # Vertices of the polygon
    for i in range(n + 1):  # Loop to include the first and last vertex
        angle = i * angle_step
        x = radius * math.cos(angle)
        y = radius * math.sin(angle)
        glVertex2f(x, y)

    glEnd()

# Initialize GLUT
glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)
glutInitWindowSize(500, 500)
glutInitWindowPosition(500,250)
glutCreateWindow("Regular Polygon with GL_TRIANGLE_FAN")
glutDisplayFunc(display)
glutMainLoop()
```

Drawing Regular Polygons with Triangle Fan in Python and OpenGL

This code utilizes OpenGL to render a wireframe regular polygon using the `GL_TRIANGLE_FAN` primitive. You can modify the code to create polygons with any number of sides.

Number of Sides:

- `n = 9` (changeable): This variable defines the number of sides in the polygon. Change it to any desired value (e.g., 3 for a triangle, 4 for a square, etc.)

Display Callback Function (display())

- `glClear(GL_COLOR_BUFFER_BIT)`: Clears the color buffer, erasing the previous frame.
- `draw_polygon()`: Calls the function responsible for drawing the polygon.
- `glutSwapBuffers()`: Swaps the front and back buffers for smooth animation without flickering.

Polygon Drawing Function (draw_polygon())

- `radius = 0.8`: Defines the radius of the polygon, controlling its size.
- `angle_step = 2 * math.pi / n`: Calculates the angle increment between each vertex, ensuring a regular polygon shape.
- **Wireframe Mode and Color:**
  - `glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)`: Sets the polygon rendering mode to wireframe, displaying only the edges.
  - `glColor3f(1.0, 0, 0)`: Sets the drawing color to red for the polygon.
- **Drawing with Triangle Fan:**
  - `glBegin(GL_TRIANGLE_FAN)`: Starts drawing a polygon using the Triangle Fan primitive.
    - This primitive efficiently creates a filled polygon by connecting all vertices to a central point.
  - `glVertex2f(0.0, 0.0)`: Specifies the first vertex at the center of the polygon (0, 0).
- **Looping for Vertices:**
  - `for i in range(n + 1)`: Iterates through `n + 1` vertices to ensure the first and last points are connected, closing the polygon shape.
  - Inside the loop:
    - `angle = i * angle_step`: Calculates the angle for the current vertex based on the loop iteration and the calculated angle increment.
    - `x = radius * math.cos(angle)` and `y = radius * math.sin(angle)`: Calculates the x and y coordinates of the current vertex using the radius and the calculated angle.
    - `glVertex2f(x, y)`: Specifies the vertex position for the current point on the polygon's circumference.
  - `glEnd()`: Ends drawing the polygon.

GLUT Initialization and Configuration

- `glutInit()`: Initializes the GLUT library.
- `glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)`: Sets the display mode for double buffering and RGB colors.
- `glutInitWindowSize(500, 500)`: Defines the window size to be 500x500 pixels.
- `glutInitWindowPosition(500,250)`: Sets the initial window position to (500, 250) on the screen.
- `glutCreateWindow("Regular Polygon with GL_TRIANGLE_FAN")`: Creates the window with the specified title.
- `glutDisplayFunc(display)`: Assigns the `display()` function to be called for rendering the scene.
- `glutMainLoop()`: Enters the main event loop, continuously listening for events and calling the registered callback function (`display()`) to update the window contents.

Summary

This code demonstrates how to leverage OpenGL with Python to draw regular polygons using the `GL_TRIANGLE_FAN` primitive. It showcases efficient rendering and control over the number of sides and polygon size. Remember to adjust the `n` variable to create different regular polygons.