Cozy Circles In Regular Polygons

Vamshi Jandhyala

4/21/2025

A nice puzzle by Xavier Durawa for the week of 4/20.

1 Puzzle

You start with a regular triangle. At the midpoint of each side you draw a circle on the inside of the triangle where the circle is tangent to the side. Each circle is the same size and its radius is maximized such that the interior circles touch each other. Below is a diagram to illustrate.

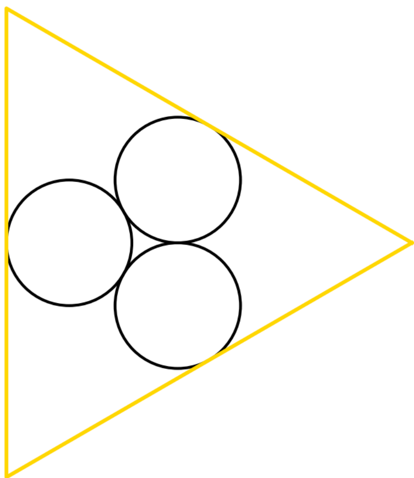

Figure 1: Circles in a triangle

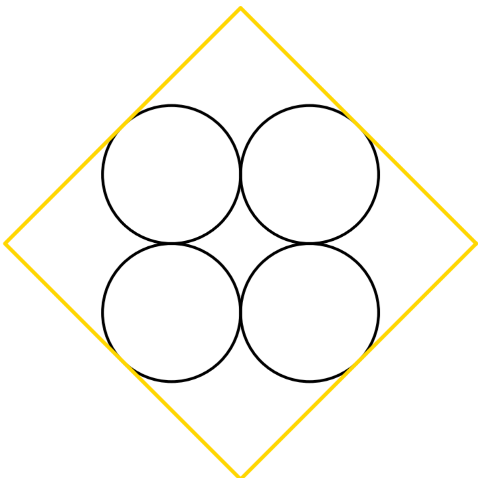

Figure 2: Circles in a square

If we consider a regular triangle, what is the ratio of the area of the 3 interior circles to the area of the triangle? How about a square? How about a regular n-gon?

2 Solution

Let the regular polygon have n sides and its side be s.Let the radius of each circle be r. Let us consider two adjacent sides of a regular polygon. Let C be the common vertex, points F and D be the points of tangency of the circles and points G and H be the centers of the two circles. The figure below illustrates these points when the regular polygon is an equilateral triangle.

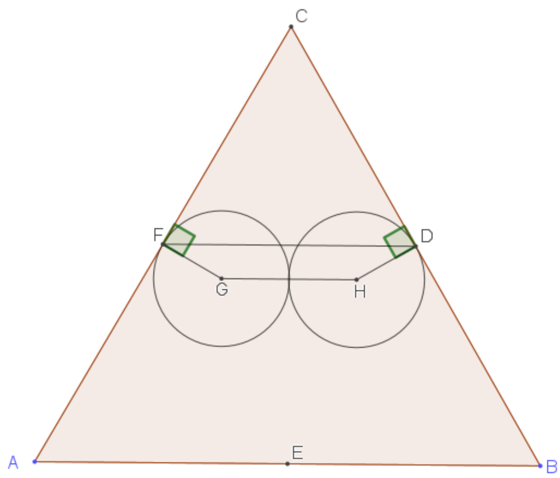

Figure 3: Circles in a triangle

The internal angle at C is

$$\angle C = \frac{(n-2)\pi}{n}. \tag{2.1}$$

It is easy to see that length of FD is

$$|FD| = 2\left(\frac{s}{2}\right)\cos\left(\frac{\pi - \frac{(n-2)\pi}{n}}{2}\right) = s\cos\left(\frac{\pi}{n}\right). \tag{2.2}$$

We have,

$$\angle GFD = \angle HDF = \frac{\pi}{2} - \frac{\pi}{n} = \frac{\pi(n-2)}{2n}. \tag{2.3}$$

As $|GF| = |DH| = r$ and $|GH| = 2r$, we also have,

$$\begin{aligned} |FD| &= 2r\cos(\angle GFD) + |GH| \\ \Rightarrow s\cos\left(\frac{\pi}{n}\right) &= 2r\left(\cos\left(\frac{\pi(n-2)}{2n}\right) + 1\right) \\ \Rightarrow r &= \frac{s\cos\left(\frac{\pi}{n}\right)}{2\left(\sin\left(\frac{\pi}{n}\right)+1\right)}. \end{aligned} \tag{2.4}$$

The area of a n-sided regular polygon of side s is given by,

$$\frac{ns^2}{4}\cot\left(\frac{\pi}{n}\right). \tag{2.5}$$

The area of all the circles is given by,

$$n\pi\left(\frac{s\cos\left(\frac{\pi}{n}\right)}{2\left(\sin\left(\frac{\pi}{n}\right)+1\right)}\right)^2. \tag{2.6}$$

The required ratio is therefore,

$$\boldsymbol{\frac{\pi}{2}\frac{\sin\left(\frac{2\pi}{n}\right)}{\left(\sin\left(\frac{\pi}{n}\right)+1\right)^2}}. \tag{2.7}$$

For a triangle, $n = 3$ so the ratio is,

$$\frac{\pi}{2}\frac{\frac{\sqrt{3}}{2}}{\left(1+\frac{\sqrt{3}}{2}\right)^2} = \boldsymbol{\frac{\pi\sqrt{3}}{7+4\sqrt{3}}}. \tag{2.8}$$

For a square, $n = 4$ so the ratio is,

$$\frac{\pi}{2}\frac{1}{\left(1+\frac{1}{\sqrt{2}}\right)^2} = \boldsymbol{\frac{\pi}{3+2\sqrt{2}}}. \tag{2.9}$$

3 Python code

The Python code to draw the circles for a given regular polygon is given below. Here is some sample output:

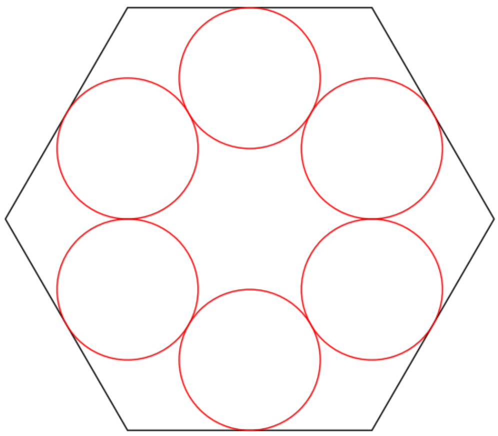

Figure 4: Circles in a hexagon

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon, Circle
import math

def draw_regular_polygon_with_circles(n,
                                     polygon_edge_color='black',
                                     polygon_fill_color='none',
                                     circle_color='blue',
                                     show_figure=True):
    """
    Draw a regular polygon with n sides and identical circles that touch each side at
its midpoint.

    Parameters:
    -----------
    n : int
        Number of sides in the polygon
    side_length : float
        Length of each side of the polygon
    circle_radius : float
        Radius of each circle
    polygon_edge_color : str
        Color of the polygon edge
    polygon_fill_color : str
        Color of the polygon fill (use 'none' for transparent)
    circle_color : str
        Color of the circles
```

```
    show_figure : bool
        Whether to show the plot immediately

    Returns:
    --------
    matplotlib.figure.Figure
        The figure containing the drawing
    """
    # Create figure
    fig = plt.figure(figsize=(8, 8))
    ax = fig.add_subplot(111)

    # Calculate the radius of the circumscribed circle
    R = 1 / (2 * math.sin(math.pi / n))

    # Calculate coordinates for the polygon vertices
    vertices = []
    for i in range(n):
        angle = 2 * math.pi * i / n
        vertices.append((R * math.cos(angle), R * math.sin(angle)))

    # Draw the polygon
    polygon = Polygon(vertices, closed=True, edgecolor=polygon_edge_color,
                      facecolor=polygon_fill_color if polygon_fill_color != 'none'
else 'none',
                      alpha=0.3 if polygon_fill_color != 'none' else 1)
    ax.add_patch(polygon)


    circle_radius = math.cos(math.pi/n)/(2*(1+math.sin(math.pi/n)))

    # For each side, calculate circle center position
    for i in range(n):
        j = (i + 1) % n

        # Calculate midpoint of the current side
        midpoint_x = (vertices[i][0] + vertices[j][0]) / 2
        midpoint_y = (vertices[i][1] + vertices[j][1]) / 2

        # Calculate the normal vector to the side
        dx = vertices[j][0] - vertices[i][0]
        dy = vertices[j][1] - vertices[i][1]

        # Rotate 90 degrees to get the normal vector (perpendicular to side)
        normal_x = -dy
        normal_y = dx

        # Normalize the normal vector
        normal_length = math.sqrt(normal_x**2 + normal_y**2)
        normal_x = normal_x / normal_length
        normal_y = normal_y / normal_length

        # Check if normal points inward (toward center of polygon)
```

```
        center_to_mid_x = midpoint_x
        center_to_mid_y = midpoint_y
        dot_product = normal_x * center_to_mid_x + normal_y * center_to_mid_y

        # If dot product is positive, normal points outward, so invert it
        if dot_product > 0:
            normal_x = -normal_x
            normal_y = -normal_y

        # Calculate the center of the circle
        # The circle center is at distance circle_radius from the midpoint along the
normal
        circle_center_x = midpoint_x + normal_x * circle_radius
        circle_center_y = midpoint_y + normal_y * circle_radius

        # Draw the circle
        circle = Circle((circle_center_x, circle_center_y), circle_radius,
                        fill=False, edgecolor=circle_color)
        ax.add_patch(circle)

    # Set equal aspect ratio and limits
    ax.set_aspect('equal')
    margin = R * 1.2
    ax.set_xlim(-margin, margin)
    ax.set_ylim(-margin, margin)

    # Remove axis, grid, and frame
    ax.set_axis_off()

    if show_figure:
        plt.tight_layout()
        plt.show()

    return fig


# Draw a hexagon with circles
fig1 = draw_regular_polygon_with_circles(
    n=6,                     # Hexagon
    polygon_edge_color='black',
    polygon_fill_color='none',
    circle_color='red',
    show_figure=False
)

# Draw a triangle with circles
fig2 = draw_regular_polygon_with_circles(
    n=3,                     # Triangle
    polygon_edge_color='black',
    polygon_fill_color='none',
    circle_color='red',
    show_figure=False
)
```

```python
# Draw a triangle with circles
fig3 = draw_regular_polygon_with_circles(
    n=4,                      # Triangle
    polygon_edge_color='black',
    polygon_fill_color='none',
    circle_color='red',
    show_figure=False
)


# Display the figures
plt.figure(fig1.number)
plt.savefig('hexagon_with_circles.png', bbox_inches='tight', dpi=100)
plt.figure(fig2.number)
plt.savefig('triangle_with_circles.png', bbox_inches='tight', dpi=100)
plt.figure(fig3.number)
plt.savefig('square_with_circles.png', bbox_inches='tight', dpi=100)
plt.show()
```