SPRAWOZDANIE

Zajęcia: Grafika komputerowa
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium II
Data
Temat: Zadanie PYGAME

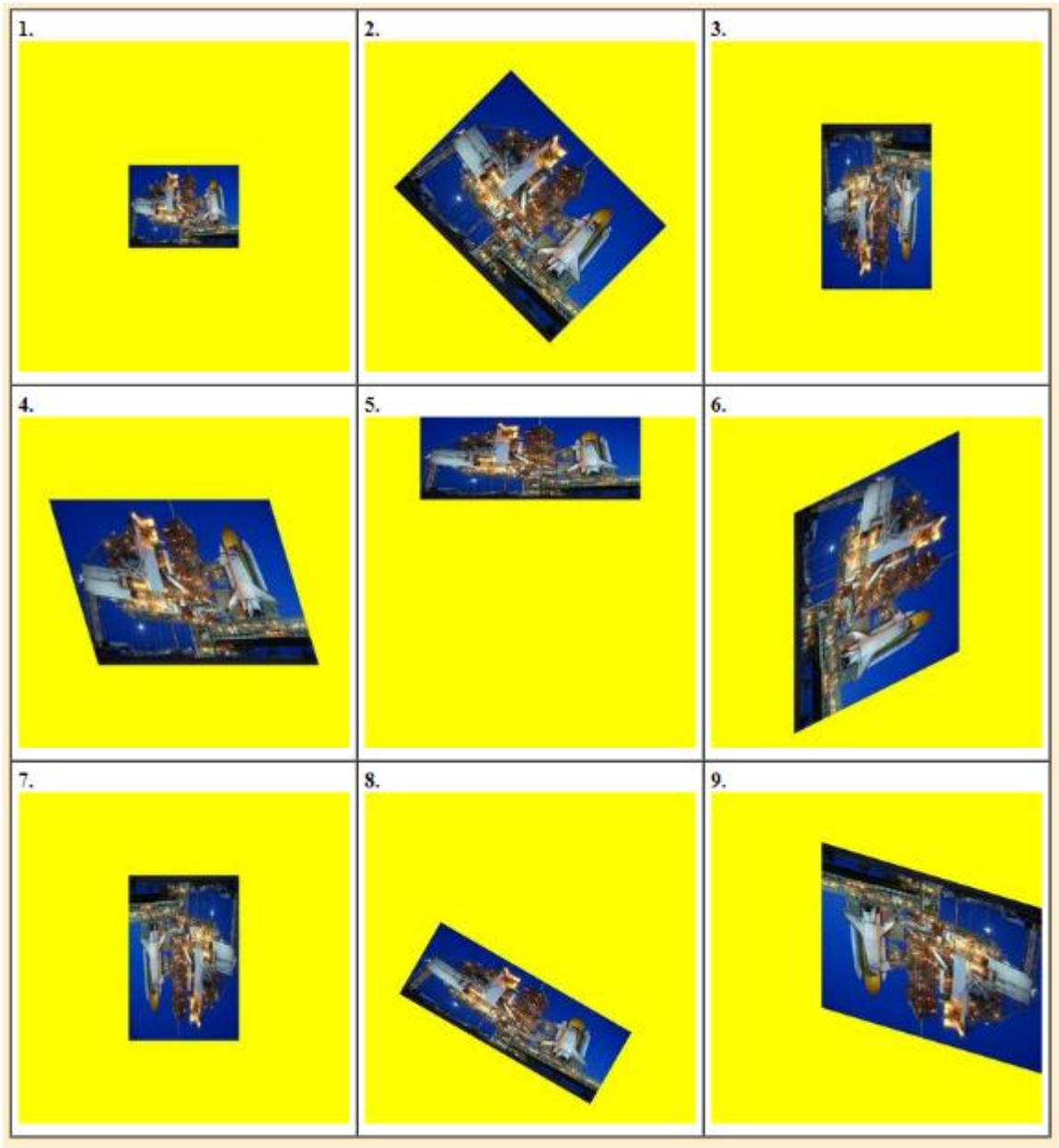
Wariant 8 + 4

Jakub Bąk
Informatyka I stopień,
stacjonarne,
4 semestr,
Gr.3b

1. Polecenie

Na podstawie wylosowanego numeru (8), należy narysować wielokąt o ilości kątów równej numer plus 4, oraz wykonanie dziewięciu operacji na tym wielekącię używając biblioteki pygame.

2. Wprowadzane dane:



Na podstawie podanego obrazka, należało odwzorować funkcje w programie do przekształcenia podanej figury.

3. Wykorzystane komendy:

Link do github: https://github.com/Szeladin/grafika.git

```python
import pygame
import math

pygame.init()
screen = pygame.display.set_mode((600, 600))
screen.fill((255, 255, 255))

# Dodecagon points
def calculate_dodecagon_points(center, radius):
    points = []
    for i in range(12):
        angle = math.radians(30 + i * 30)
        x = center[0] + radius * math.cos(angle)
        y = center[1] + radius * math.sin(angle)
        points.append((x, y))
    return points

center = (300, 300)
radius = 150
dodecagon_points = calculate_dodecagon_points(center, radius)

# Function to draw the dodecagon with  colors
def draw_dodecagon(surface, points):
    surface.fill((255, 255, 255, 0))
    colors = [(255, 0, 0), (255, 255, 0), (0, 0, 255)]  # Red, Yellow, Blue
    for i in range(len(points)):
        start_point = points[i]
        end_point = points[(i + 1) % len(points)]
        color = colors[i // 4]
        pygame.draw.line(surface, color, start_point, end_point, 2)
    screen.fill((255, 255, 255))
    screen.blit(surface, (0, 0))
    pygame.display.flip()

# Surface for the dodecagon
dodecagon_surface = pygame.Surface((600, 600), pygame.SRCALPHA)
draw_dodecagon(dodecagon_surface, dodecagon_points)

# Function to draw the transformed dodecagon centered on the screen
def draw_transformed_dodecagon(transformed_surface):
    screen.fill((255, 255, 255))
    rect = transformed_surface.get_rect(center=(300, 300))
    screen.blit(transformed_surface, rect.topleft)
    pygame.display.flip()

run = True
while run:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit(0)

    keys = pygame.key.get_pressed()

    if keys[pygame.K_1]:  # Option 1: Make it smaller
        scaled_surface = pygame.transform.scale(dodecagon_surface, (300, 300))
        draw_transformed_dodecagon(scaled_surface)
    elif keys[pygame.K_2]:  # Option 2: Rotate 45 degrees from the center
        rotated_surface = pygame.transform.rotate(dodecagon_surface, 45)
        draw_transformed_dodecagon(rotated_surface)
    elif keys[pygame.K_3]:  # Option 3: Flip upside down
```

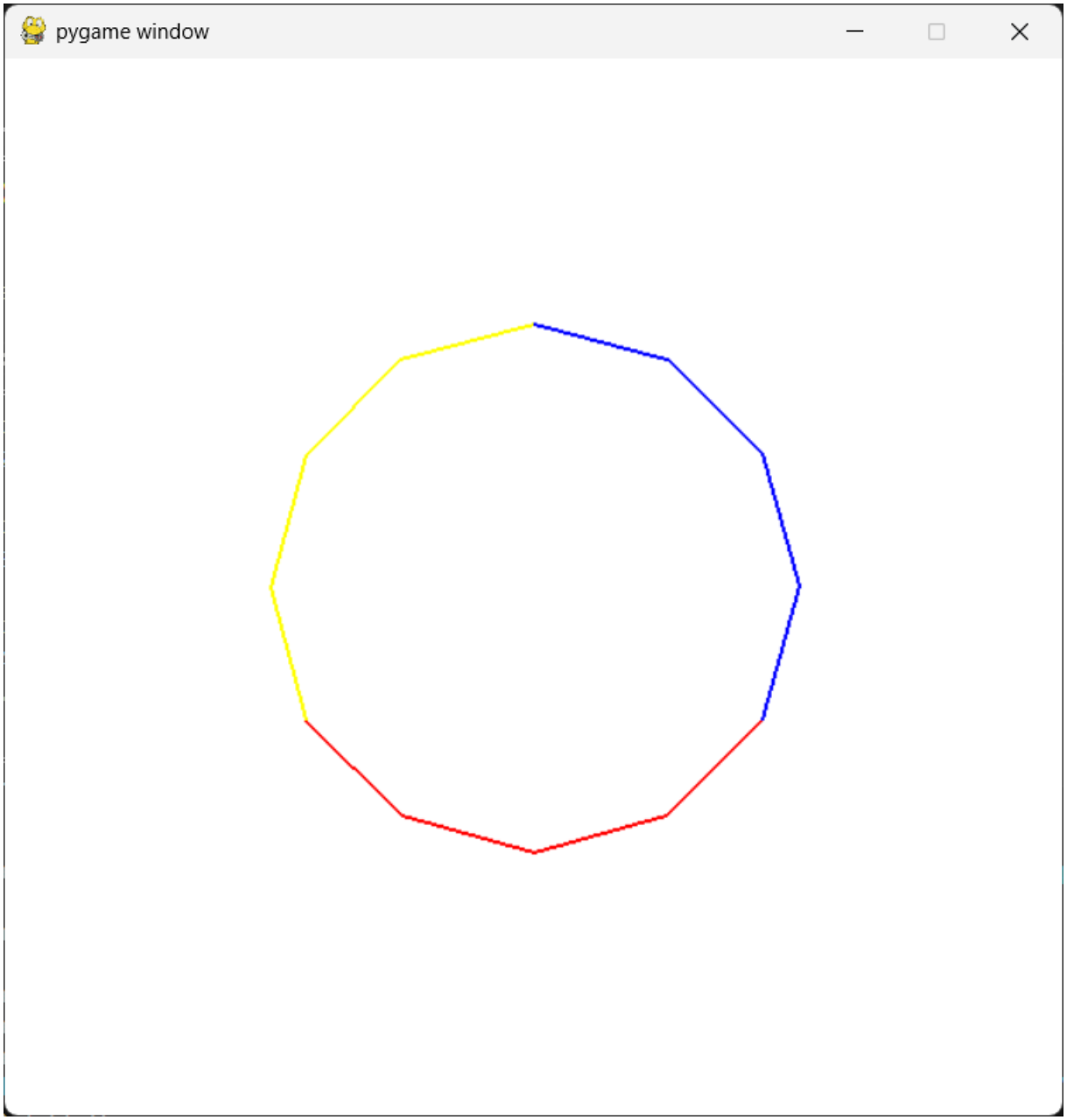
```
62.             flipped_surface = pygame.transform.flip(dodecagon_surface, False, True)
63.             draw_transformed_dodecagon(flipped_surface)
64.         elif keys[pygame.K_4]:  # Option 4: Make it lean
65.             lean_factor = 0.5
66.             leaned_points = [(x + lean_factor * y, y) for x, y in dodecagon_points]
67.             leaned_surface = pygame.Surface((600, 600), pygame.SRCALPHA)
68.             draw_dodecagon(leaned_surface, leaned_points)
69.         elif keys[pygame.K_5]:  # Option 5: Move it to the top
70.             moved_points = [(x, y - 150) for x, y in dodecagon_points]
71.             moved_surface = pygame.Surface((600, 600), pygame.SRCALPHA)
72.             draw_dodecagon(moved_surface, moved_points)
73.         elif keys[pygame.K_6]:  # Option 6: Lean and rotate 90 degrees
74.             lean_factor = 0.5
75.             leaned_points = [(x + lean_factor * y, y) for x, y in dodecagon_points]
76.             leaned_surface = pygame.Surface((600, 600), pygame.SRCALPHA)
77.             draw_dodecagon(leaned_surface, leaned_points)
78.             rotated_surface = pygame.transform.rotate(leaned_surface, 90)
79.             draw_transformed_dodecagon(rotated_surface)
80.         elif keys[pygame.K_7]:  # Option 7: Flip upside down and horizontally
81.             flipped_surface = pygame.transform.flip(dodecagon_surface, True, True)
82.             draw_transformed_dodecagon(flipped_surface)
83.         elif keys[pygame.K_8]:  # Option 8: Rotate 45 degrees from the center and reduce y by
half
84.             rotated_surface = pygame.transform.rotate(dodecagon_surface, 45)
85.             reduced_surface = pygame.transform.scale(rotated_surface, (600, 300))
86.             draw_transformed_dodecagon(reduced_surface)
87.         elif keys[pygame.K_9]:  # Option 9: Lean, flip upside down, flip horizontally
88.             lean_factor = 0.5
89.             leaned_points = [(x + lean_factor * y, y) for x, y in dodecagon_points]
90.             leaned_surface = pygame.Surface((600, 600), pygame.SRCALPHA)
91.             draw_dodecagon(leaned_surface, leaned_points)
92.             flipped_surface = pygame.transform.flip(leaned_surface, True, True)
93.             draw_transformed_dodecagon(flipped_surface)
94.
```

Program rysuje wielokąt i w zależności od kliknięte go numeru na klawiaturze przekształca on figurę tak by odwzorowywała podaną funkcję z przykładu.

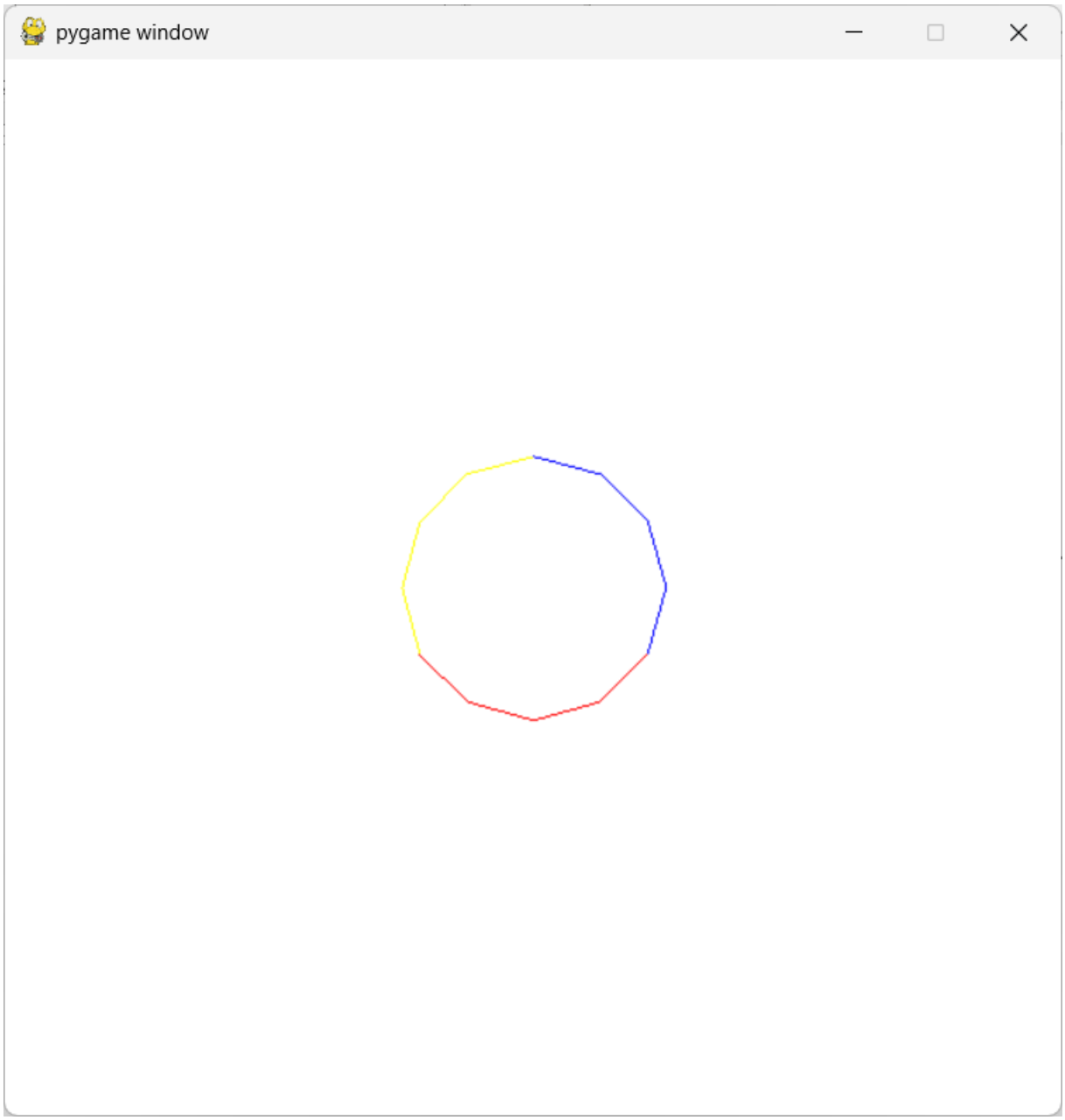
4. Wyniki działania

Transformacje obrazka:

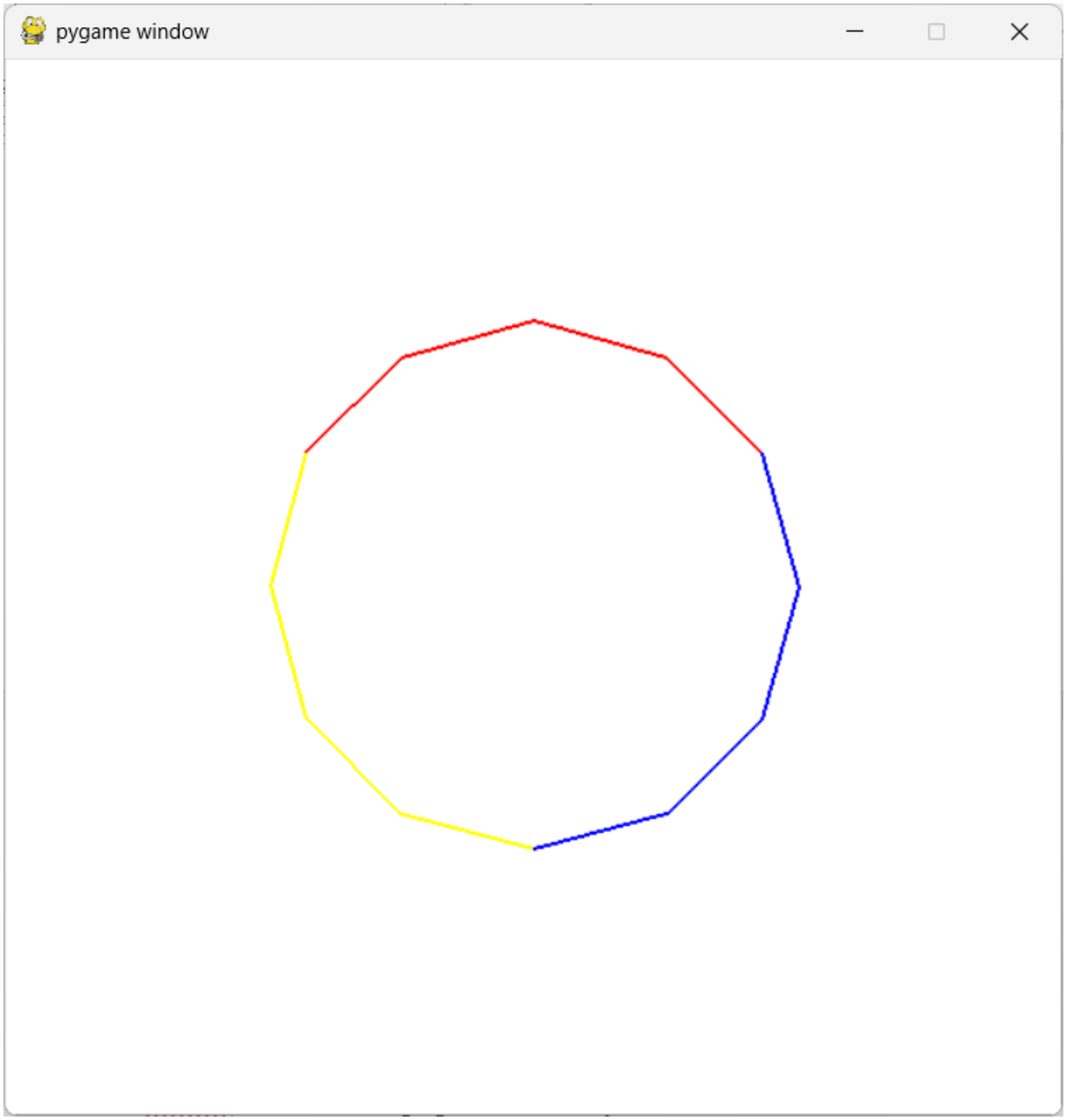
4.0. Obrazek Domyślny



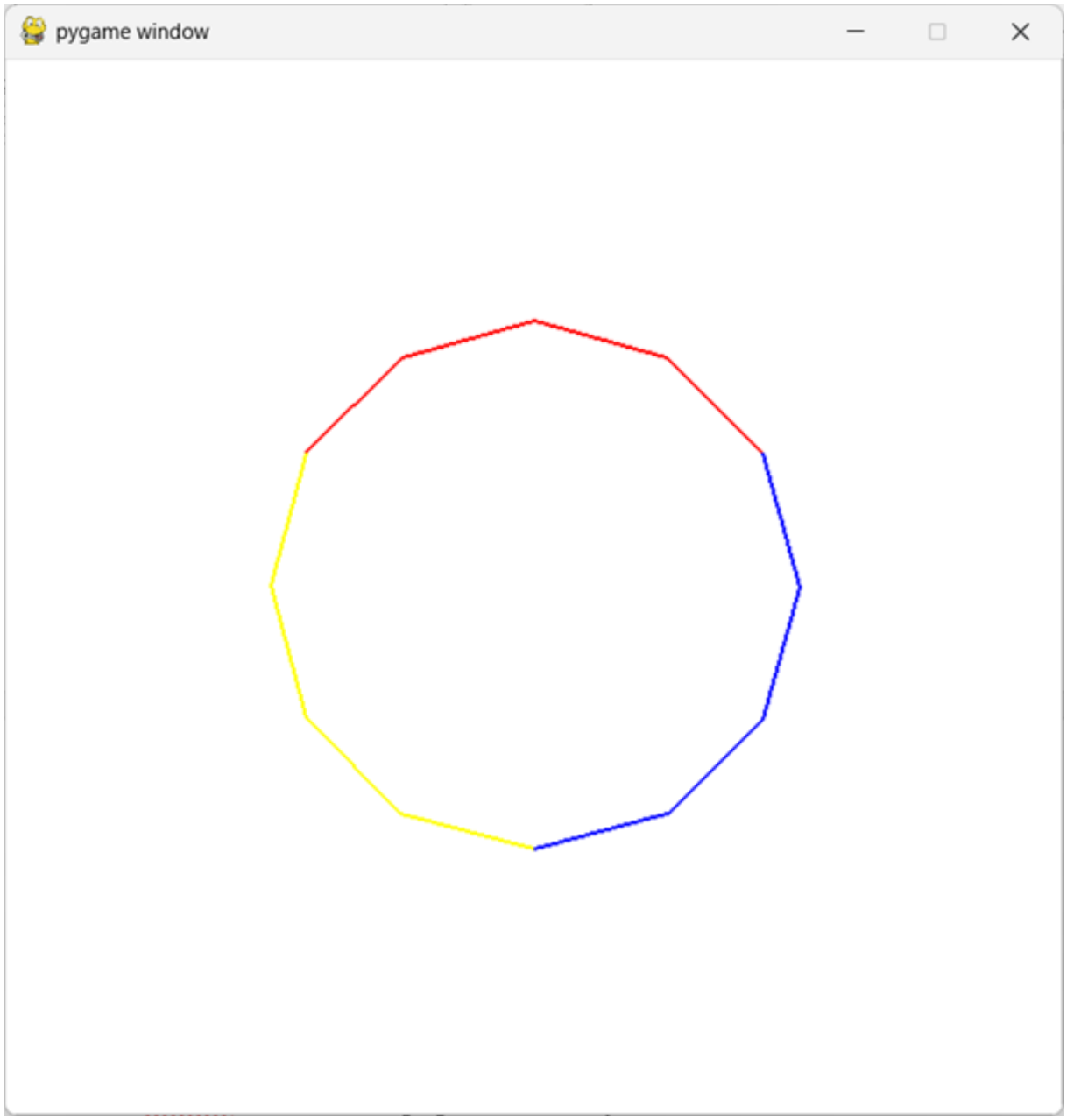
4.1. Pomniejszenie



4.2. Obrót o 45 stopni

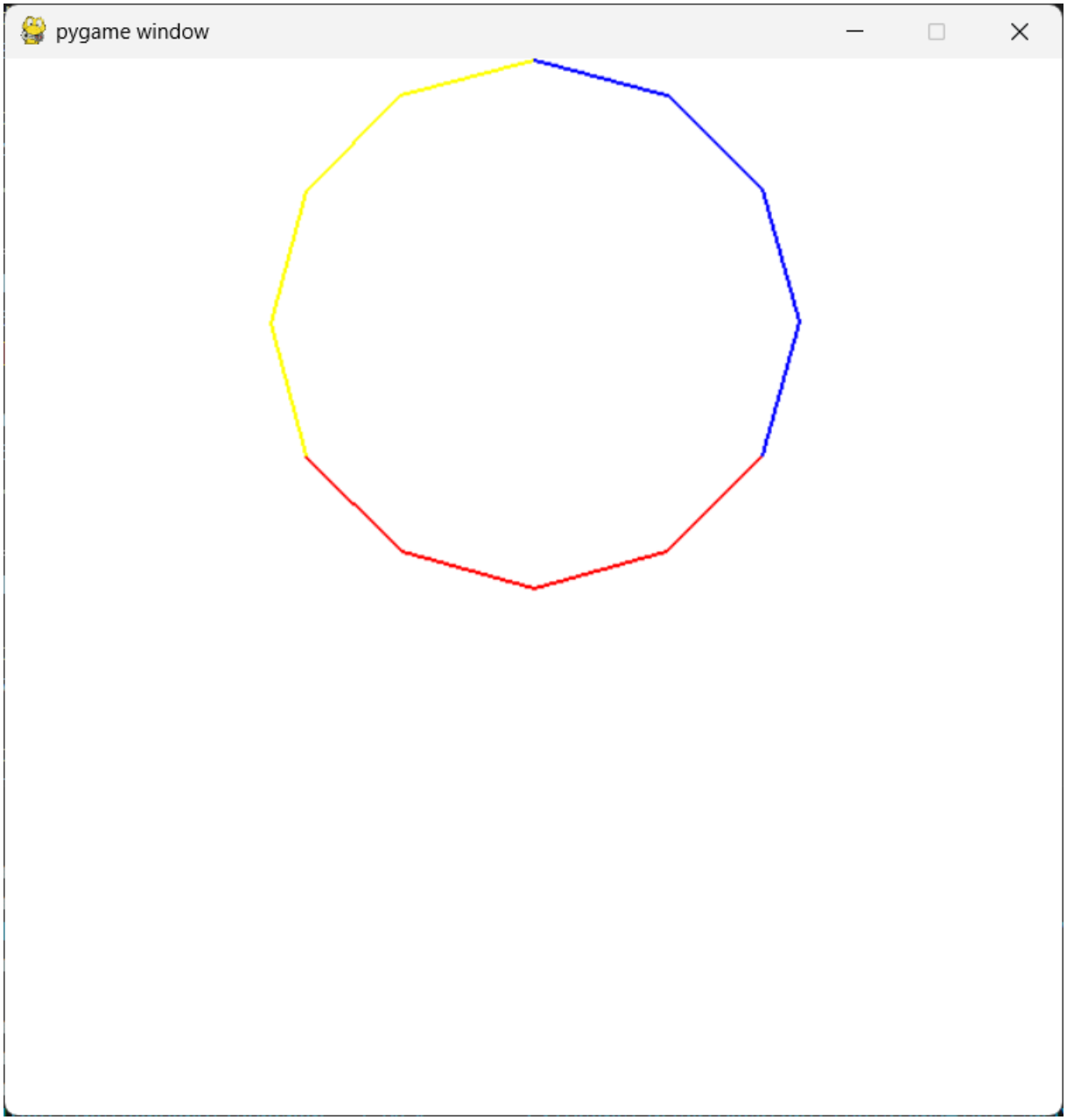


4.3. Obrót do góry nogami

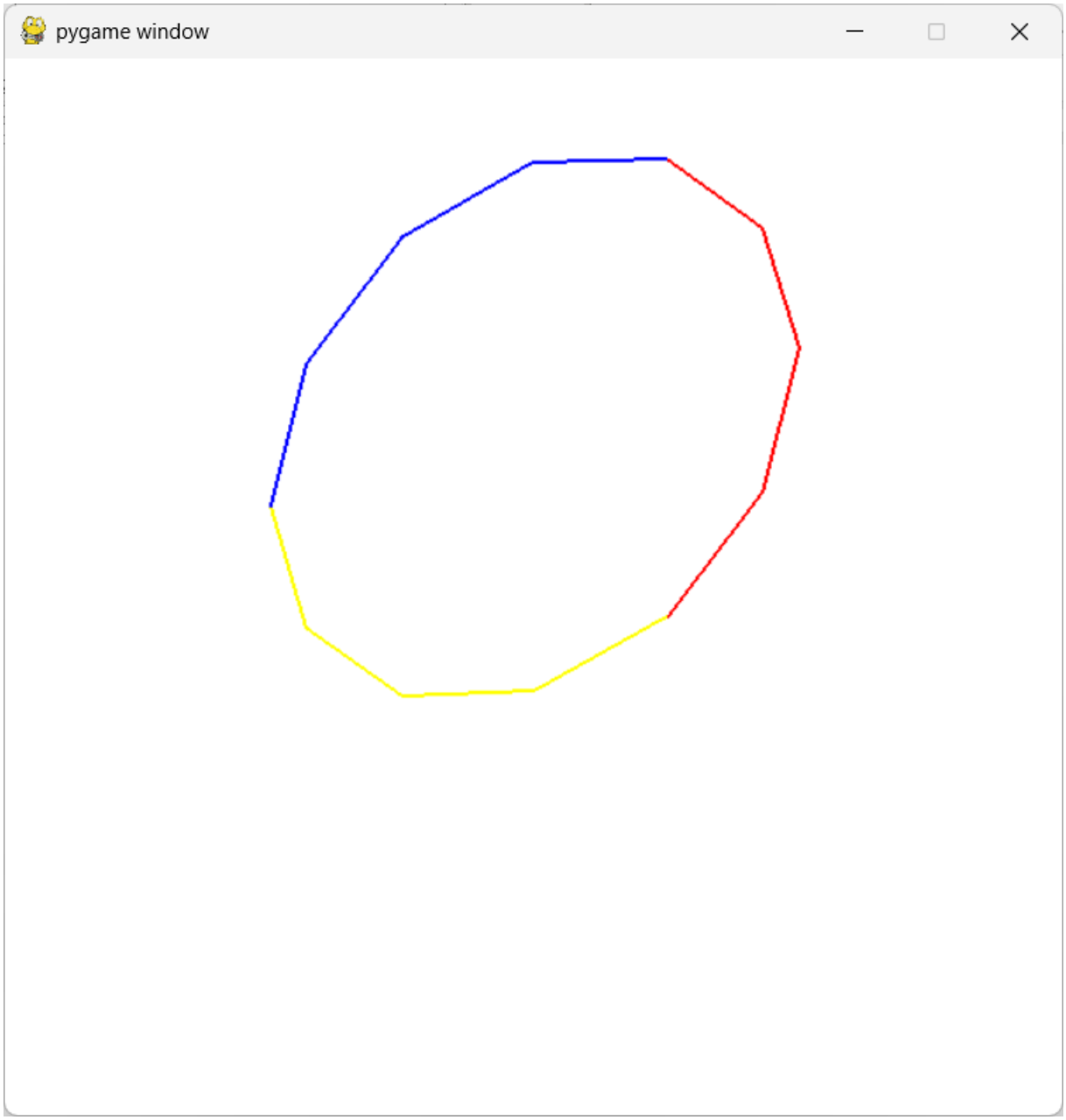


4.4. Pochylenie

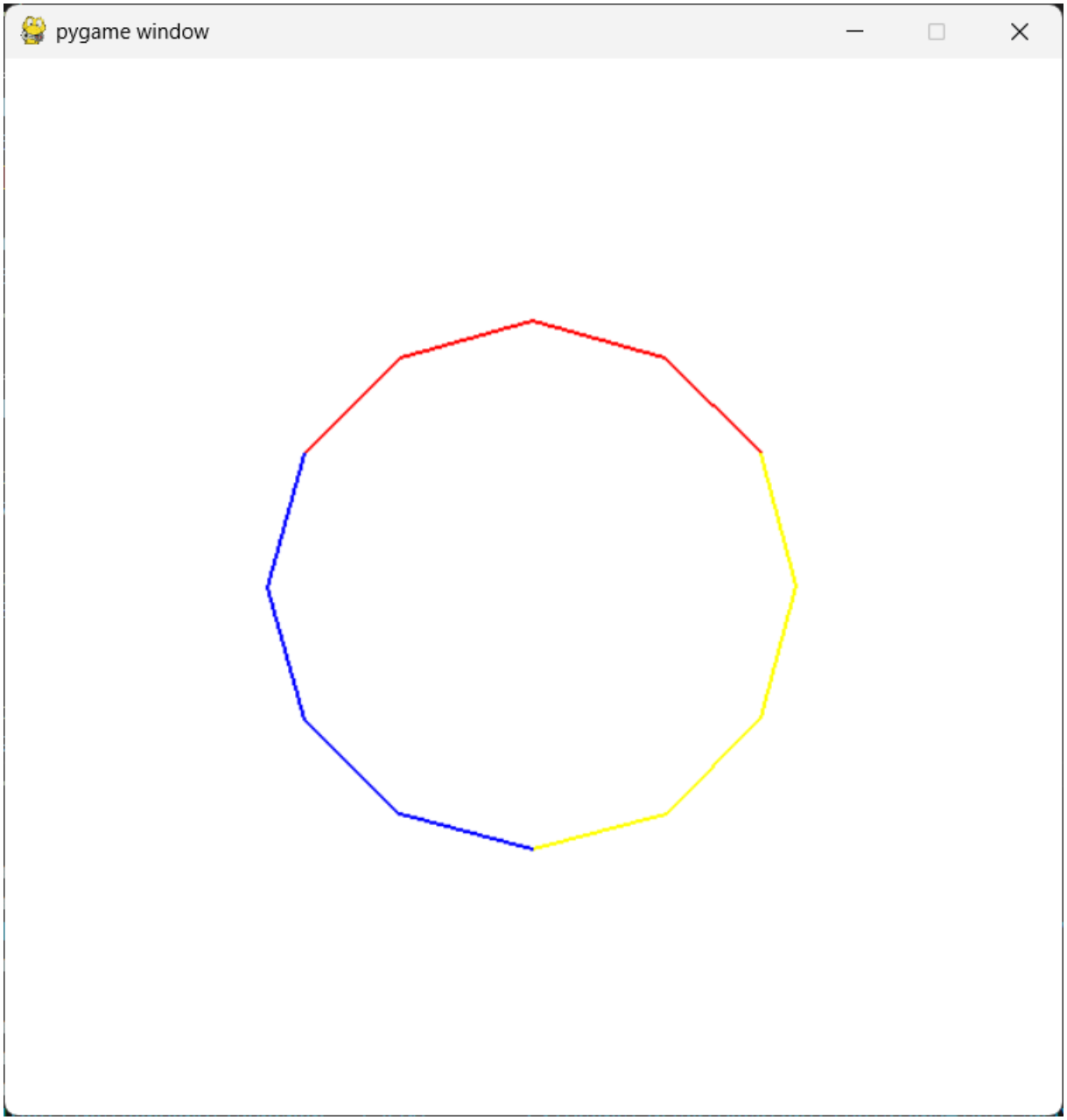
4.5. Przestawienie do górnej granicy



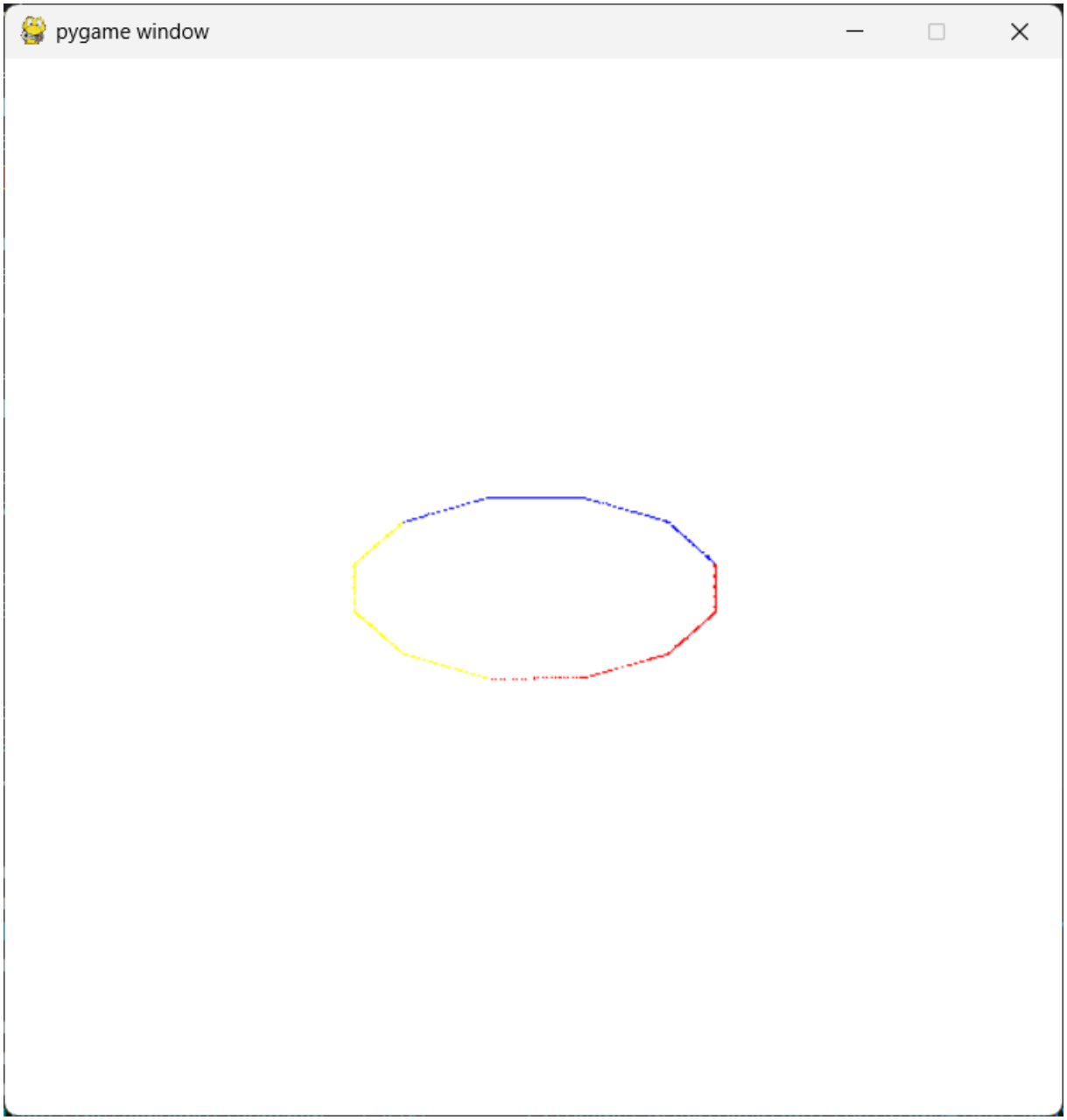
4.6. Pochylenie i obrót o 90 stopni



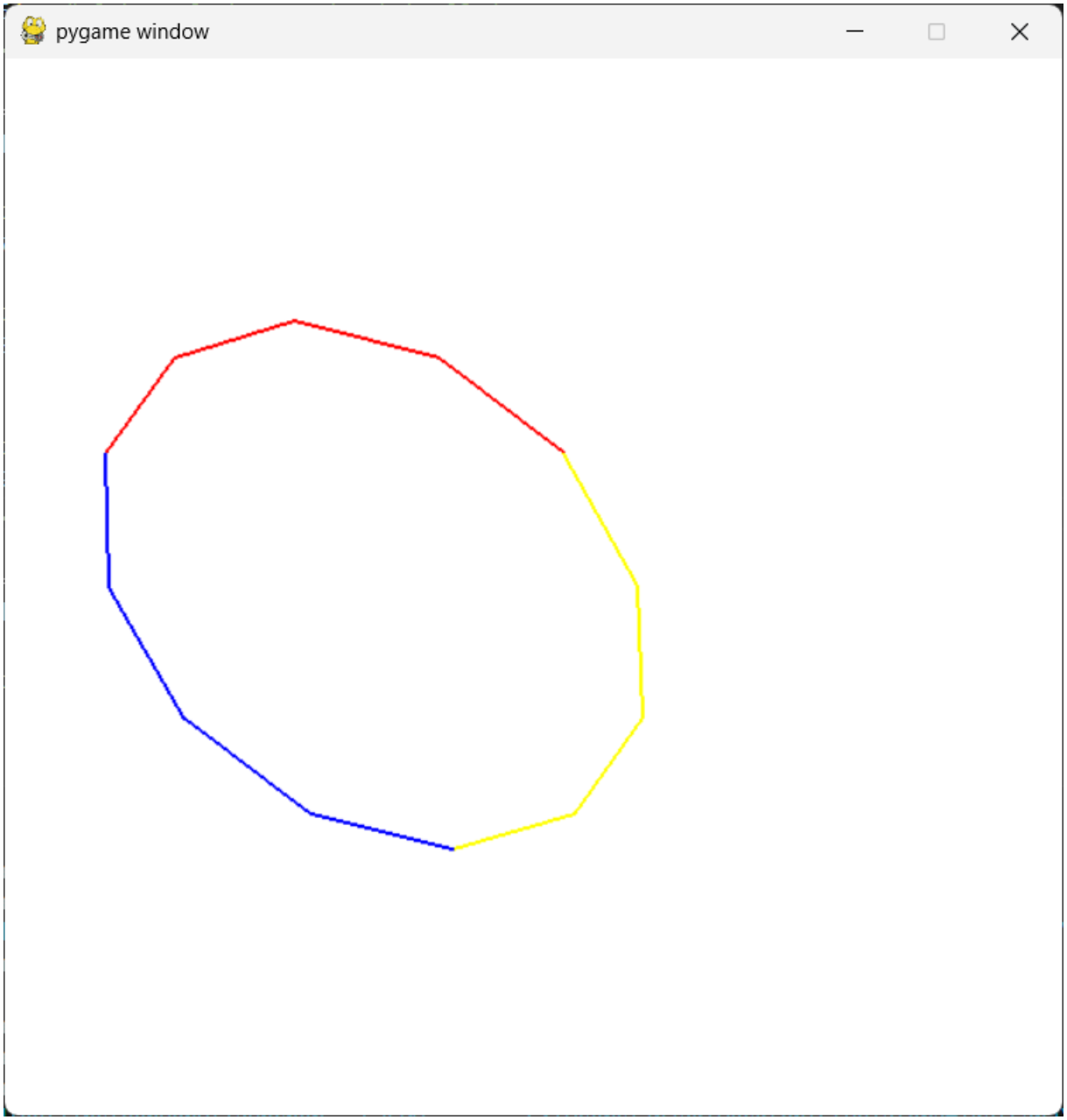
4.7. Obrót w pionie i poziomie



4.8. Obrót o 45 stopni i zmniejszenie wysokości (y) o połowę



4.9. Pochylenie i obrót pionowy i poziomy



5. Wnioski:

Pygame to potężna biblioteka do tworzenia grafiki 2D w języku Python, która oferuje narzędzia do rysowania i manipulowania obrazami. Dzięki niej można w prosty sposób rysować różne kształty, takie jak linie, prostokąty, wielokąty, okręgi czy elipsy. Pygame udostępnia zaawansowane funkcje transformacji grafiki, umożliwiające zmianę rozmiaru obiektów (skalowanie), ich obracanie pod dowolnym kątem oraz odbijanie w pionie i poziomie.