**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,

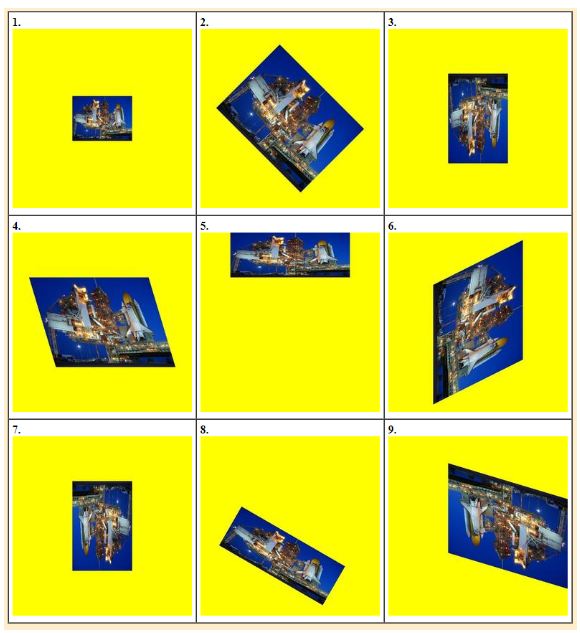
4 semestr,

Gr.3b

# 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.

# Wprowadzane dane:



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

# Wykorzystane komendy:

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

1. import pygame

2. import math

3.

4. pygame.init()

5. screen = pygame.display.set\_mode((600, 600))

6. screen.fill((255, 255, 255))

7.

8. # Dodecagon points

9. def calculate\_dodecagon\_points(center, radius):

10.     points = []

11.     for i in range(12):

12.         angle = math.radians(30 + i \* 30)

13.         x = center[0] + radius \* math.cos(angle)

14.         y = center[1] + radius \* math.sin(angle)

15.         points.append((x, y))

16.     return points

17.

18. center = (300, 300)

19. radius = 150

20. dodecagon\_points = calculate\_dodecagon\_points(center, radius)

21.

22. # Function to draw the dodecagon with  colors

23. def draw\_dodecagon(surface, points):

24.     surface.fill((255, 255, 255, 0))

25.     colors = [(255, 0, 0), (255, 255, 0), (0, 0, 255)]  # Red, Yellow, Blue

26.     for i in range(len(points)):

27.         start\_point = points[i]

28.         end\_point = points[(i + 1) % len(points)]

29.         color = colors[i // 4]

30.         pygame.draw.line(surface, color, start\_point, end\_point, 2)

31.     screen.fill((255, 255, 255))

32.     screen.blit(surface, (0, 0))

33.     pygame.display.flip()

34.

35. # Surface for the dodecagon

36. dodecagon\_surface = pygame.Surface((600, 600), pygame.SRCALPHA)

37. draw\_dodecagon(dodecagon\_surface, dodecagon\_points)

38.

39. # Function to draw the transformed dodecagon centered on the screen

40. def draw\_transformed\_dodecagon(transformed\_surface):

41.     screen.fill((255, 255, 255))

42.     rect = transformed\_surface.get\_rect(center=(300, 300))

43.     screen.blit(transformed\_surface, rect.topleft)

44.     pygame.display.flip()

45.

46. run = True

47. while run:

48.     for event in pygame.event.get():

49.         if event.type == pygame.QUIT:

50.             pygame.quit()

51.             exit(0)

52.

53.     keys = pygame.key.get\_pressed()

54.

55.     if keys[pygame.K\_1]:  # Option 1: Make it smaller

56.         scaled\_surface = pygame.transform.scale(dodecagon\_surface, (300, 300))

57.         draw\_transformed\_dodecagon(scaled\_surface)

58.     elif keys[pygame.K\_2]:  # Option 2: Rotate 45 degrees from the center

59.         rotated\_surface = pygame.transform.rotate(dodecagon\_surface, 45)

60.         draw\_transformed\_dodecagon(rotated\_surface)

61.     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)

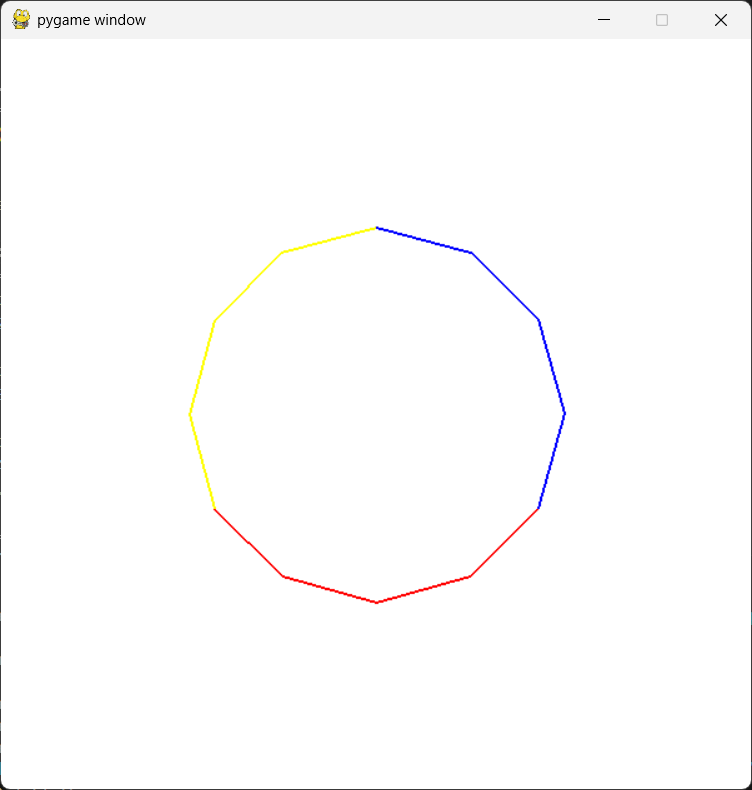
94.

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

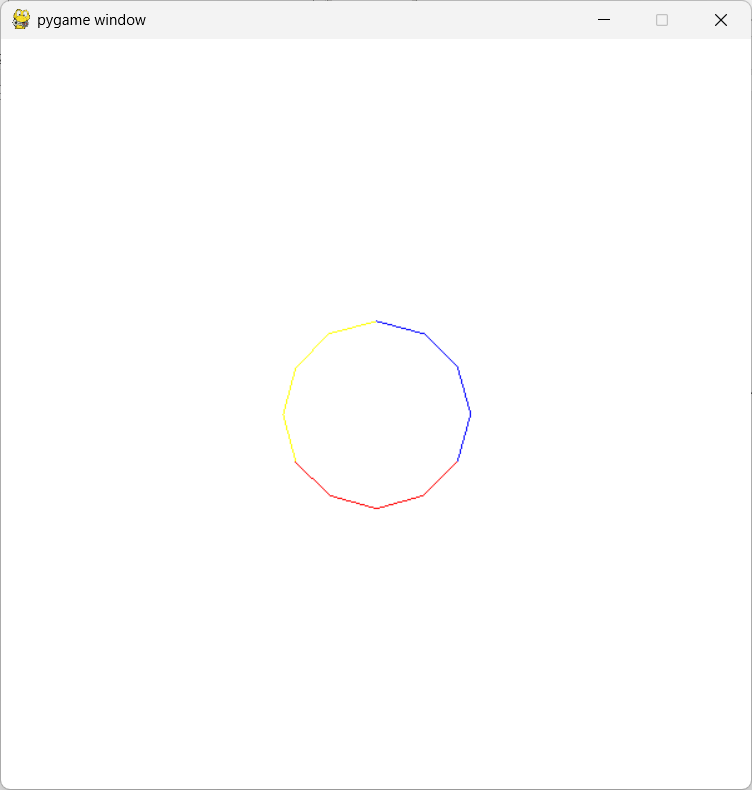
# Wyniki działania

Transformacje obrazka:

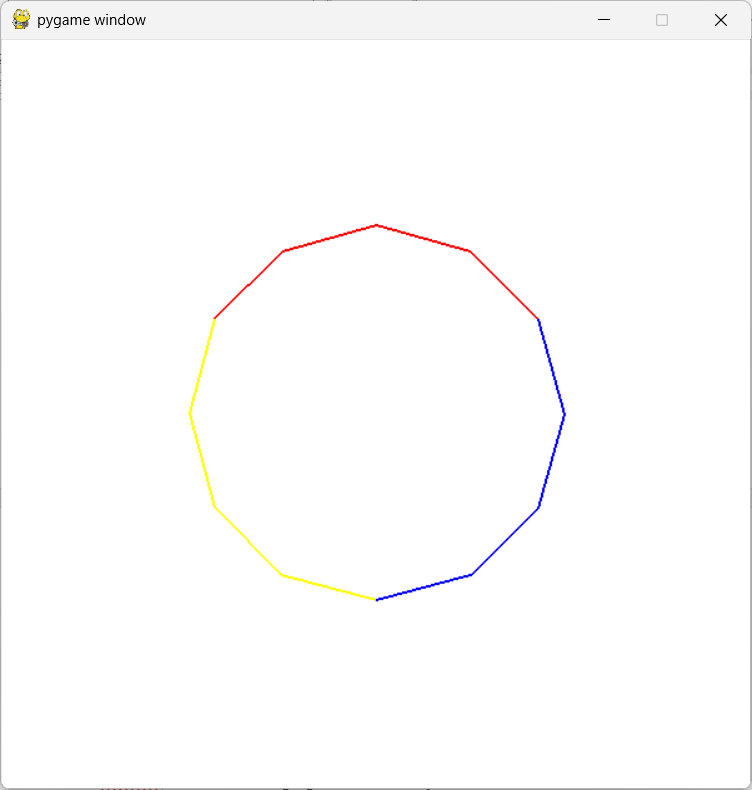
## Obrazek Domyślny



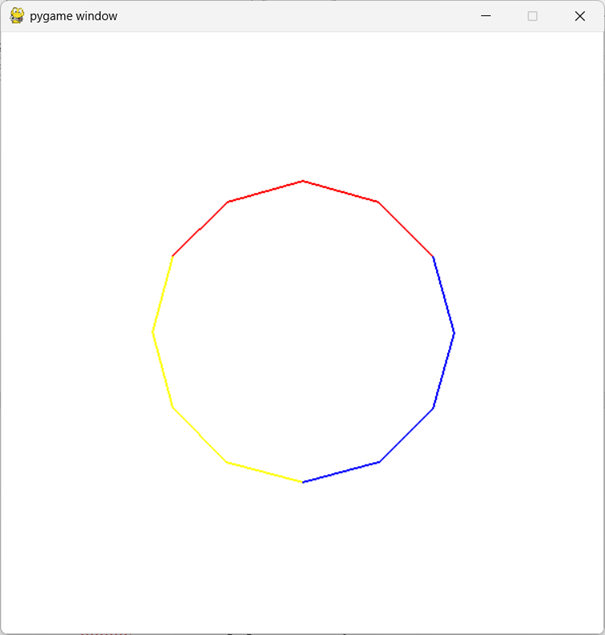
## Pomniejszenie



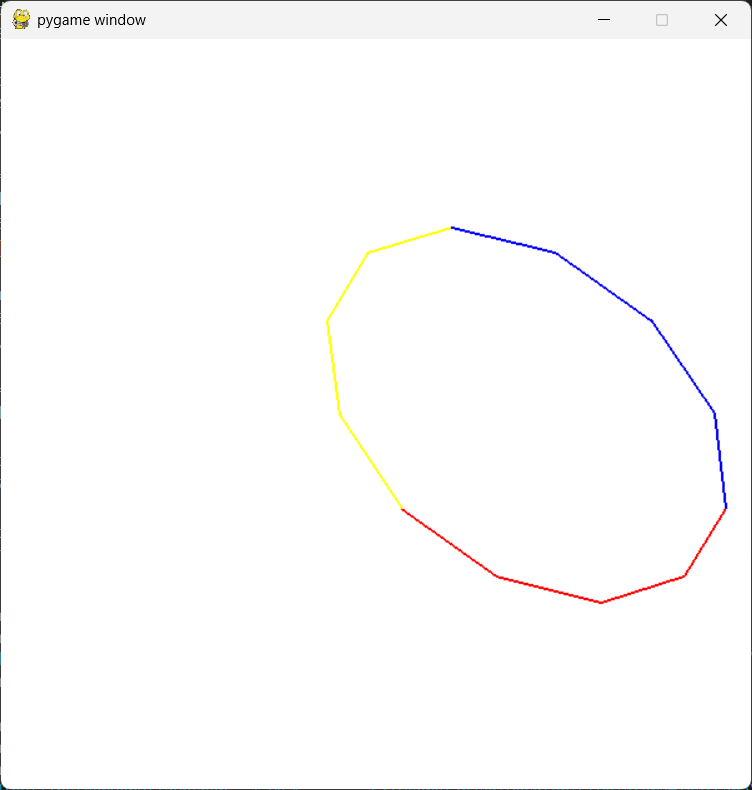
## Obrót o 45 stopni



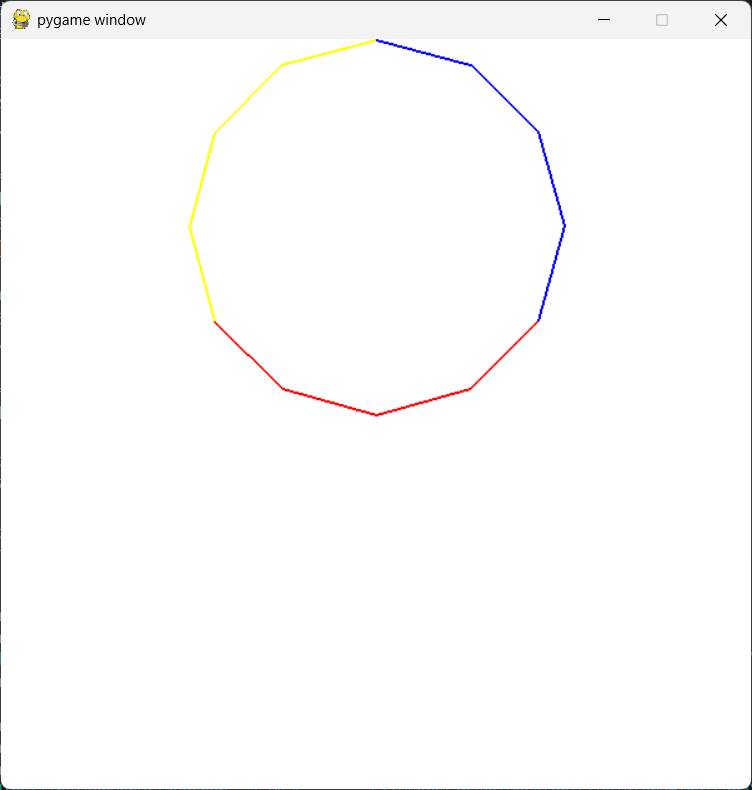
## Obrót do góry nogami



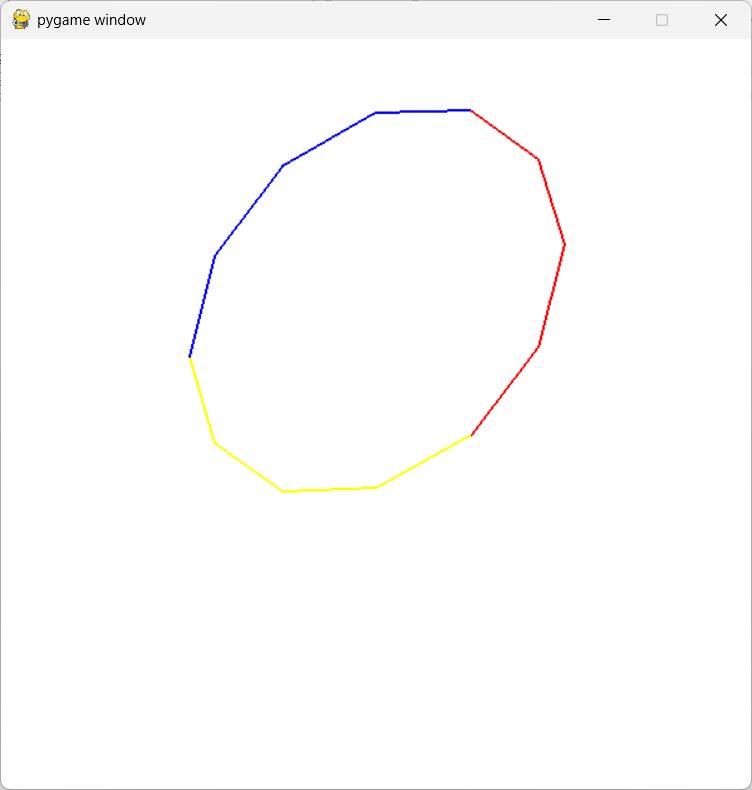
## Pochylenie



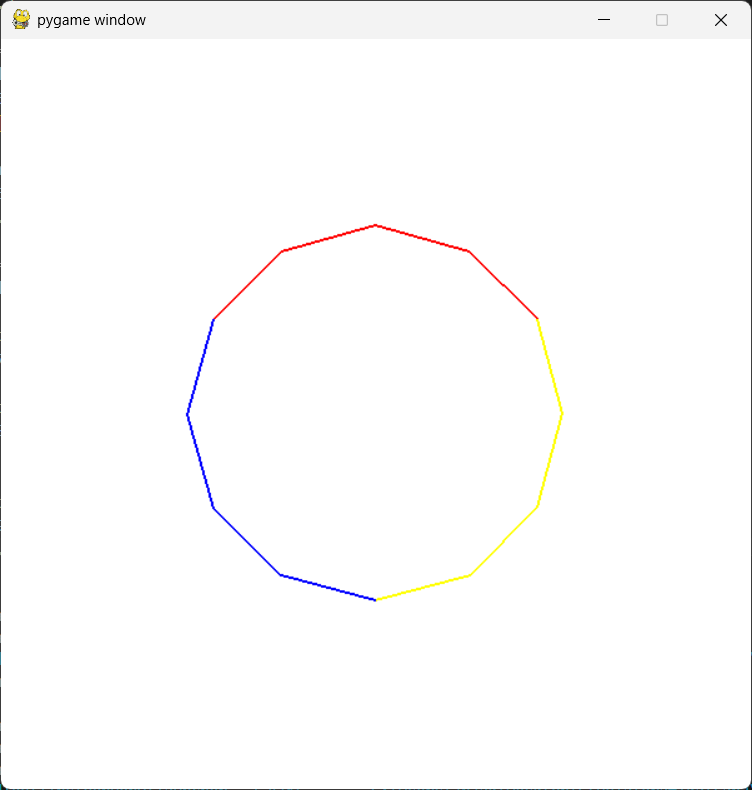
## Przestawienie do górnej granicy



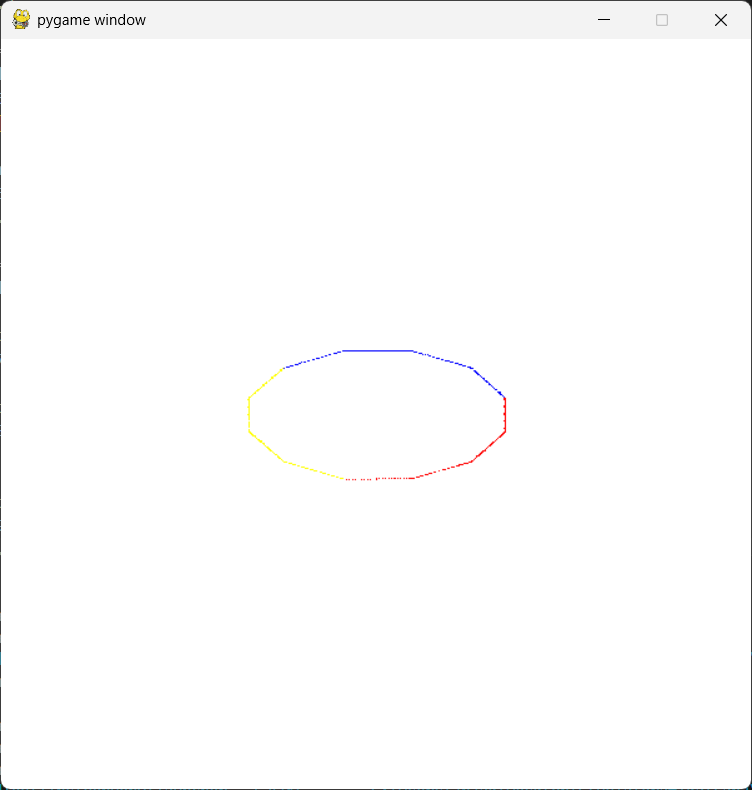
## Pochylenie i obrót o 90 stopni



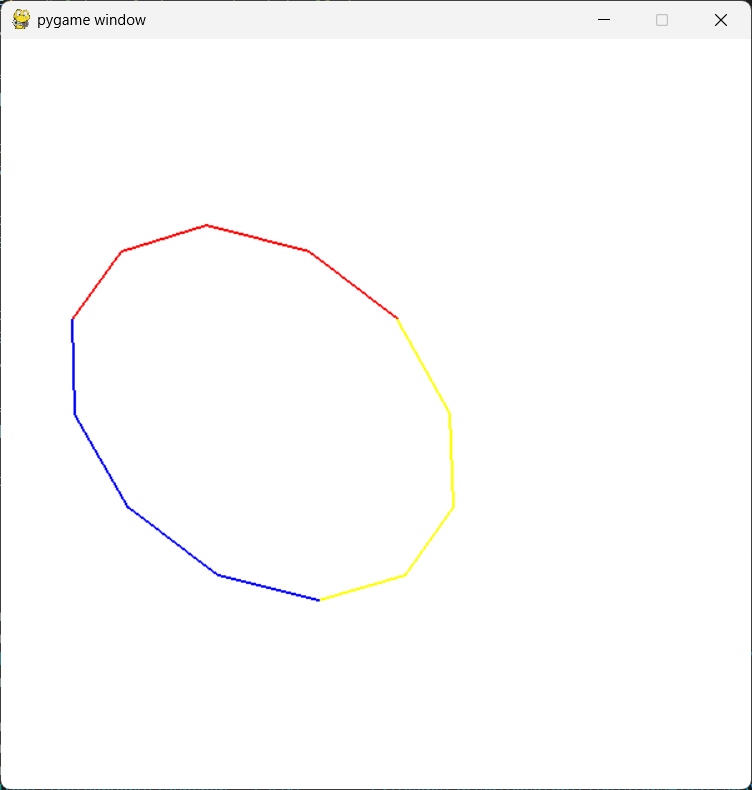
## Obrót w pionie i poziomie



## Obrót o 45 stopni i zmniejszenie wysokości (y) o połowę



## Pochylenie i obrót pionowy i poziomy



# 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.