**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: mgr Mikołaj Grygiel

**Laboratorium 3**

10.03.2024

**Temat:** Przekształcenia 2D z użyciem PYGAME

**Wariant**: 1

Radosław Skrzypczyński

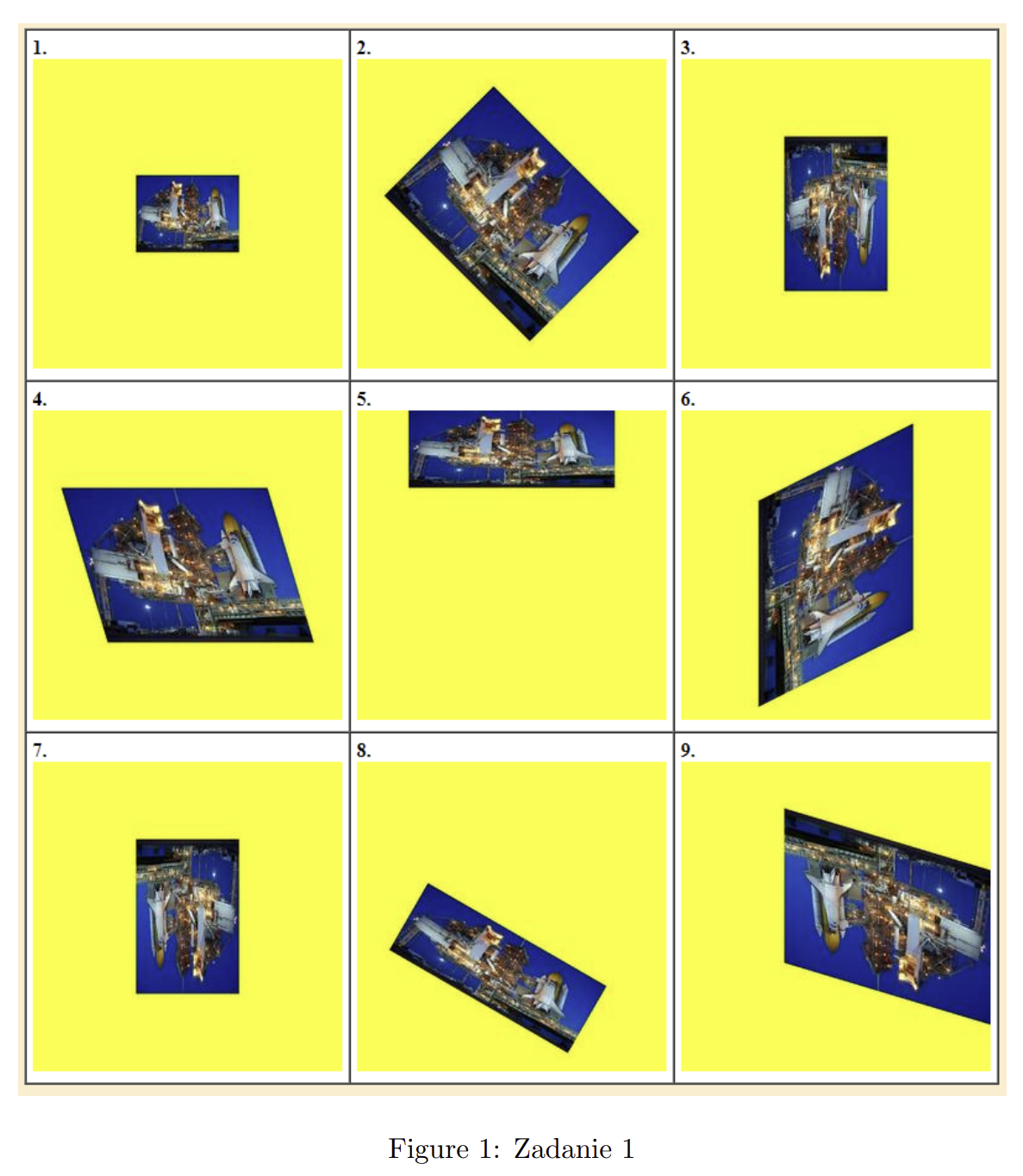
Informatyka I stopień,

stacjonarne,

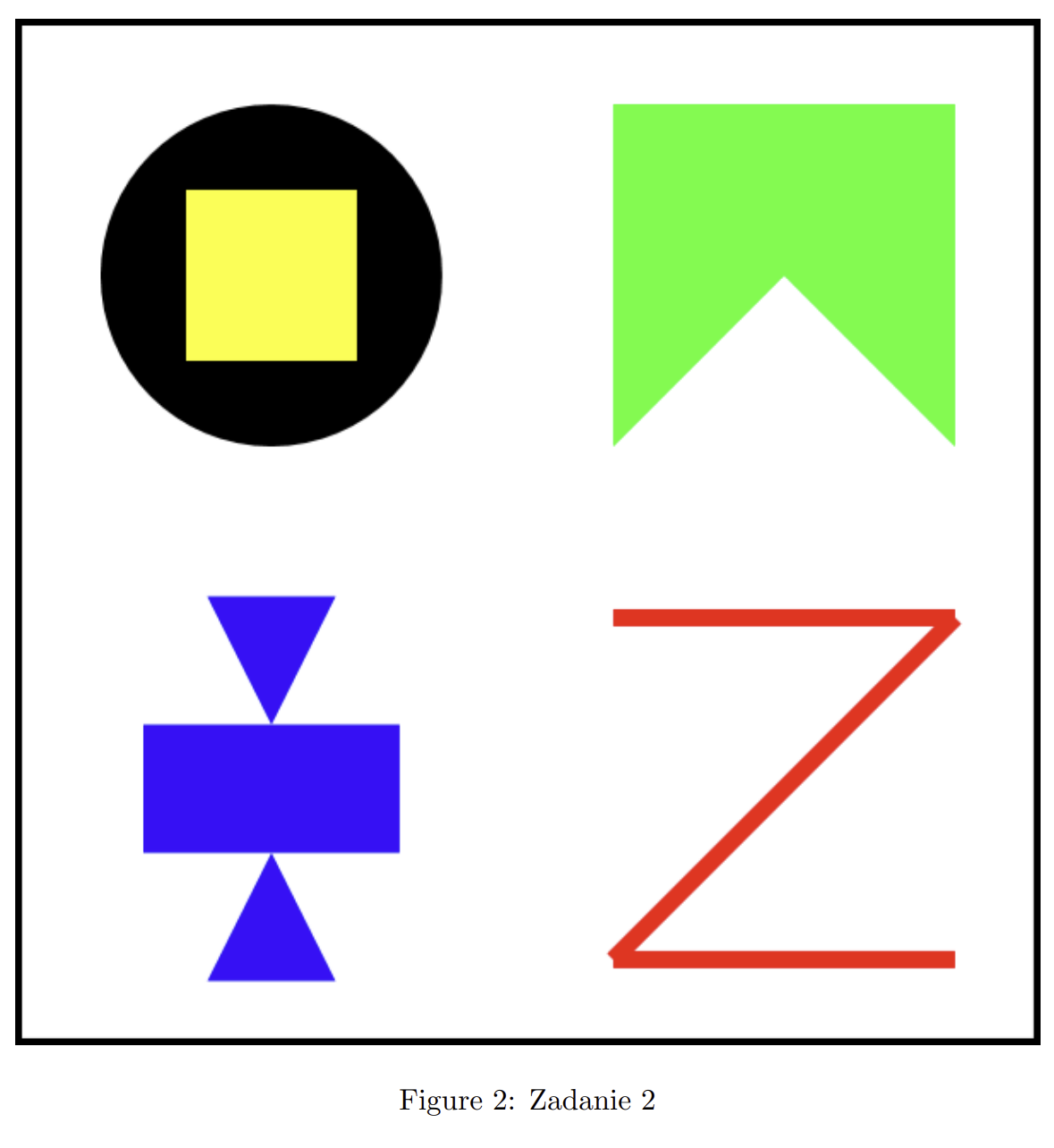
4 semestr,

Gr. 1B

1. **Polecenie:**
2. Pokazany jest obraz shuttle.jpg w panelu. Narysowa´c zamiast obrazu wielokat wed lug wariantu (liczba n). Okno ma wymiary 600 na 600 pikseli, a wielokat ma promie´n 150 pikseli. Kolejne zadanie polega na stosowaniu odpowiednich przekszta lce´n do wielokata (lub bedziesz potrzebowa l kombinacji przekszta lce´n) po naci´snieciu na klawisze od 1 do 9 (patrz Fig. 1).

****

1. Narysowa´c figure okre´slona wariantem (patrz Fig. 2). Dostepne sa trzy podstawowe kszta lty: ko lo, kwadrat, tr´ojkat. Podstawowe przekszta lcenia dostepne sa przez pygame.transform

****

**2. Wprowadzane dane:**

**Zadanie 1:**

import math  
import pygame  
  
window\_size = (600, 600)  
radius = 150  
vertices\_count = 5  
  
pygame.init()  
  
window = pygame.display.set\_mode(window\_size)  
pygame.display.set\_caption("Pięciokąt foremny")  
clock = pygame.time.Clock()  
  
#color declarations  
WHITE = (0, 0, 0)  
RED = (255, 0, 0)  
GREEN = (0, 255, 0)  
YELLOW = (255, 255, 0)  
VIOLET = (128, 0, 128)  
LIGHT\_BLUE = (0, 255, 255)  
ORANGE = (255, 165, 0)  
BLUE = (0, 0, 255)  
GRAY = (128, 128, 128)  
  
center\_x = window\_size[0] // 2  
center\_y = window\_size[1] // 2  
  
original\_vertices = [(center\_x + int(radius \* math.sin(2 \* math.pi \* i / vertices\_count)),  
 center\_y + int(radius \* math.cos(2 \* math.pi \* i / vertices\_count)))  
 for i in range(vertices\_count)]  
  
test = []  
for i in range(vertices\_count):  
 x = center\_x + int(radius \* math.sin(2 \* math.pi \* i / vertices\_count))  
 y = center\_y + int(radius \* math.cos(2 \* math.pi \* i / vertices\_count))  
 test.append((x, y))  
 print(x)  
 print(y)  
  
  
  
  
vertices = original\_vertices.copy()  
pressed\_button = 1  
  
def rotateVertices(verticesToRotate, angle):  
 verticesToRotate = [(2 \* x - center\_x, 2 \* y - center\_y) for x, y in verticesToRotate]  
 angleRad = math.radians(angle)  
 verticesToRotate = [(int((x - center\_x) \* math.cos(angleRad) - (y - center\_y) \* math.sin(angleRad) + center\_x),  
 int((x - center\_x) \* math.sin(angleRad) + (y - center\_y) \* math.cos(angleRad) + center\_y))  
 for x, y in verticesToRotate]  
 return verticesToRotate  
  
  
angle = 0  
display\_polygon = True  
button\_number = None  
font = pygame.font.Font(None, 36)  
isRunning = True  
  
while isRunning:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 running = False  
 elif event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_1:  
 vertices = original\_vertices.copy()  
 display\_polygon = True  
 button\_number = 1  
 elif event.key == pygame.K\_2:  
 vertices = [(2 \* x - center\_x, 2 \* y - center\_y) for x, y in original\_vertices]  
 angle = math.radians(45)  
 vertices = [(int((x - center\_x) \* math.cos(angle) - (y - center\_y) \* math.sin(angle) + center\_x),  
 int((x - center\_x) \* math.sin(angle) + (y - center\_y) \* math.cos(angle) + center\_y))  
 for x, y in vertices]  
 display\_polygon = True  
 button\_number = 2  
 elif event.key == pygame.K\_3:  
 vertices = [(x, 2 \* center\_y - y) for x, y in original\_vertices]  
 vertices = [(x, int(2 \* (y - center\_y) + center\_y)) for x, y in vertices]  
 display\_polygon = True  
 button\_number = 3  
 elif event.key == pygame.K\_4:  
 shear\_factor = 0.4  
 vertices = [(x + shear\_factor \* y, y) for x, y in original\_vertices]  
  
 min\_x = min(vertices, key=lambda point: point[0])[0]  
 max\_x = max(vertices, key=lambda point: point[0])[0]  
 shift\_x = (min\_x + max\_x) / 2 - center\_x  
 vertices = [(x - shift\_x, y) for x, y in vertices]  
  
 display\_polygon = True  
 button\_number = 4  
 elif event.key == pygame.K\_5:  
 scale\_factor\_x = 2  
 scale\_factor\_y = 1  
 translation\_y = -158  
  
 vertices = [(scale\_factor\_x \* x, scale\_factor\_y \* y) for x, y in original\_vertices]  
  
 vertices = [(x, y + translation\_y) for x, y in vertices]  
  
 min\_x = min(vertices, key=lambda point: point[0])[0]  
 max\_x = max(vertices, key=lambda point: point[0])[0]  
 shift\_x = (min\_x + max\_x) / 2 - center\_x  
 vertices = [(x - shift\_x, y) for x, y in vertices]  
  
 display\_polygon = True  
 button\_number = 5  
 elif event.key == pygame.K\_6:  
 shear\_factor\_y = -0.4  
 angle\_rotation = 0  
  
 vertices = [(x, x \* shear\_factor\_y + y) for x, y in original\_vertices]  
  
 angle\_rad = math.radians(angle\_rotation)  
 vertices = [(int((x - center\_x) \* math.cos(angle\_rad) - (y - center\_y) \* math.sin(angle\_rad) + center\_x),  
 int((x - center\_x) \* math.sin(angle\_rad) + (y - center\_y) \* math.cos(angle\_rad) + center\_y))  
 for x, y in vertices]  
  
 min\_y = min(vertices, key=lambda point: point[1])[1]  
 max\_y = max(vertices, key=lambda point: point[1])[1]  
 shift\_y = (min\_y + max\_y) / 2 - center\_y  
 vertices = [(x, y - shift\_y) for x, y in vertices]  
  
 display\_polygon = True  
 button\_number = 6  
 elif event.key == pygame.K\_7:  
 vertices = [(x, 2 \* center\_y - y) for x, y in original\_vertices]  
 vertices = [(x, int(2 \* (y - center\_y) + center\_y)) for x, y in vertices]  
 display\_polygon = True  
 button\_number = 7  
 elif event.key == pygame.K\_8:  
 scale\_factor\_x = 2  
 scale\_factor\_y = 1  
 angle\_rotation = 30  
 translation\_y = -50  
 translation\_x = -250  
  
 vertices = [(scale\_factor\_x \* x, scale\_factor\_y \* y) for x, y in original\_vertices]  
  
 angle\_rad = math.radians(angle\_rotation)  
 vertices = [(int((x - center\_x) \* math.cos(angle\_rad) - (y - center\_y) \* math.sin(angle\_rad) + center\_x),  
 int((x - center\_x) \* math.sin(angle\_rad) + (y - center\_y) \* math.cos(angle\_rad) + center\_y))  
 for x, y in vertices]  
  
 vertices = [(x, y + translation\_y) for x, y in vertices]  
  
 vertices = [(x + translation\_x, y) for x, y in vertices]  
  
 display\_polygon = True  
 button\_number = 8  
 elif event.key == pygame.K\_9:  
 angle\_rotation = 180  
 shear\_factor = 0.3  
 translation\_x = 165  
  
 angle\_rad = math.radians(angle\_rotation)  
 vertices = [(int((x - center\_x) \* math.cos(angle\_rad) - (y - center\_y) \* math.sin(angle\_rad) + center\_x),  
 int((x - center\_x) \* math.sin(angle\_rad) + (y - center\_y) \* math.cos(angle\_rad) + center\_y))  
 for x, y in original\_vertices]  
  
 vertices = [(x, x \* shear\_factor + y) for x, y in vertices]  
  
 vertices = [(x + translation\_x, y) for x, y in vertices]  
  
 display\_polygon = True  
 button\_number = 9  
  
 # Wypełnienie tła  
 window.fill(YELLOW)  
  
 # Rysowanie dziesięciokąta z ciemniejszym środkiem (jeśli zmienna display\_polygon jest True)  
 if display\_polygon:  
 pygame.draw.polygon(window, RED, vertices)  
 pygame.draw.polygon(window, RED, vertices, 0) # Wypełnienie środka  
 pygame.draw.polygon(window, BLUE, vertices, 2) # Rysowanie obramówki  
  
 # Rysowanie numeru przycisku w lewym górnym rogu  
 if button\_number is not None:  
 number\_text = font.render(str(button\_number), True, GRAY)  
 number\_rect = number\_text.get\_rect()  
 number\_rect.topleft = (10, 10)  
 pygame.draw.rect(window, YELLOW, number\_rect)  
 window.blit(number\_text, number\_rect.topleft)  
  
 # Aktualizacja ekranu  
 pygame.display.flip()  
  
# Zakończenie działania pygame  
pygame.quit()

**Zadanie 2:**

import pygame  
import sys  
  
# Initialize Pygame  
pygame.init()  
  
# Set up display  
window\_size = (400, 400)  
screen = pygame.display.set\_mode(window\_size)  
pygame.display.set\_caption("Draw Shapes")  
  
# Define colors  
BLACK = (0, 0, 0)  
YELLOW = (255, 255, 0)  
  
# Define shapes' dimensions  
circle\_center = (200, 200)  
circle\_radius = 150  
square\_size = 150  
square\_pos = (200 - square\_size // 2, 200 - square\_size // 2)  
  
# Main loop  
running = True  
while running:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 running = False  
  
 # Clear the screen  
 screen.fill((255, 255, 255))  
  
 # Draw a black circle  
 pygame.draw.circle(screen, BLACK, circle\_center, circle\_radius)  
  
 # Draw a yellow square inside the circle  
 pygame.draw.rect(screen, YELLOW, (\*square\_pos, square\_size, square\_size))  
  
 # Update the display  
 pygame.display.flip()  
  
# Quit Pygame  
pygame.quit()  
sys.exit()

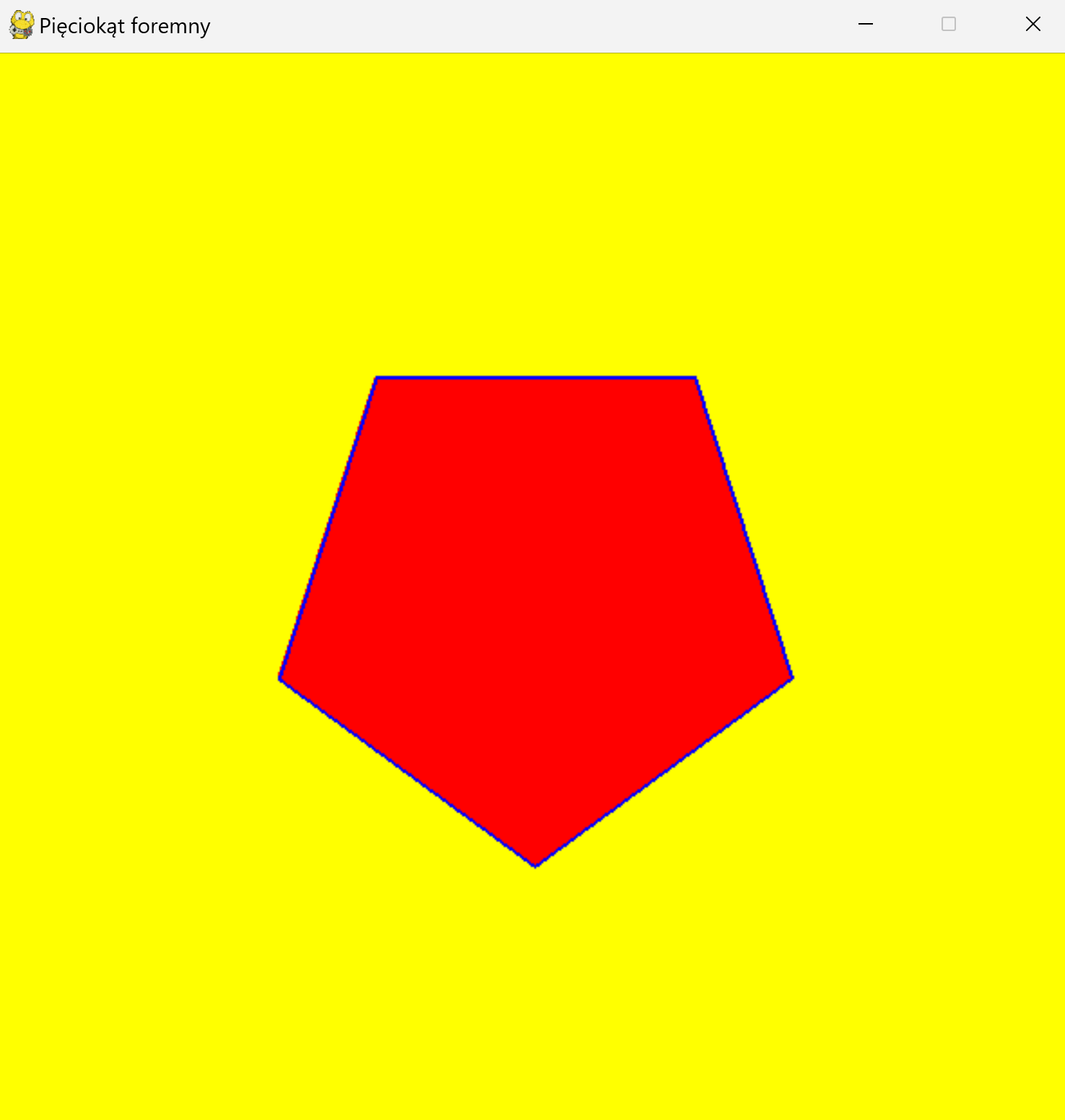
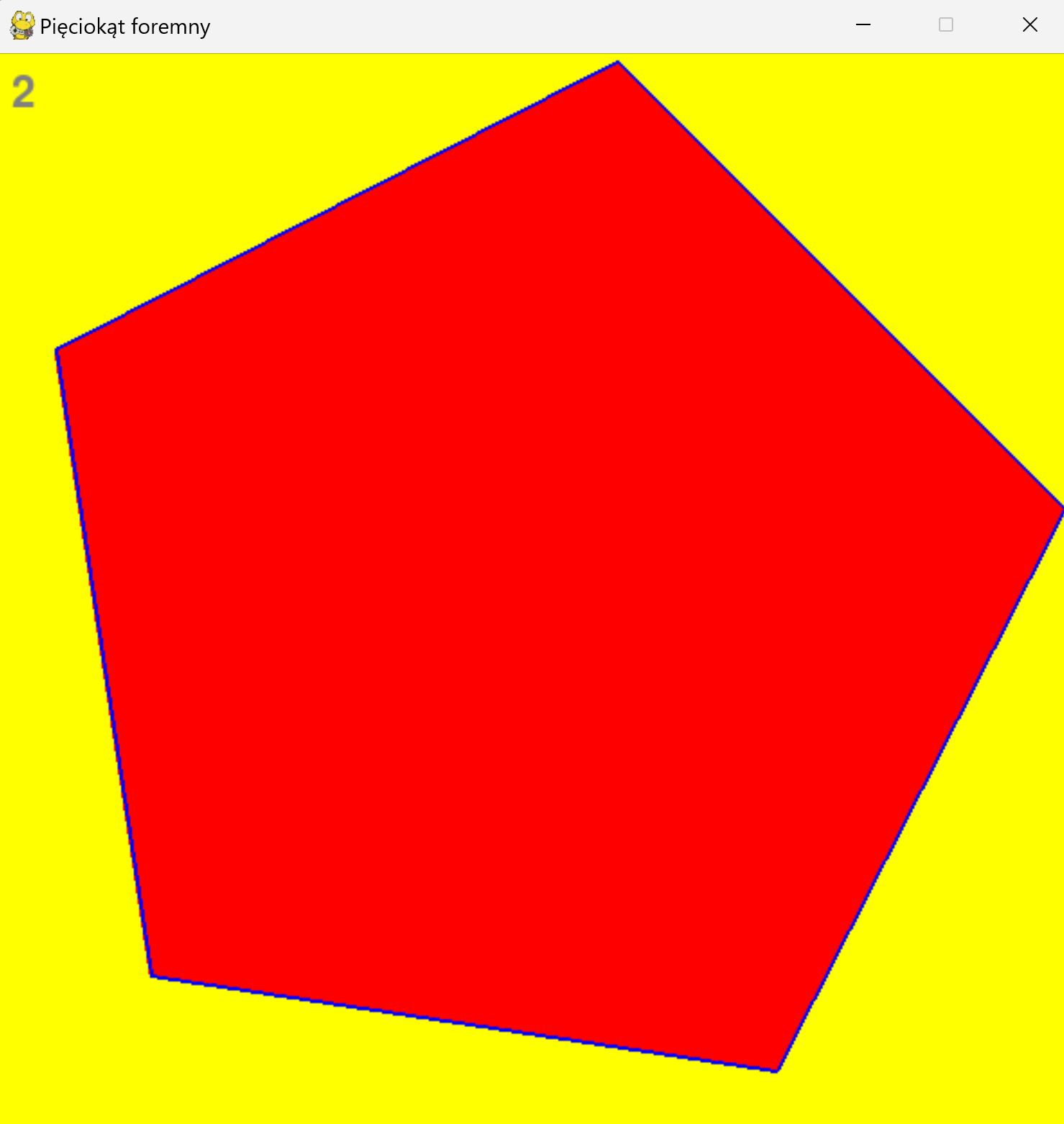
**3. Wykorzystane komendy:**

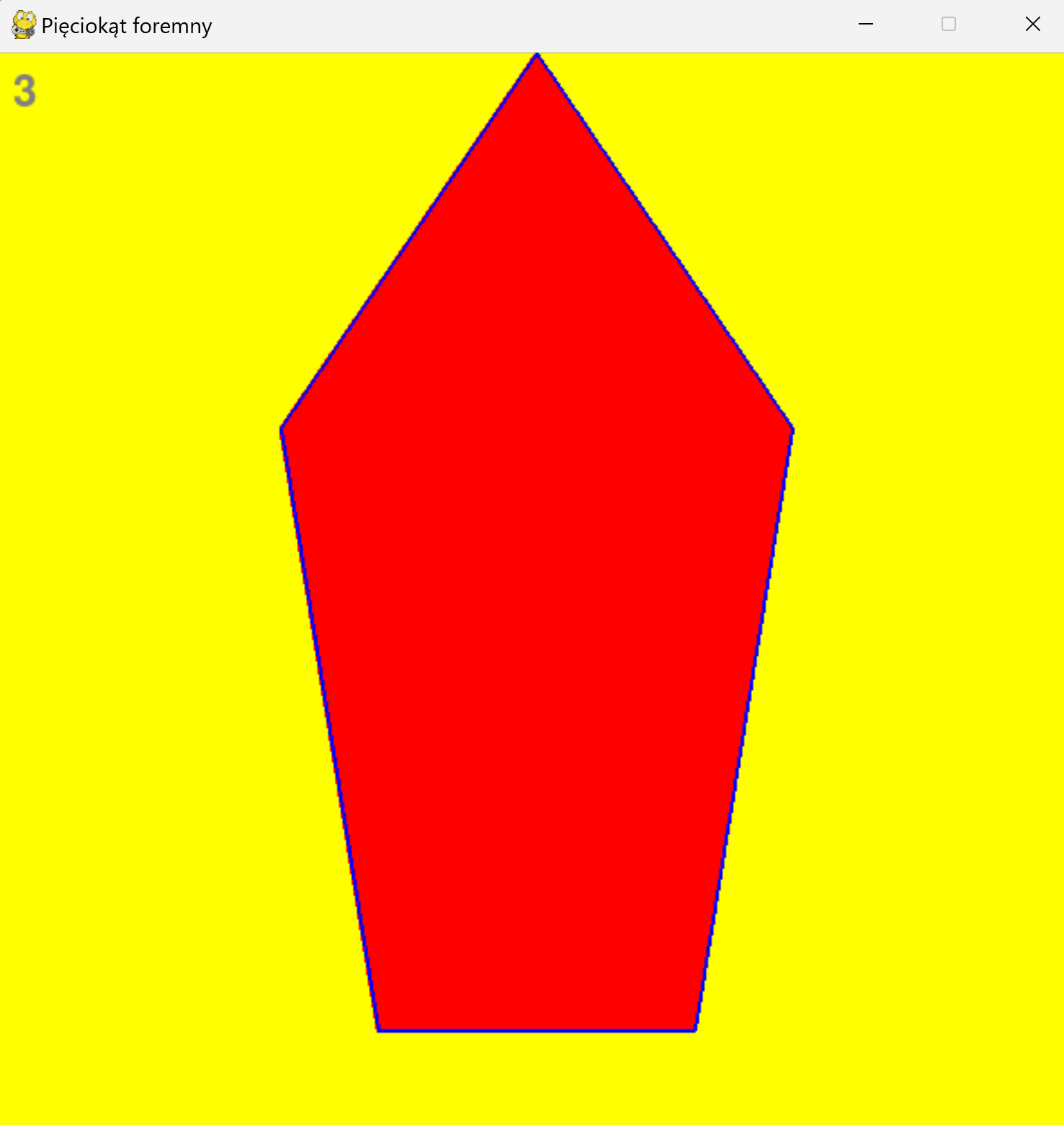
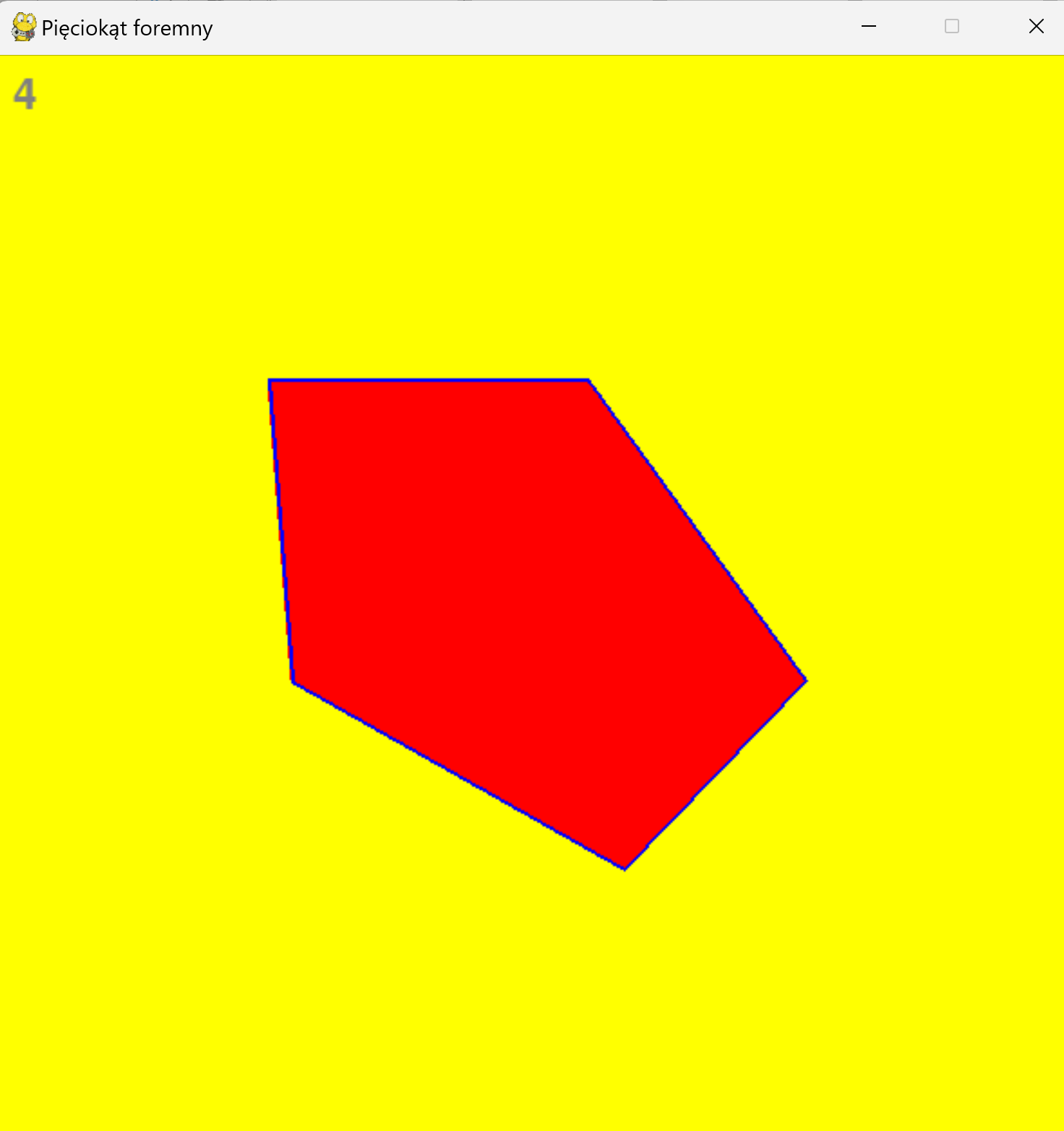
W zadaniu 1: klawisze od 1-9

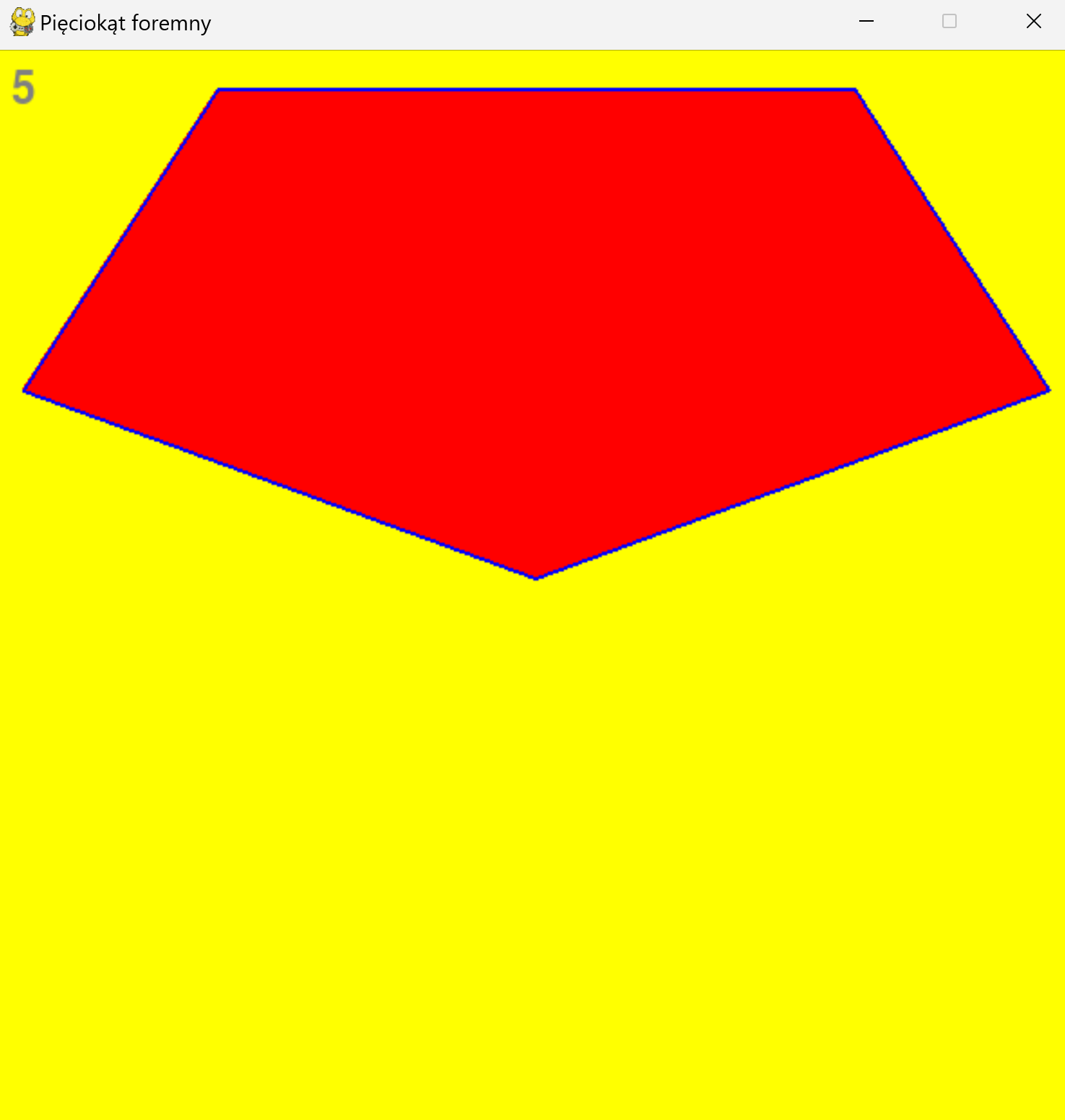
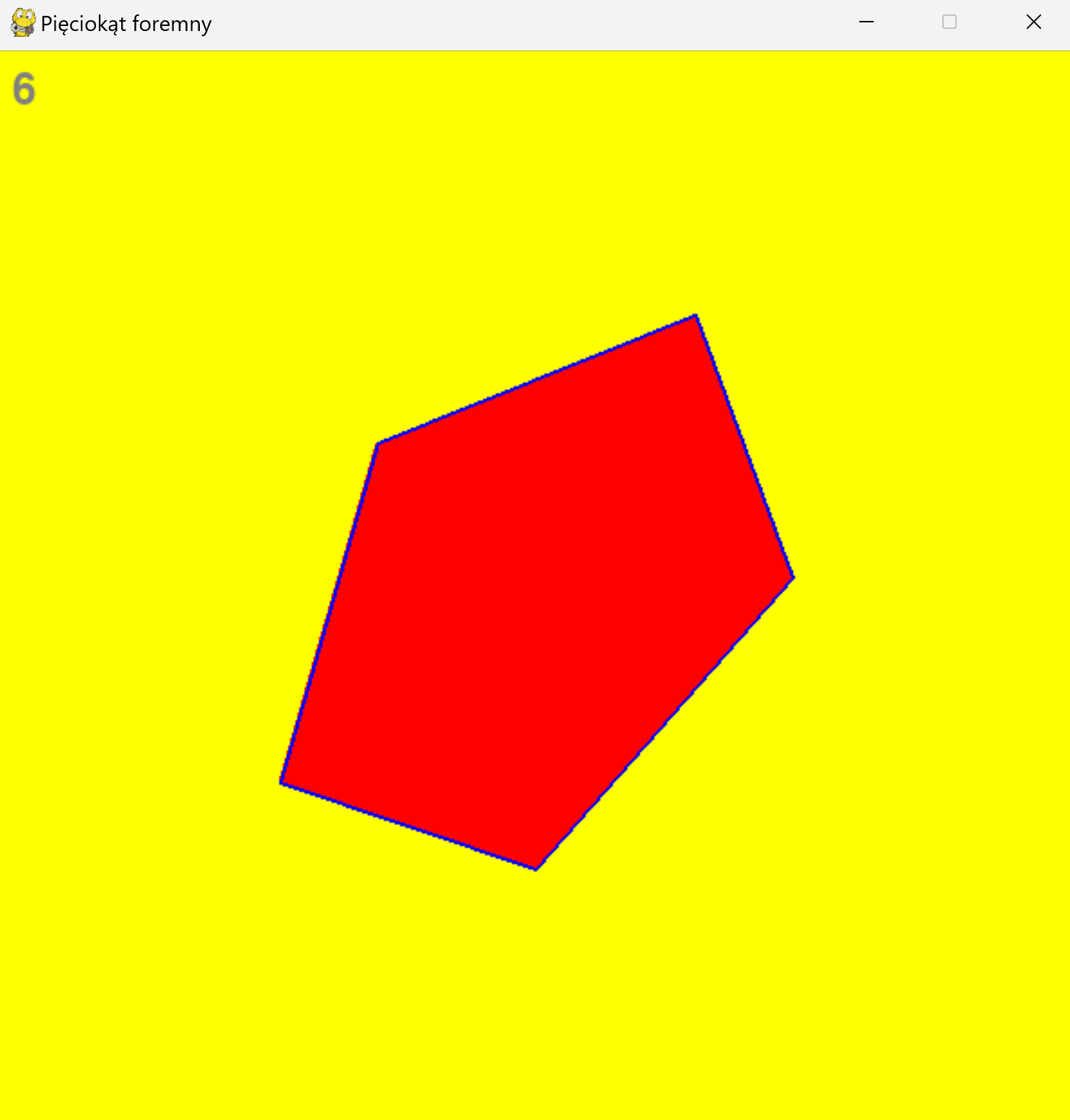
Link do zdalnego repozytorium: https://github.com/Slayzerus/UBB\_GrafikaKomputerowa/tree/main/Lab%302

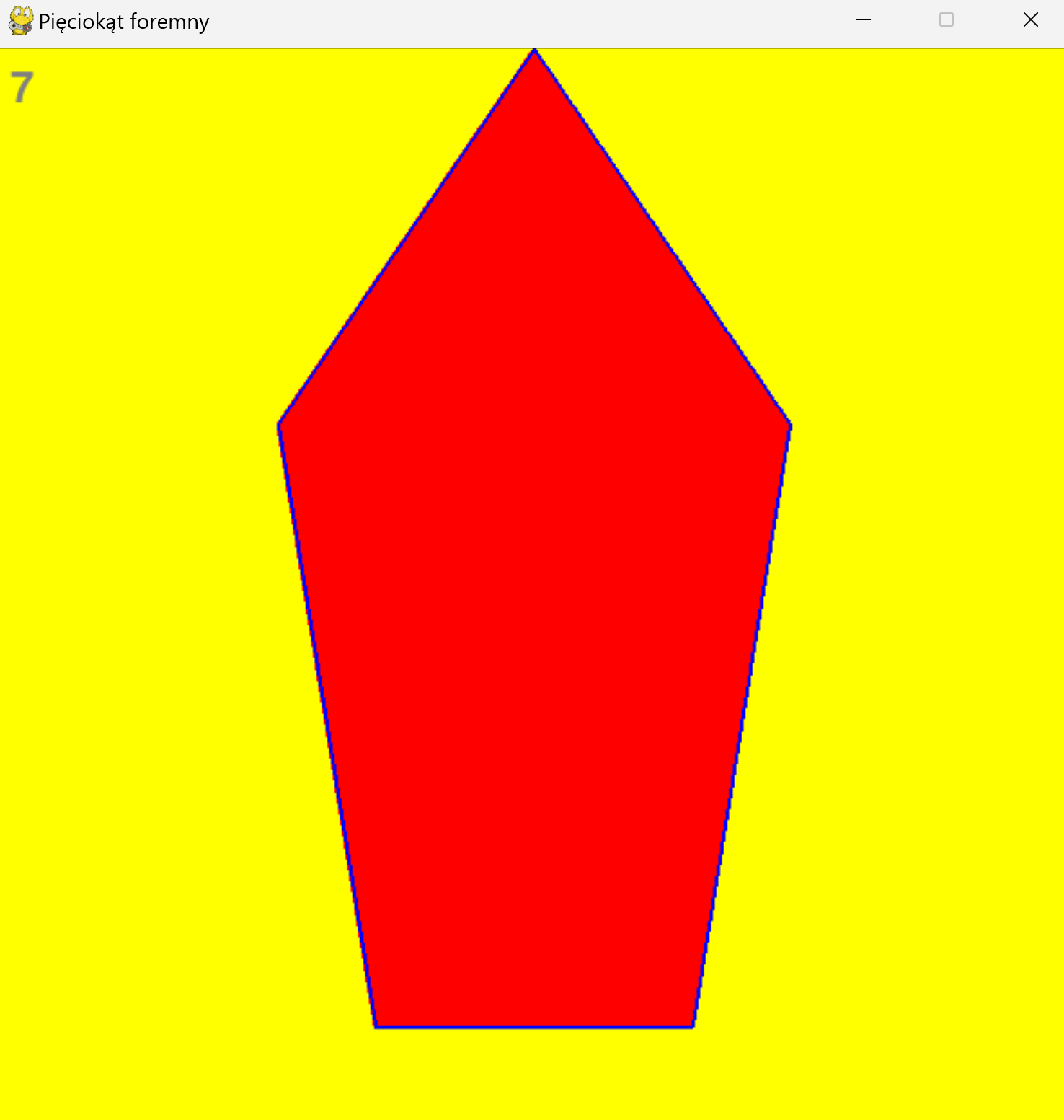
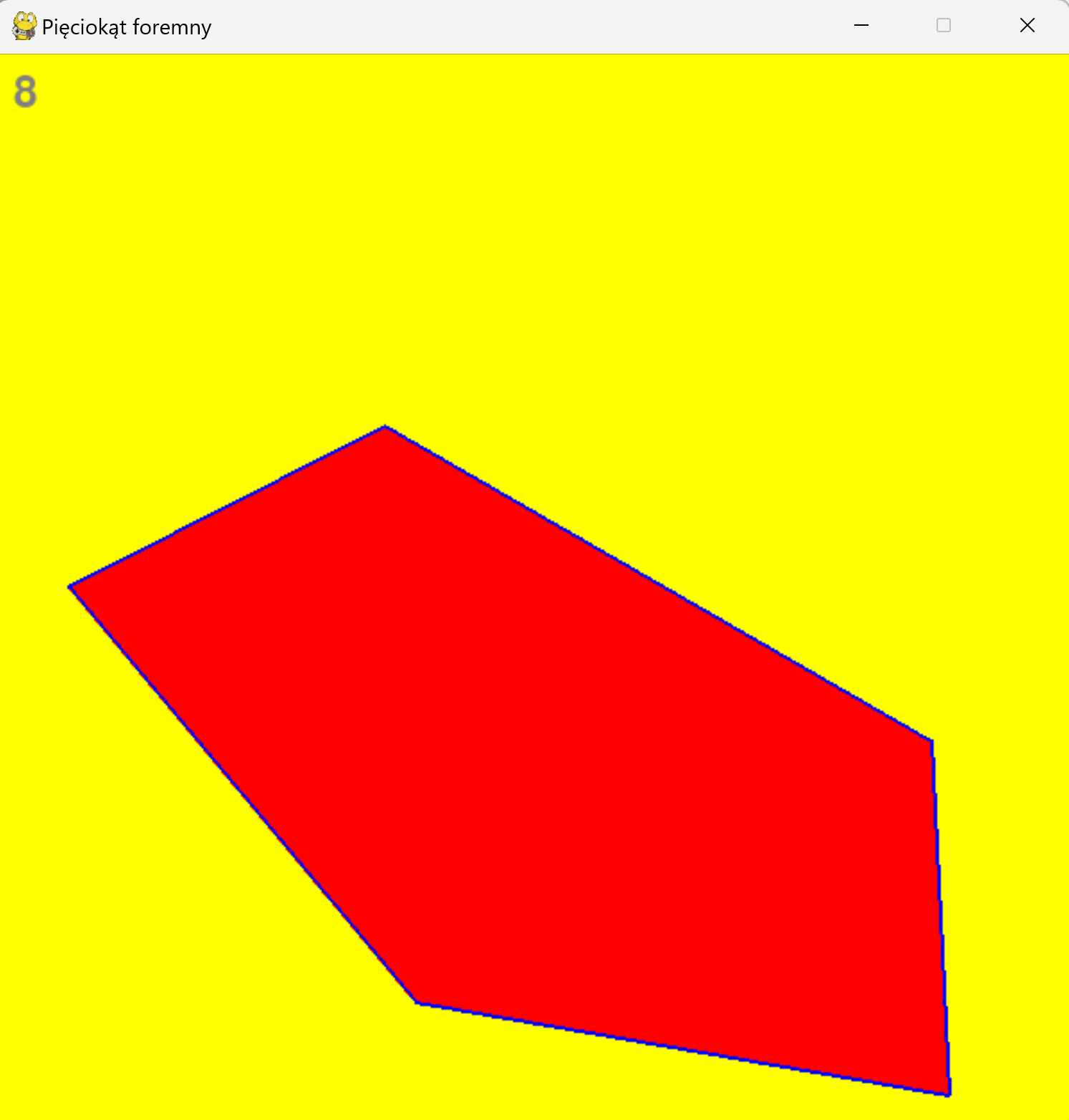
**4. Wynik działania:**

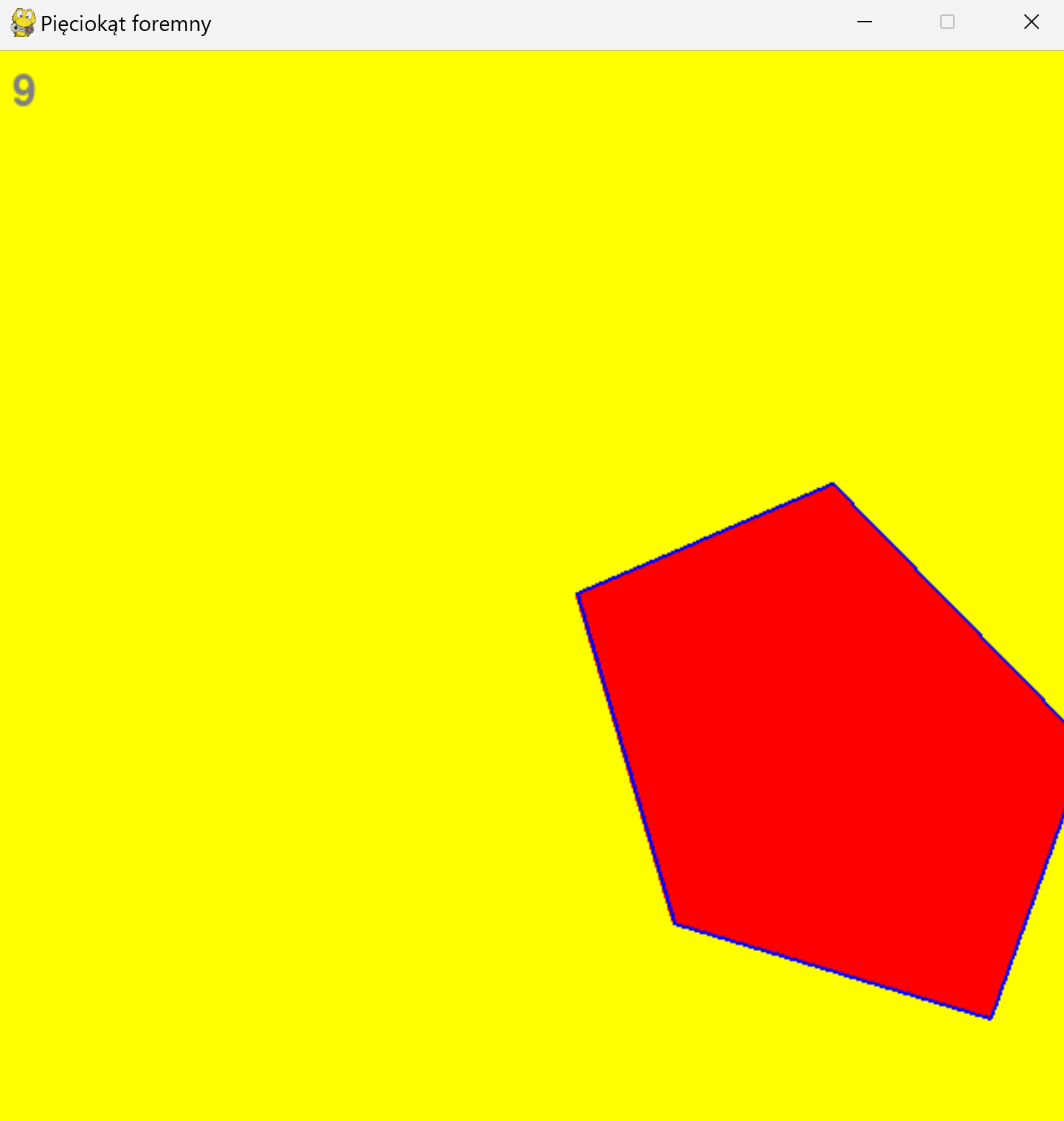
**Zadanie 1:**

**** ****

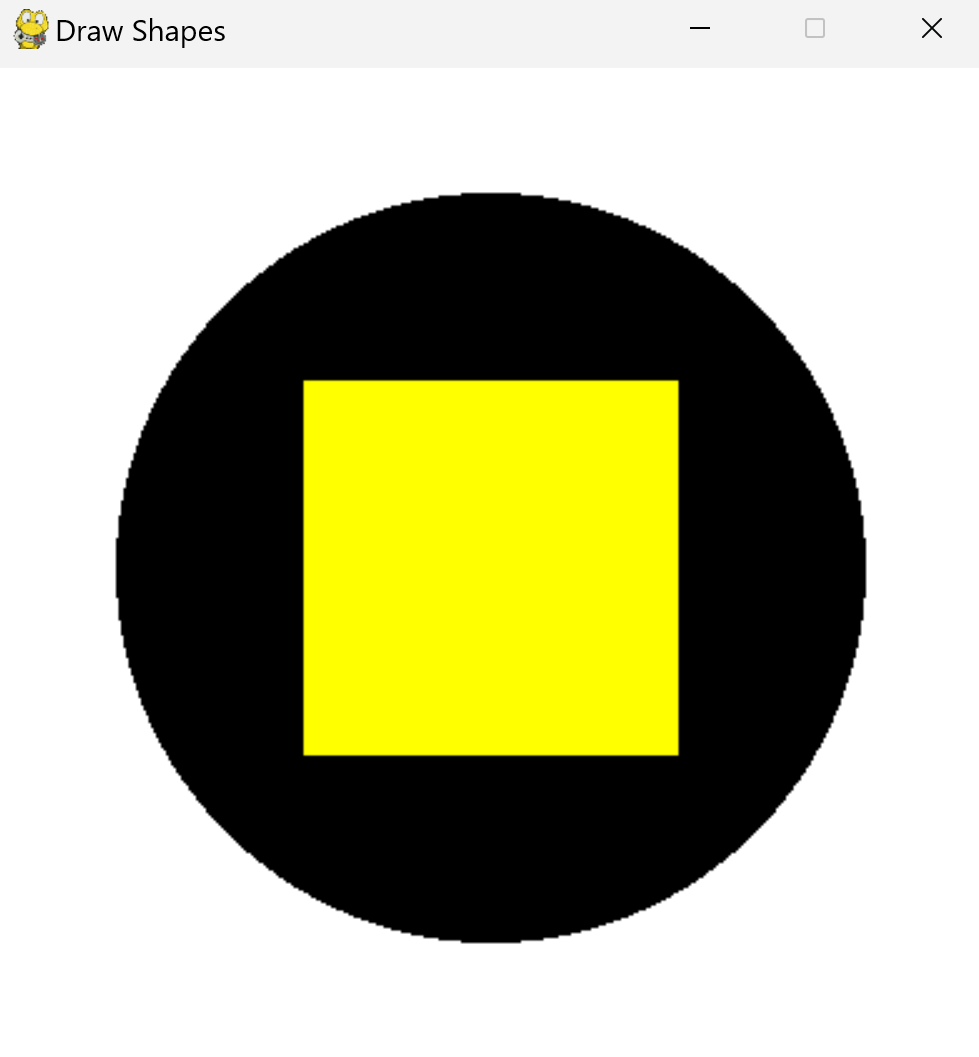
**** ****

**** ****

**** ****

****

**Zadanie 2:**

****

**5. Wnioski:**

Użycie biblioteki Pygame umożliwia tworzenie dynamicznych i interaktywnych aplikacji graficznych w Pythonie. Podstawą działania każdej aplikacji w Pygame jest odpowiednia obsługa pętli głównej, która utrzymuje działanie okna oraz obsługę zdarzeń. Tworzenie wielokątów i innych kształtów w Pygame jest proste i intuicyjne dzięki funkcjom takim jak pygame.draw.polygon. Kluczowym elementem zadania było wykorzystanie odpowiednich przekształceń po naciśnięciu klawiszy, co demonstruje możliwości interaktywne Pygame. Zadanie pokazuje, że Pygame może być używane do tworzenia nie tylko gier, ale również innych aplikacji graficznych wymagających interakcji użytkownika i dynamicznych zmian w grafice.

Pygame oferuje łatwe metody rysowania podstawowych kształtów, takich jak koło, kwadrat i trójkąt, co pozwala na szybkie prototypowanie graficzne. Podstawowe przekształcenia, takie jak przesunięcie, obrót czy skalowanie, są dostępne w Pygame i można je łatwo zastosować do rysowanych kształtów za pomocą pygame.transform. Zadanie wykazało, że nawet proste rysunki mogą wymagać przekształceń i odpowiedniego zarządzania położeniem oraz rozmiarem kształtów w oknie. Wykorzystanie Pygame do rysowania i przekształcania kształtów pozwala na lepsze zrozumienie podstaw grafiki komputerowej oraz programowania zdarzeń. Przekształcenia w Pygame są elastyczne i umożliwiają łatwe modyfikacje kształtów, co jest kluczowe dla tworzenia interaktywnych i dynamicznych aplikacji graficznych.