**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: prof. dr hab. Vasyl Martsenyuk,

**Laboratorium 2**

10.03.2024

**Temat:** Przeksztalcenia 2D w bibliotece pygame

**Wariant 2**

Mateusz Kukcuzka

Informatyka I stopień,

niestacjonarne,

4 semestr,

Gr.2A

1. **Polecenie:**

1. Pokazany jest obraz shuttle.jpg w panelu. Narysowac zamiast obrazu

wielokat wedlug wariantu (liczba n). Okno ma wymiary 600 na 600 pikseli,

a wielokat ma promien 150 pikseli. Kolejne zadanie polega na stosowaniu

odpowiednich przekszta lce´n do wielokata (lub bedziesz potrzebowal kombinacji przeksztalcen) po nacisnieciu na klawisze od 1 do 9 (patrz Fig. 1).

2. Narysowac figure okreslona wariantem (patrz Fig. 2). Dostepne sa trzy podstawowe ksztalty: kolo, kwadrat, trojkat. Podstawowe przeksztalcenia dostepne sa przez pygame.transform

1. **Wprowadzane dane:**
2. **Wykorzystane dane:**

<https://github.com/Kajdsky1337/GK_LAB3>

**1.**

import pygame

import math

pygame.init()

window\_width = 600

window\_height = 600

window = pygame.display.set\_mode((window\_width, window\_height))

pygame.display.set\_caption("Przekształcenie wielokąta")

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

def draw\_hexagon():

center = (window\_width // 2, window\_height // 2)

radius = 150

num\_sides = 17

angle\_increment = (2 \* math.pi) / num\_sides

points = []

for i in range(num\_sides):

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

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

points.append((x, y))

return points

def transform\_hexagon(points, option):

if option == 2:

points = rotate\_polygon(points, 45)

elif option == 3:

points = rotate\_polygon(points, 180)

elif option == 4:

points = skew\_polygon(points, 1.5)

elif option == 5:

points = align\_top(points)

elif option == 6:

points = skew\_polygon(points, 1.5)

points = rotate\_polygon(points, 180)

elif option == 7:

points = rotate\_polygon(points, 180)

points = mirror\_polygon(points, "vertical")

elif option == 8:

points = rotate\_polygon(points, 45)

points = align\_bottom(points)

points = widen\_polygon(points, 1.5)

elif option == 9:

points = rotate\_polygon(points, 180)

points = skew\_polygon(points, 1.5)

points = align\_right(points)

return points

def rotate\_polygon(points, angle):

center = (window\_width // 2, window\_height // 2)

angle\_radians = math.radians(angle)

rotated\_points = []

for point in points:

x = center[0] + math.cos(angle\_radians) \* (point[0] - center[0]) - math.sin(angle\_radians) \* (point[1] - center[1])

y = center[1] + math.sin(angle\_radians) \* (point[0] - center[0]) + math.cos(angle\_radians) \* (point[1] - center[1])

rotated\_points.append((x, y))

return rotated\_points

def skew\_polygon(points, factor):

center = (window\_width // 2, window\_height // 2)

skewed\_points = []

for point in points:

x = point[0] + factor \* (point[1] - center[1])

y = point[1]

skewed\_points.append((x, y))

return skewed\_points

def mirror\_polygon(points, axis):

if axis == "vertical":

center\_x = sum(point[0] for point in points) / len(points)

mirrored\_points = [(2 \* center\_x - point[0], point[1]) for point in points]

elif axis == "horizontal":

center\_y = sum(point[1] for point in points) / len(points)

mirrored\_points = [(point[0], 2 \* center\_y - point[1]) for point in points]

return mirrored\_points

def align\_top(points):

min\_y = min(point[1] for point in points)

shifted\_points = [(point[0], point[1] - min\_y) for point in points]

return shifted\_points

def align\_bottom(points):

max\_y = max(point[1] for point in points)

shifted\_points = [(point[0], point[1] - max\_y + window\_height) for point in points]

return shifted\_points

def align\_right(points):

max\_x = max(point[0] for point in points)

shifted\_points = [(point[0] - max\_x + window\_width, point[1]) for point in points]

return shifted\_points

def widen\_polygon(points, factor):

center\_y = sum(point[1] for point in points) / len(points)

widened\_points = []

for point in points:

x = point[0]

y = center\_y + factor \* (point[1] - center\_y)

widened\_points.append((x, y))

return widened\_points

running = True

option = 1

points = draw\_hexagon()

font = pygame.font.Font(None, 14)

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.KEYDOWN:

if event.key == pygame.K\_1:

option = 1

elif event.key == pygame.K\_2:

option = 2

elif event.key == pygame.K\_3:

option = 3

elif event.key == pygame.K\_4:

option = 4

elif event.key == pygame.K\_5:

option = 5

elif event.key == pygame.K\_6:

option = 6

elif event.key == pygame.K\_7:

option = 7

elif event.key == pygame.K\_8:

option = 8

elif event.key == pygame.K\_9:

option = 9

window.fill(WHITE)

transformed\_points = transform\_hexagon(points, option)

pygame.draw.polygon(window, BLACK, transformed\_points)

if option == 1:

mode\_text = "Wielokąt na środku okna"

elif option == 2:

mode\_text = "Wielokąt przekręcony o 45 stopni"

elif option == 3:

mode\_text = "Wielokąt odwrócony o 180 stopni"

elif option == 4:

mode\_text = "Wielokąt pochylony w lewo"

elif option == 5:

mode\_text = "Wielokąt przy górnej krawędzi okna"

elif option == 6:

mode\_text = "Wielokąt odwrócony o 180 stopni i przekrzywiony w lewo"

elif option == 7:

mode\_text = "Wielokąt odwrócony o 180 stopni i odwrócenie lustrzane"

elif option == 8:

mode\_text = "Wielokąt odwrócony o 45 stopni oraz przy dolnej krawędzi"

elif option == 9:

mode\_text = "Wielokąt odwrócony o 180 stopni pochylony w lewo oraz przy prawej krawędzi"

mode\_text = font.render("Tryb: " + str(mode\_text), True, BLACK)

window.blit(mode\_text, (10, 580))

pygame.display.update()

pygame.quit()

**2.**

import pygame

import sys

pygame.init()

screen\_width = 400

screen\_height = 400

screen = pygame.display.set\_mode((screen\_width, screen\_height))

pygame.display.set\_caption("Square with Triangles")

square\_vertices = [

(100, 100),

(300, 100),

(300, 300),

(100, 300)

]

green = (0, 255, 0)

black = (0, 0, 0)

def draw\_triangle(vertices):

pygame.draw.polygon(screen, green, vertices, 0)

def draw\_square\_with\_triangles(vertices):

draw\_triangle([vertices[0], vertices[1], vertices[2]])

draw\_triangle([vertices[0], vertices[3], vertices[1]])

running = True

while running:

screen.fill(black)

draw\_square\_with\_triangles(square\_vertices)

for event in pygame.event.get():

if event.type == pygame.QUIT:

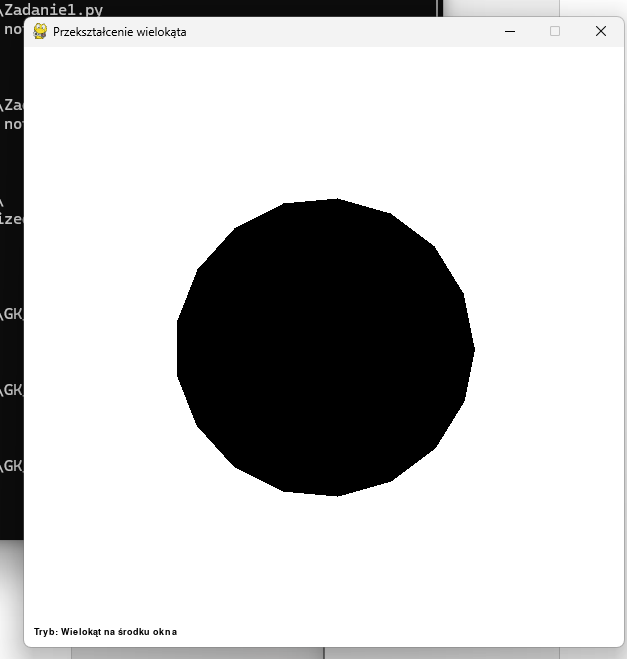
running = False

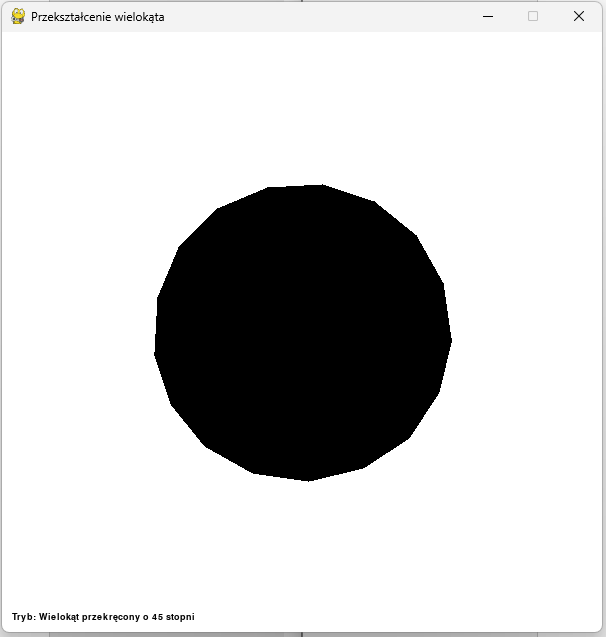
pygame.display.flip()

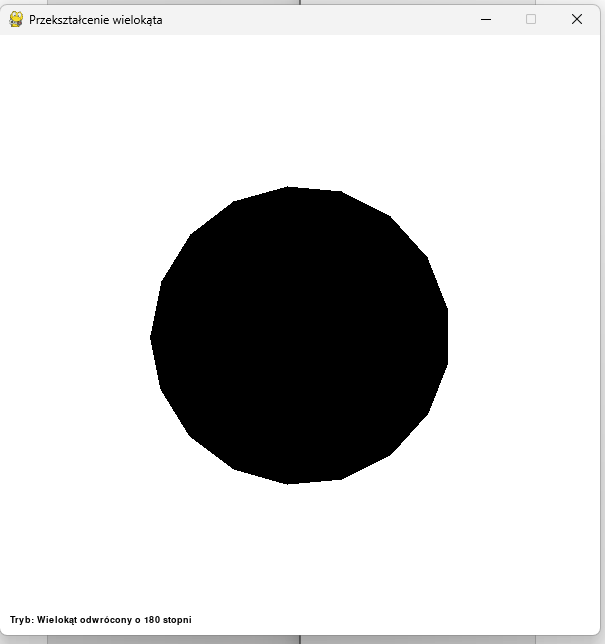
pygame.quit()

sys.exit()

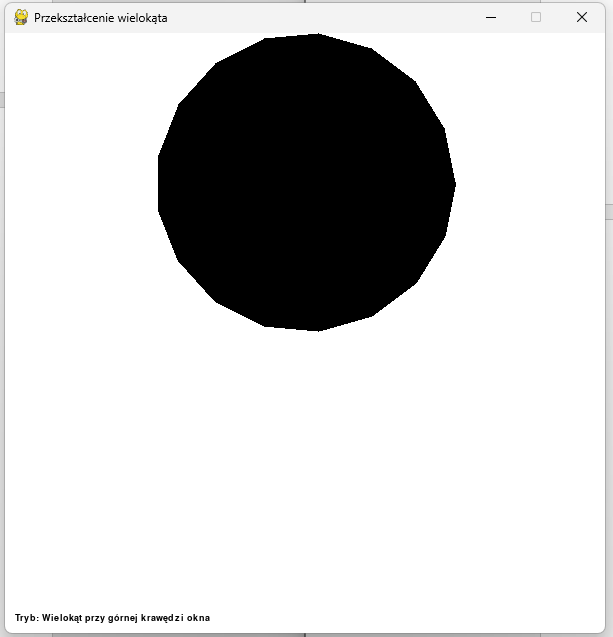
**4. Wynik działania:**

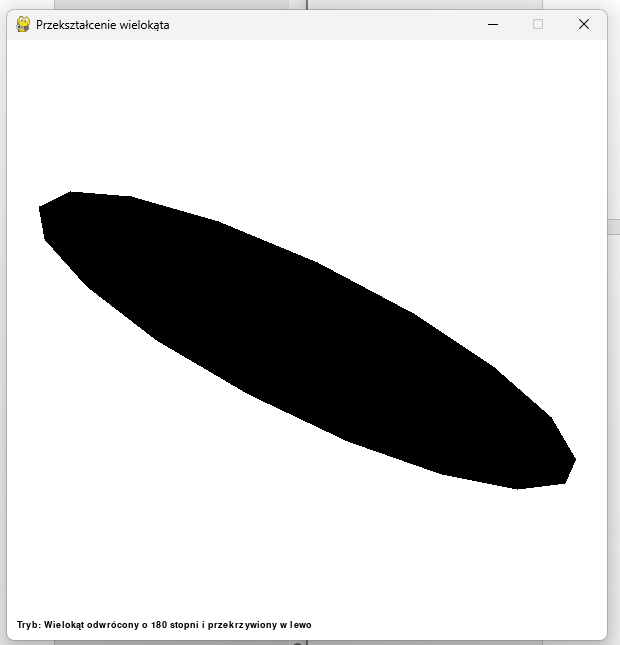




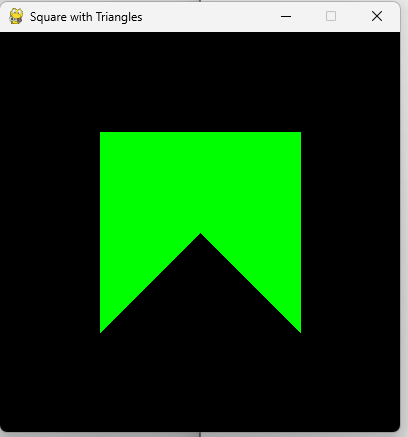








2)



1. **Wnioski:**

Przekształcenia 2D to kluczowy element grafiki komputerowej, który umożliwia manipulowanie obiektami w przestrzeni dwuwymiarowej. Użycie biblioteki Pygame ułatwia implementację różnych przekształceń dzięki bogatej funkcjonalności dostępnej dla deweloperów. Kluczowe jest zrozumienie matematyki stojącej za przekształceniami 2D, aby efektywnie wykorzystać te techniki w praktyce. Przesunięcie, skalowanie i obrót to fundamentalne rodzaje przekształceń, które można z łatwością zaimplementować w Pygame, co otwiera szerokie możliwości projektowe w tworzeniu gier i aplikacji. Efekty wizualne, uzyskane dzięki przekształceniom 2D, mogą znacznie zwiększyć atrakcyjność i funkcjonalność projektów graficznych, zachęcając użytkowników do interakcji. Mimo że Pygame oferuje bogate możliwości, skuteczne wykorzystanie przekształceń wymaga praktyki i doświadczenia w programowaniu, aby osiągnąć najlepsze wyniki. Wykorzystanie przekształceń 2D w Pygame daje programistom nieograniczone możliwości kreowania interaktywnych środowisk wizualnych, zarówno w celach edukacyjnych, jak i rozrywkowych.