Uniwersytet Bielsko-Bialski

Sprawozdanie

Zajęcia: Grafika Komputerowa
(Ćwiczenia laboratoryjne)
Prowadzący: prof dr. hab. Vasyl Martsenyuk

Laboratoria nr: 2

Temat ćwiczenia:

Przekształcenia 2D z użyciem PYGAME

Maksymilian Wójcik
Imformatyka I stopnia
niestacjonarne
4 semestr
gr. 1A

1.Polecenie:

1. Pokazany jest obraz \texttt{shuttle.jpg} w panelu. Narysować zamiast obrazu wielokąt według wariantu (liczba n). Okno ma wymiary 600 na 600 pikseli, a wielokąt ma promień 150 pikseli.

Kolejne zadanie polega na stosowaniu odpowiednich przekształceń do wielokąta (lub będziesz potrzebował kombinacji przekształceń) po naciśnięciu na klawisze od 1 do 9 (patrz Fig. \ref{fig:zad1}).\\

2. Narysować figurę określoną wariantem (patrz Fig. \ref{fig:zad2}). Dostępne są trzy podstawowe kształty: koło, kwadrat, trójkąt.

2. Wykorzystane komendy:

3. Wynik działania:

a)

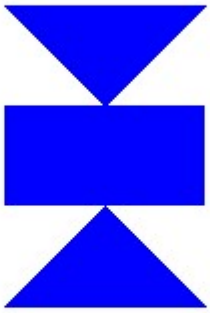
```python
import pygame
import sys

# Initialize Pygame
pygame.init()

# Set up the display
screen_size = (600, 600)
screen = pygame.display.set_mode(screen_size)
pygame.display.set_caption('Centered Blue Shapes')

# Colors
blue = (0, 0, 255)
background_color = (255, 255, 255)

# Fill the background
screen.fill(background_color)

# Define the shapes' positions for a centered layout
# Vertical offsets for spacing and centering
vertical_offset_triangle = 200  # Adjust as needed to fit the design
vertical_offset_rectangle = vertical_offset_triangle + 50  # Rectangle height + some space
vertical_offset_second_triangle = vertical_offset_rectangle + 50  # Additional space for the second trian

# First triangle (rotated 180 degrees) at the top
triangle1_points = [(250, vertical_offset_triangle), (350, vertical_offset_triangle), (300, vertical_offs

# Rectangle directly below the first triangle
rectangle_y = vertical_offset_rectangle
rectangle = (250, rectangle_y, 100, 50)

# Second triangle below the rectangle
triangle2_points = [(250, vertical_offset_second_triangle + 50), (350, vertical_offset_second_triangle +

# Drawing the shapes centered
pygame.draw.polygon(screen, blue, triangle1_points)
pygame.draw.rect(screen, blue, rectangle)
pygame.draw.polygon(screen, blue, triangle2_points)

# Update the display
pygame.display.flip()

# Main loop to keep the window open
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

b)

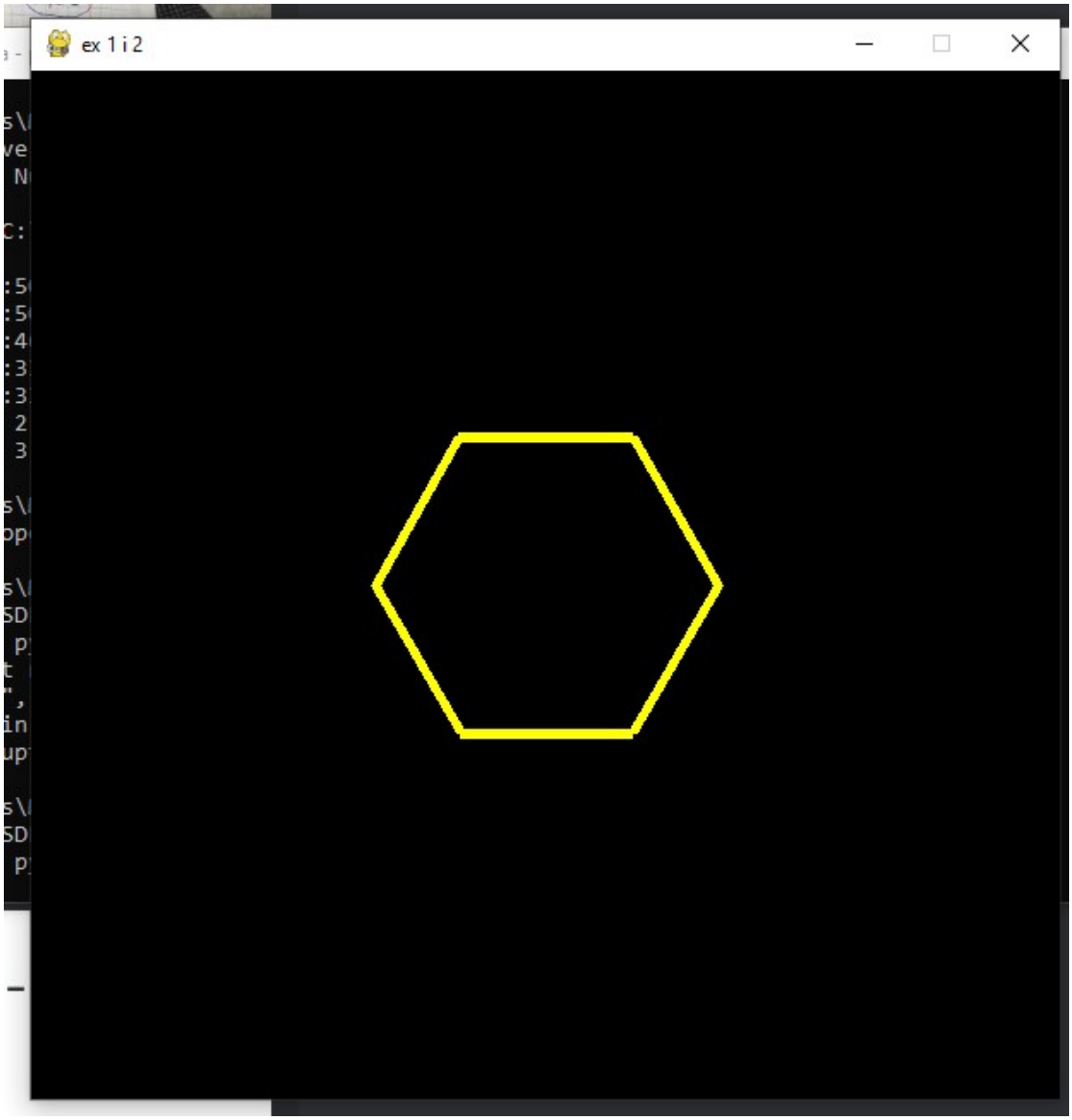
```python
def handle_key_event(event):
    win.fill(CZARNY)
    draw_angle(6)  # czternastokąt
    changed = None

    if event.key == pygame.K_1:
        changed = pygame.transform.scale(win, (int(width * 0.35), int(height * 0.35)))
    elif event.key == pygame.K_2:
        changed = pygame.transform.rotate(win, 45)
    elif event.key == pygame.K_3:
        changed = pygame.transform.flip(win, 0, 1)
    elif event.key == pygame.K_4:
        changed = pygame.transform.scale_by(win, (0.35, 1))
        changed = pygame.transform.rotozoom(changed, 45, 1)
    elif event.key == pygame.K_5:
        changed = pygame.transform.scale(win, (width, int(height * 0.35)))
    elif event.key == pygame.K_6:
        changed = pygame.transform.scale_by(win, (0.35, 1))
        changed = pygame.transform.rotozoom(changed, 180, 1)
    elif event.key == pygame.K_7:
        changed = pygame.transform.scale_by(win, (0.5, 1))
        changed = pygame.transform.flip(changed, 1, 0)
    elif event.key == pygame.K_8:
        changed = pygame.transform.scale_by(win, (1, 0.4))
        changed = pygame.transform.rotate(changed, -20)
    elif event.key == pygame.K_9:
        changed = pygame.transform.scale_by(win, (0.35, 1))
        changed = pygame.transform.rotozoom(changed, 90, 1)

    if changed:
        win.blit(changed, ((width - changed.get_width()) // 2, (height - changed.get_height()) // 2))
        pygame.display.flip()

def start():
    run = True
    while run:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                pygame.quit()
            elif event.type == pygame.KEYDOWN:
                handle_key_event(event)
```

4.Wynik działania:

a)



b)



5.Wnioski

- Użycie matplotlib pozwala na łatwe rysowanie i przekształcanie figur geometrycznych.

- Przekształcenia takie jak rotacja, skalowanie i przesunięcie mogą być stosowane do wielokąta za pomocą odpowiednich funkcji.

- Możemy definiować różne podstawowe kształty (koło, kwadrat, trójkąt) i rysować je w oknie o zadanych wymiarach.

Ten kod jest podstawą do dalszych modyfikacji i rozbudowy, na przykład dodania obsługi klawiatury w czasie rzeczywistym, co można osiągnąć przy użyciu bibliotek takich jak tkinter lub innych interfejsów GUI.

Kod źródłowy: https://github.com/mwojcik123/UBB-GK-MW