**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №1**

**з навчальної дисципліни «Computer Vision»**

**Тема:**

**ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ПОБУДОВИ ТА ПЕРЕТВОРЕННЯ КООРДИНАТ**

**ПЛОЩИННИХ (2D) ТА ПРОСТОРОВИХ (3D) ОБ’ЄКТІВ**

**Виконав:**

Студент 3 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Трикош І. В.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

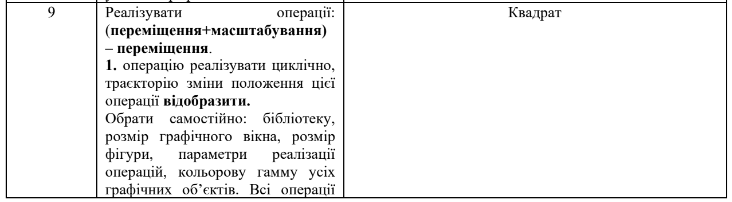
## **І. Мета:**

Виявити дослідити та узагальнити особливості формування та перетворення координат площинних (2d) та просторових (3d) об’єктів.

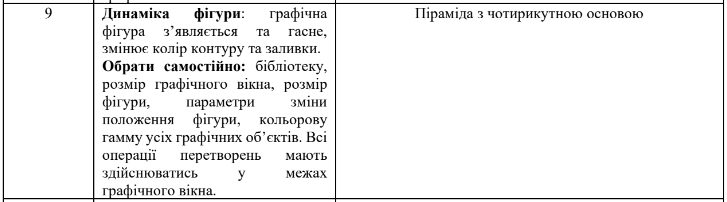
## **ІІ. Завдання:**

**Завдання ІІІ рівня**

Здійснити синтез математичних моделей та розробити програмний скрипт, що реалізує базові операції 2D перетворень над геометричними примітивами. Для розробки використовувати матричні операції та технології композиційних перетворень. Вхідна матриця координат кутів геометричної фігури має бути розширеною.



Здійснити синтез математичних моделей та розробити програмний скрипт, що реалізує базові операції 3D перетворень над геометричними примітивами: аксонометрична проекція будь-якого типу та з циклічне обертання (анімація) 3D графічного об’єкту навколо будь-якої обраної внутрішньої віссю. Траєкторію обертання не відображати. Для розробки використовувати матричні операції. Вхідна матриця координат кутів геометричної фігури має бути розширеною.



## **ІІІ. Результати виконання лабораторної роботи.**

* 1. **Синтезована математична модель**

Відповідно до умов задачі синтезовано математичні моделі операцій над структурами вхідних графічних об’єктів.

Модель є частково композиційною і реалізує послідовність дій переміщення – масштабування і просто переміщення для 2D об’єкту і обертання – переміщення – проєкція для 3D об’єкту.

Масштабування 2D об’єкту відбувається наступним чином:

де та – коефіцієнти масштабування.

Переміщення 2D об’єкту відбувається наступним чином:

де та – зміщення по x та y відповідно.

Обертання 3D об’єкту навколо осі Y відбувається наступним чином:

де – кут повороту (в радіанах).

Переміщення 3D об’єкту відбувається наступним чином:

де – координати точки переміщення.

Аксонометрична проєкція 3D об’єкту виконується наступним чином:

де – кут повороту (в радіанах) відносно осей.

Ортогональна проєкція 3D об’єкту на площину XY здійснюється наступним чином:

* 1. **Результати архітектурного проектування та їх опис**

На рис. 1 зображена блок-схема до моделі, що оперує 2D об’єктом:

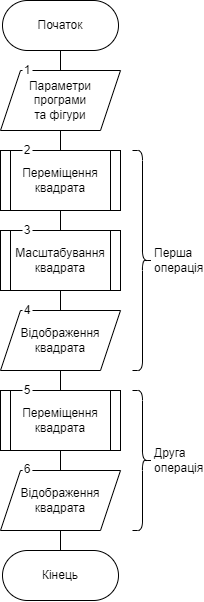


Рисунок 1 – Блок схема алгоритму для обробки 2D об’єктів

Робота алгоритму розпочинається з формування сегменту початкових параметрів. Сюди входить ширина та висота вікна, розмір сторони квадрата, крок переміщення та коефіцієнт масштабування, формування двох точок квадрата. Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.1.

У блоці 2 реалізується переміщення квадрата, а саме двох його точок, з певним кроком, який заданий у блоці 1.

У блоці 3 відбувається масштабування квадрата, тобто його точок, з коефіцієнтом масштабування, що заданий у блоці 1.

У блоці 4 реалізується відображення квадрату у графічному вікні за допомогою бібліотеки graphics. На цьому виконання першої операції (переміщення + масштабування) завершено.

У блоці 5 реалізується переміщення квадрата, який заданий у блоці 1 на крок, що теж заданий у блоці 1.

У блоці 6 відбувається відображення квадрату у графічному вікні за допомогою бібліотеки graphics. На цьому виконання другої операції (переміщення) завершено.

На рис. 2 зображена блок-схема до моделі, що оперує 3D об’єктом:

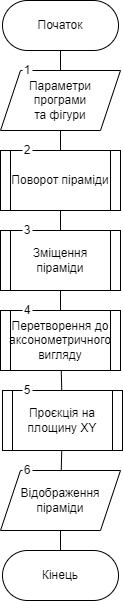


Рисунок 2 – Блок схема алгоритму для обробки 3D об’єктів

Робота алгоритму розпочинається з формування сегменту початкових параметрів. Сюди входить ширина та висота вікна, розмір сторони основи піраміди, висота піраміди, точка переміщення, кут повороту фігури, кут проєкції та формування точок піраміди (вершина та 4 точки основи). Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.2.

У блоці 2 реалізується поворот піраміди навколо осі Y на заданий у блоці 1 кут.

У блоці 3 виконується зміщення піраміди на координати, що задані в блоці 1.

У блоці 4 відбувається аксонометричне перетворення, а саме поворот по осі Y та X на кут, що заданий у блоці 1.

У блоці 5 відбувається проєктування піраміди на площину XY.

У блоці 6 відбувається відображення піраміди у графічному вікні за допомогою бібліотеки graphics. На цьому виконання алгоритму завершено.

* 1. **Опис структури проекту програми**

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проєкт.

Проєкт базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

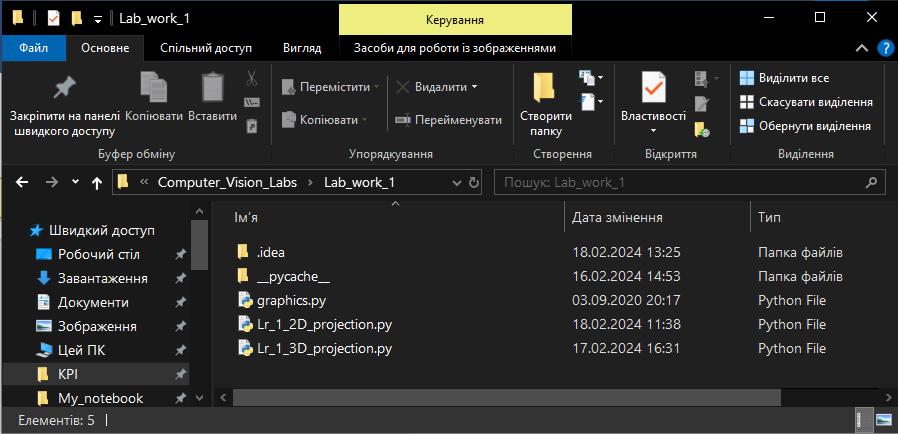


Рисунок 3 – Структура проєкту

Lab\_work\_1 – головний каталог проєкту

graphics.py – файл графічної бібліотеки graphics

Lr\_1\_2D\_projection.py – файл програмного коду до першого завдання

Lr\_1\_3D\_projection.py – файл програмного коду до другого завдання

* 1. **Результати роботи програми відповідно до завдання (допускається у формі скриншотів)**

Результатом роботи програми є сукупність послідовності графічних вікон, що реалізують умови завдання лабораторної роботи.

1. Переміщення та масштабування квадрата:

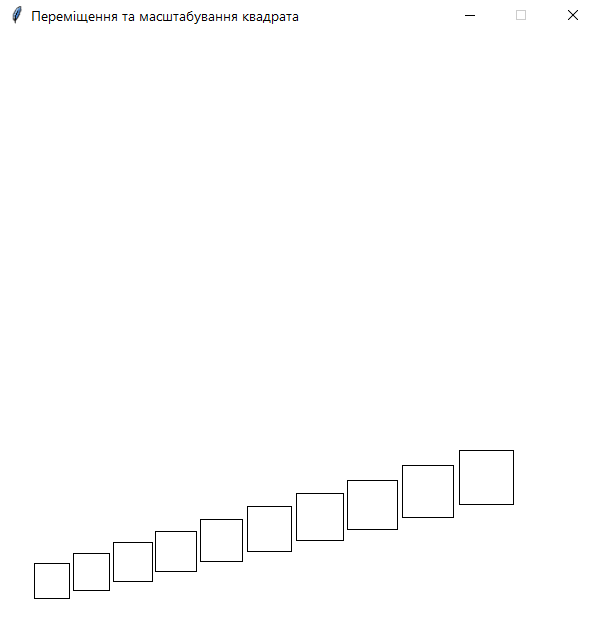


Рисунок 4 – Переміщення та масштабування квадрата

1. Переміщення квадрата:



Рисунок 5 – Переміщення квадрата

1. Циклічне обертання за годинниковою стрілкою навколо осі Y та аксонометрична проєкція на площину XY, зміна кольору та контуру піраміди:

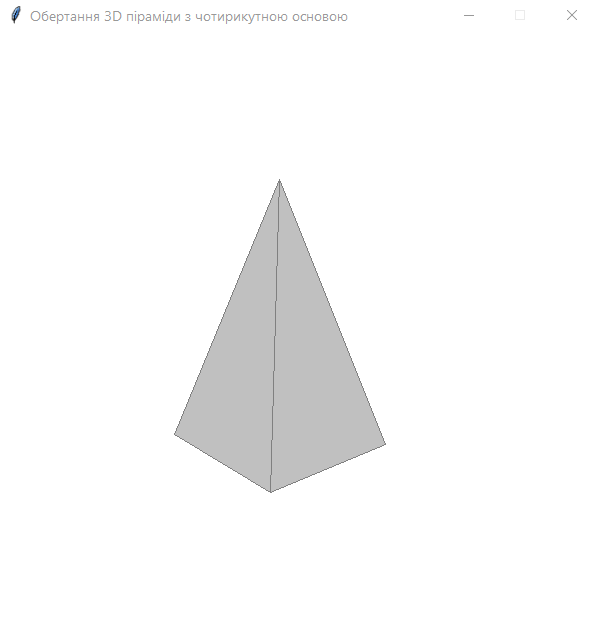


Рисунок 6 – Обертання та аксонометрична проєкція піраміди

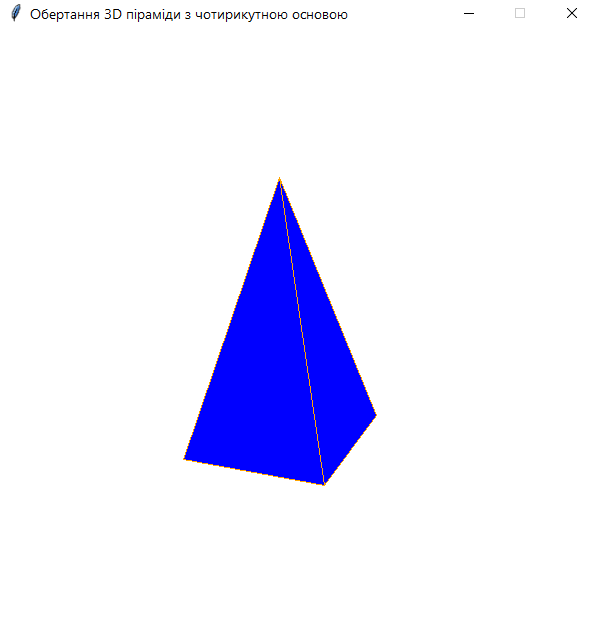


Рисунок 7 – Зміна кольору та контуру піраміди

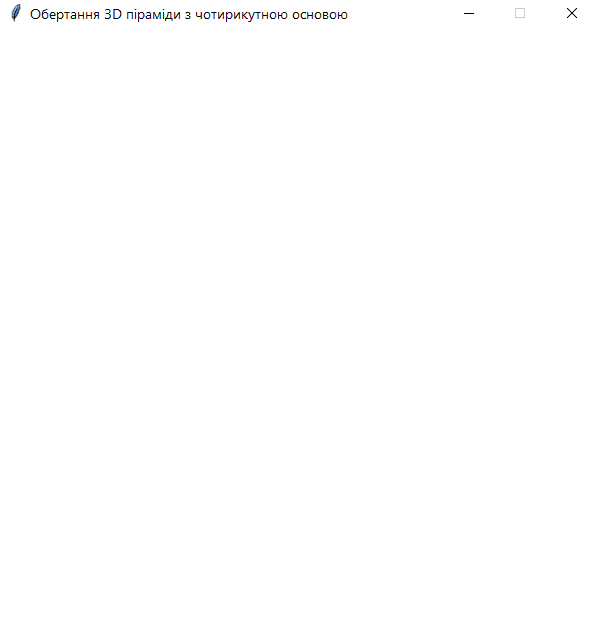


Рисунок 8 – Зникнення піраміди

Таким чином, представлені результати у повному обсязі відповідають завданню лабораторної роботи.

* 1. **Програмний код, що забезпечує отримання результату (допускається у формі скриншотів)**

Програмний код послідовно реалізує алгоритми рис.1-2 та спрямовано на отримання результатів, поданих на рис.4-8.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

Обчислення проводились з розширеною матрицею координат квадрата та піраміди з чотирикутною основою.

При цьому використано можливості Python бібліотек: graphics; time; numpy; math; random.

Контексні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

Lr\_1\_2D\_projection.py

*"""  
--------------------- 2D - геометричні перетворення ---------------------------  
Завдання:  
Програма повинна будувати 2D графічний об’єкт (квадрат) та реалізовувати його перетворення у матричній формі:  
1. Переміщення + масштабування в режимі анімації;  
2. Переміщення.  
  
"""  
  
from* graphics *import* \*  
*import* time  
*import* numpy *as* np  
  
*# Параметри вікна*WINDOW\_WIDTH = 600  
WINDOW\_HEIGHT = 600  
  
*# Розмір сторони квадрата*SQUARE\_SIZE = 35  
  
*# Крок переміщення*dx = 35  
dy = 35  
  
*# Коефіцієнти масштабування*s\_x = 1.05  
s\_y = 1.05  
  
  
*def* get\_default\_square\_coordinates():  
 *""" Отримати координати стартового квадрату """* x\_1 = SQUARE\_SIZE  
 y\_1 = WINDOW\_HEIGHT - 2 \* SQUARE\_SIZE  
 x\_2 = 2 \* SQUARE\_SIZE  
 y\_2 = WINDOW\_HEIGHT - SQUARE\_SIZE  
 *return* (x\_1, y\_1), (x\_2, y\_2)  
  
  
*def* move\_point(x\_1, y\_1):  
 *""" Перемістити точку """* p = np.array([[x\_1, y\_1, 1]])  
 t = np.array([[1, 0, dx], [0, 1, -dy], [0, 0, 1]])  
 tt = t.T  
 total = p.dot(tt)  
 x11 = total[0, 0]  
 y11 = total[0, 1]  
 *return* x11, y11  
  
  
*def* scale\_point(x\_1, y\_1):  
 *""" Масштабувати точку """* p = np.array([[x\_1, y\_1, 1]])  
 s = np.array([[s\_x, 0, 0], [0, s\_y, 0], [0, 0, 1]])  
 total = p.dot(s)  
 x11 = total[0, 0]  
 y11 = total[0, 1]  
 *return* x11, y11  
  
  
*def* stop\_animation():  
 *"""Зупинити анімацію"""* time.sleep(0.2)  
  
  
*# ---------------- I. ПЕРЕМІЩЕННЯ ТА МАСШТАБУВАННЯ КВАДРАТА З АНІМАЦІЄЮ ------------------------*win = GraphWin("Переміщення та масштабування квадрата", WINDOW\_WIDTH, WINDOW\_HEIGHT)  
win.setBackground('white')  
  
(x1, y1), (x2, y2) = get\_default\_square\_coordinates()  
  
steps = 10  
*for* i *in* range(steps):  
 stop\_animation()  
  
 *# Намалювати квадрат* obj = Rectangle(Point(x1, y1), Point(x2, y2))  
 obj.draw(win)  
  
 *# Перемістити точки квадрата* x1, y1 = move\_point(x1, y1)  
 x2, y2 = move\_point(x2, y2)  
  
 *# Масштабувати точки квадрата* x1, y1 = scale\_point(x1, y1)  
 x2, y2 = scale\_point(x2, y2)  
  
win.getMouse()  
win.close()  
  
  
*# ---------------- II. ПЕРЕМІЩЕННЯ КВАДРАТА ------------------------*win = GraphWin("Переміщення квадрата", WINDOW\_WIDTH, WINDOW\_HEIGHT)  
win.setBackground('white')  
  
(x1, y1), (x2, y2) = get\_default\_square\_coordinates()  
  
*# Намалювати квадрат*obj = Rectangle(Point(x1, y1), Point(x2, y2))  
obj.draw(win)  
  
stop\_animation()  
  
*# Перемістити точки квадрата*x1, y1 = move\_point(x1, y1)  
x2, y2 = move\_point(x2, y2)  
  
*# Намалювати квадрат*obj = Rectangle(Point(x1, y1), Point(x2, y2))  
obj.draw(win)  
  
win.getMouse()  
win.close()

Lr\_1\_3D\_projection.py

*"""  
--------------------- 3D - геометричні перетворення ---------------------------  
Завдання:  
Синтез 3D об'єкту (піраміда з чотирикутною основою) та його геометричне перетворення:  
1. Аксонометрична проєкція будь-якого типу;  
2. Циклічне обертання 3D об'єкту навколо будь-якої обраної внутрішньої осі.  
  
"""  
  
import* random  
  
*from* graphics *import* \*  
*import* numpy *as* np  
*import* math  
  
*# Параметри вікна*WINDOW\_WIDTH = 600  
WINDOW\_HEIGHT = 600  
  
*# Параметри піраміди*PYRAMID\_WIDTH = 150  
PYRAMID\_HEIGHT = 300  
  
*# Координати піраміди. Перша - вершина*pyramid = np.array([[0, 0, 0, 1],  
 [-PYRAMID\_WIDTH//2, PYRAMID\_HEIGHT, -PYRAMID\_WIDTH//2, 1],  
 [+PYRAMID\_WIDTH//2, PYRAMID\_HEIGHT, -PYRAMID\_WIDTH//2, 1],  
 [-PYRAMID\_WIDTH//2, PYRAMID\_HEIGHT, +PYRAMID\_WIDTH//2, 1],  
 [+PYRAMID\_WIDTH//2, PYRAMID\_HEIGHT, +PYRAMID\_WIDTH//2, 1]])  
  
  
*def* project\_on\_x\_y(figure):  
 *""" Спроєктувати на площину XY """* t = np.array([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 0, 0], [0, 0, 0, 1]])  
 projection\_x\_y = figure.dot(t)  
 *return* projection\_x\_y  
  
  
*def* shift\_figure(figure, l, m, n):  
 *""" Змістити фігуру """* t = np.array([[1, 0, 0, l], [0, 1, 0, m], [0, 0, 1, n], [0, 0, 0, 1]])  
 tt = t.T  
 shifted\_figure = figure.dot(tt)  
 *return* shifted\_figure  
  
  
*def* degrees\_to\_radians(angle):  
 *""" Конвертувати градуси в радіани """  
 return* (math.pi \* angle) / 180  
  
  
*def* to\_axonometric\_view(figure, theta\_d\_1, theta\_d\_2):  
 *""" Зробити аксонометричну проєкцію """* theta\_r\_1 = degrees\_to\_radians(theta\_d\_1)  
 theta\_r\_2 = degrees\_to\_radians(theta\_d\_2)  
  
 t1 = np.array([[math.cos(theta\_r\_1), 0, -math.sin(theta\_r\_1), 0],  
 [0, 1, 0, 0],  
 [math.sin(theta\_r\_1), 0, math.cos(theta\_r\_1), 0],  
 [0, 0, 0, 1]])  
 view\_y = figure.dot(t1)  
  
 t2 = np.array([[1, 0, 0, 0],  
 [0, math.cos(theta\_r\_2), math.sin(theta\_r\_2), 0],  
 [0, -math.sin(theta\_r\_2), math.cos(theta\_r\_2), 0],  
 [0, 0, 0, 1]])  
 view\_y\_x = view\_y.dot(t2)  
 *return* view\_y\_x  
  
  
*def* rotate\_y(figure, theta\_d):  
 *""" Виконати поворот по осі Y """* theta\_r = degrees\_to\_radians(theta\_d)  
 t = np.array([[math.cos(theta\_r), 0, -math.sin(theta\_r), 0],  
 [0, 1, 0, 0],  
 [math.sin(theta\_r), 0, math.cos(theta\_r), 0],  
 [0, 0, 0, 1]])  
 f\_y = figure.dot(t)  
 *return* f\_y  
  
  
*class* Figure:  
 *def* \_\_init\_\_(*self*, window):  
 *self*.window = window  
  
 *self*.side1 = Polygon()  
 *self*.side2 = Polygon()  
 *self*.side3 = Polygon()  
 *self*.side4 = Polygon()  
 *self*.side5 = Polygon()  
  
 *self*.counter = 0  
  
 *def* \_\_clear\_figure(*self*):  
 *""" Очистити попередню фігуру """  
 self*.side1.undraw()  
 *self*.side2.undraw()  
 *self*.side3.undraw()  
 *self*.side4.undraw()  
 *self*.side5.undraw()  
  
 *def* \_\_set\_color(*self*):  
 *"""  
 Встановити колір фігури на кожні 10 кроків.  
 З ймовірністю 0.2 фігура буде білого кольору, інакше буде випадковий колір заливки та каркасу.  
 """* colors = ["black", "blue", "red", "gray", "yellow", "silver", "navy", "orange", "cyan"]  
  
 *if self*.counter % 10 == 0:  
 *self*.outline\_color = "white"  
 *self*.fill\_color = "white"  
 *if* random.random() < 0.8:  
 *self*.outline\_color = random.choice(colors)  
 *self*.fill\_color = random.choice(colors)  
  
 *self*.side1.setOutline(*self*.outline\_color)  
 *self*.side1.setFill(*self*.fill\_color)  
 *self*.side2.setOutline(*self*.outline\_color)  
 *self*.side2.setFill(*self*.fill\_color)  
 *self*.side3.setOutline(*self*.outline\_color)  
 *self*.side3.setFill(*self*.fill\_color)  
 *self*.side4.setOutline(*self*.outline\_color)  
 *self*.side4.setFill(*self*.fill\_color)  
 *self*.side5.setOutline(*self*.outline\_color)  
 *self*.side5.setFill(*self*.fill\_color)  
  
 *self*.counter += 1  
  
 *def* \_\_draw\_figure(*self*):  
 *""" Намалювати фігуру """  
 self*.side1.draw(*self*.window)  
 *self*.side2.draw(*self*.window)  
 *self*.side3.draw(*self*.window)  
 *self*.side4.draw(*self*.window)  
 *self*.side5.draw(*self*.window)  
  
 *def* visualize(*self*, figure):  
 *""" Візуалізувати фігуру """* e\_x, e\_y = figure[0, 0], figure[0, 1]  
 a\_x, a\_y = figure[1, 0], figure[1, 1]  
 b\_x, b\_y = figure[2, 0], figure[2, 1]  
 c\_x, c\_y = figure[3, 0], figure[3, 1]  
 d\_x, d\_y = figure[4, 0], figure[4, 1]  
  
 *self*.\_\_clear\_figure()  
  
 *self*.side1 = Polygon(Point(a\_x, a\_y), Point(b\_x, b\_y), Point(d\_x, d\_y), Point(c\_x, c\_y))  
 *self*.side2 = Polygon(Point(a\_x, a\_y), Point(b\_x, b\_y), Point(e\_x, e\_y))  
 *self*.side3 = Polygon(Point(a\_x, a\_y), Point(c\_x, c\_y), Point(e\_x, e\_y))  
 *self*.side4 = Polygon(Point(b\_x, b\_y), Point(d\_x, d\_y), Point(e\_x, e\_y))  
 *self*.side5 = Polygon(Point(c\_x, c\_y), Point(d\_x, d\_y), Point(e\_x, e\_y))  
  
 *self*.\_\_set\_color()  
 *self*.\_\_draw\_figure()  
  
  
win = GraphWin("Обертання 3D піраміди з чотирикутною основою", WINDOW\_WIDTH, WINDOW\_HEIGHT)  
win.setBackground('white')  
  
pyramid\_figure = Figure(win)  
  
*# Кут повороту фігури*rotate\_angle = 10  
*# Точка зсуву фігури*shift\_point = [325, 75, 0]  
*# Кут проєкції*theta = 30  
  
*while True*:  
 pyramid = rotate\_y(pyramid, rotate\_angle)  
 shifted\_pyramid = shift\_figure(pyramid, \*shift\_point)  
 axonometric\_pyramid = to\_axonometric\_view(shifted\_pyramid, theta, theta)  
 pyramid\_projection = project\_on\_x\_y(axonometric\_pyramid)  
 pyramid\_figure.visualize(pyramid\_projection)  
 time.sleep(0.1)

## **IV. Висновки.**

У ході виконання лабораторної роботи було проведено дослідження особливостей побудови 2D та 3D об’єктів з використанням можливостей мови програмування Python та наступних операцій над ними: переміщення, проєкція, масштабування, обертання. Дослідження показали, що моделювання графічних об’єктів відбувається у їх реальному вигляді без змін у геометричній структурі. Відображення 3D об’єктів потребує проєктування на площину за допомогою проєкцій різних типів. Це все втілюється за допомогою перетворень графічних об’єктів з використанням матричних форм і розширених матриць.

Виконав: студент Трикош І. В.