**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) ОБ’ЄКТІВ**

**Виконав:**

Студент 3 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Головня О.Р.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

**І. Мета:**

Виявити дослідити та узагальнити особливості побудови та перетворення координат площинних (2D) об’єктів.

**ІІ. Завдання:**

|  |  |  |
| --- | --- | --- |
| **Варіант** (День народження | **Технічні умови** | **Графічна фігура** |
| 3 | Реалізувати операції:  переміщення – масштабування –  обертання.  3. операцію реалізувати циклічно,  траєкторію зміни положення цієї  операції відобразити.  Обрати самостійно: бібліотеку,  розмір графічного вікна, розмір  фігури, параметри реалізації  операцій, кольорову гамму усіх  графічних об’єктів. Всі операції  перетворень мають здійснюватись  у межах графічного вікна. | Прямокутник |

**Завдання І рівня складності – максимально 7 балів.**

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. Функціонал скрипта, що розробляється має реалізувати технічних вимог табл.1

Додатку 1.

**ІІІ. Результати виконання лабораторної роботи.**

**3.1. Синтезована математична модель перетворень графічних об’єктів відповідно до індивідуального завдання.**

Програма відображає прямокутник, який зазнає послідовної та композиційної трансформації. При цьому прямокутник обертається на кут TETA, масштабується з коефіцієнтом 1.05 та переміщується на вектор (5, 10). Трансформація змінюється кожні 20 кроків.

Давайте розберемо кожну частину детальніше:

1. **TETA** - це кут обертання в радіанах.
2. **R** - це матриця обертання для обертання об'єкта навколо осі Z на кут TETA. Вона має вигляд:
3. **T** - це матриця масштабування, яка масштабує об'єкт за віссю X та Y на 1.05 разів. Форма матриці:
4. **m** - це вектор зміщення за осями X та Y. У цьому випадку, об'єкт зміщується на 5 одиниць по осі X та на 10 одиниць по осі Y.
5. **M** - це матриця зміщення, яка зміщує об'єкт за віссю X на m[0] одиниць і за віссю Y на m[1] одиниць. Форма матриці:
6. **T\_speed** - це коефіцієнт швидкості зміни масштабу для матриці T. У моєму випадку, швидкість зміни масштабу дорівнює 1.05 разів.

**3.2. Блок схема алгоритму та її опис.**

Було прийнято рішення розробити програму з циклічним виконанням відповідних операцій.

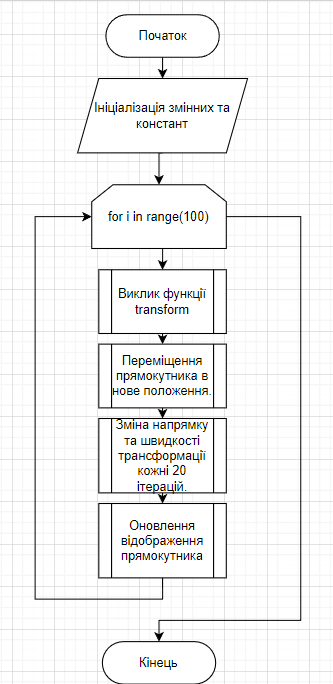


Рис.1. Блок-схема програми.

Робота програми відбувається у вікні, де програма відображає основні завдання роботи:

**Обертання (Rotation):**

* Визначення центру прямокутника як середнього значення координат його вершин.
* Створення матриці обертання **R**, яка використовується для обертання на кут **TETA** в радіанах.
* Для кожної вершини прямокутника:
  + Відняти координати центру від координат вершини.
  + Помножити координати на матрицю **R**.
  + Додати координати центру до отриманих координат.

**Масштабування (Scaling):**

* Визначення центру прямокутника як середнього значення координат його вершин.
* Створення матриці масштабування **T**, яка збільшує розмір прямокутника на **1.05** або зменшує на **1/1.05** кожну ітерацію.
* Для кожної вершини прямокутника:
  + Відняти координати центру від координат вершини.
  + Помножити координати на матрицю **T**.
  + Додати координати центру до отриманих координат.

**Обертання (Rotation):**

* Визначення центру прямокутника як середнього значення координат його вершин.
* Створення матриці обертання **R**, яка використовується для обертання на кут **TETA** в радіанах.
* Для кожної вершини прямокутника:
  + Відняти координати центру від координат вершини.
  + Помножити координати на матрицю **R**.
  + Додати координати центру до отриманих координат.

**3.3. Опис структури проекту програми в середовищі PyCharm.**

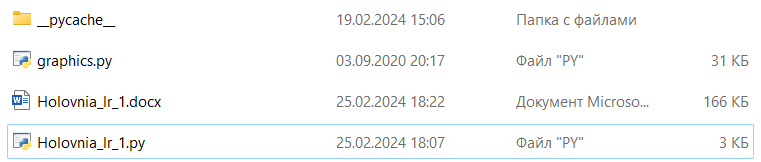
****

Рис.2. Відображення структури.

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей idle python 3.11.

Проект та має таку структуру:

Holovnia\_lr\_1.docx – звіт лабораторної роботи

graphics.py – Simple object oriented graphics library

Holovnia\_lr\_1.py – основна логіка програми

**3.4. Результати роботи програми відповідно до завдання.**

Програма видає вікно у якому відображаються усі потрібні операції. Нижче наведено порядок виконання просторових перетворень згідно з поставленим завданням.

|  |
| --- |
|  |
| Рис.3. Відображення виконання програми. |

Представлені результати у повному обсязі відповідають завданню лабораторної роботи.

**3.5. Програмний код.**

import time

import numpy as np

from graphics import GraphWin, Polygon, Point

WINDOW\_WEIGHT = 600

WINDOW\_HEIGHT = 600

CENTER = np.array((WINDOW\_WEIGHT // 2, WINDOW\_HEIGHT // 2))

ONES = np.array([

[1],

[1],

[1],

])

TETA = 3

R = np.array([

[np.cos(TETA), np.sin(TETA), 0],

[-np.sin(TETA), np.cos(TETA), 0],

[0, 0, 1],

])

T = np.array([

[1.05, 0, 0],

[0, 1.05, 0],

[0, 0, 1],

])

m = np.array([5, 10])

M = np.array([

[1, 0, 0],

[0, 1, 0],

[m[0], m[1], 1],

])

T\_speed = 1.05

win = GraphWin("Rectangle", WINDOW\_WEIGHT, WINDOW\_HEIGHT)

win.setBackground('light blue')

def remove\_rectangle(rectangle):

rectangle.setFill("light blue")

def redraw\_rectangle(points, rectangle):

if rectangle:

remove\_rectangle(rectangle)

rectangle = Polygon(\*[Point(int(x), int(y)) for (x, y) in points])

rectangle.setFill('orange')

rectangle.draw(win)

return rectangle

def rectangle\_center(points):

return np.array(((points[0][0] + points[1][0] + points[2][0] + points[3][0]) / 4,

(points[0][1] + points[1][1] + points[2][1] + points[3][1]) / 4))

def transform(points, A):

center = rectangle\_center(points)

P = np.concatenate(

(points - center, np.ones((len(points), 1))),

axis=1

)

P\_stroke = P @ A

return P\_stroke[:, :2] + center

def compositional\_transformations(rectangle\_points):

rectangle = None

for i in range(100):

rectangle\_points = transform(rectangle\_points, (R @ T @ M))

rectangle = redraw\_rectangle(rectangle\_points, rectangle)

# -----------Change direction-----------

if (i // 20) % 2 == 0:

T[0, 0], T[1, 1] = T\_speed, T\_speed

M[2, 0], M[2, 1] = m[0], m[1]

else:

T[0, 0], T[1, 1] = 1 / T\_speed, 1 / T\_speed

M[2, 0], M[2, 1] = -m[0], -m[1]

time.sleep(0.3)

remove\_rectangle(rectangle)

return rectangle

def main():

rectangle\_points = np.array([

[100, 100],

[100, 200],

[300, 200],

[300, 100],

])

rectangle = compositional\_transformations(rectangle\_points)

win.getMouse()

win.close()

if \_\_name\_\_ == '\_\_main\_\_':

main()

**3.6. Аналіз результатів відлагодження та верифікації результатів роботи програми.**

Результати від лагодження та тестування довели працездатність розробленого коду.

Верифікація функціоналу програмного коду, порівняння отриманих результатів з технічними умовами завдання на лабораторну роботу доводять, що усі завдання виконані у повному обсязі.

**IV. Висновки.**

Під час виконання лабораторної роботи проведено дослідження особливостей створення плоских (2D) об'єктів, використовуючи Python. Дослідження підтвердило, що моделювання 2D об'єктів та опрацювання плоских форм відбуваються в реальних розмірах без змін у геометричній структурі.