****

Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №1

**З дисципліни «Технології Computer Vision»**

*Дослідження технологій побудови та перетворення координат площинних (2D) та просторових (3D) обʼєктів*

|  |  |  |
| --- | --- | --- |
| Виконала  студент кафедри ІСТ ФІОТ, групи ІА-12: |  | Перевірив:  пос. Баран Д. Р. |
| Яковенко Д. О. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Київ 2024

**І. Мета:**

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

**ІІ. Завдання:**

*Лабораторія провідної ІТ-компанії реалізує масштабний проект розробки універсальної платформи з цифрової обробки зображень для задач Computer Vision. Платформа передбачає розташування back-end компоненти на власному хмарному сервері з наданням повноважень користувачам заздалегідь адаптованого front-end функціоналу універсальної платформи. Цим формується унікальна для потреб замовника ERP система з технологіями Computer Vision*

*Замовниками ресурсів платформи є: державні та комерційні компанії, що розробляють медичне обладнання з діагностування захворювань за візуальною інформацією; автоматизації аграрного бізнесу в аспекті обліку посівних територій за даними з БПЛА; візуального контролю безпекових заходів на об’єктах критичної інфраструктури: аеропорти, торгівельно-розважальні центри, житлові комплекси тощо.*

*Вам, як Computer Vision поставлено завдання.*

**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.

**Завдання ІІ рівня – максимально 8 балів.**

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.

**Завдання ІІІ рівня – максимально 9 балів.**

Реалізувати розробку програмного скрипта із функціоналом І та ІІ рівнів складності.

|  |  |  |
| --- | --- | --- |
| **Варіант** (порядковий номер в списку групи) | **Технічні умови** | **Графічна фігура** |
| 10 | Реалізувати операції:  **масштабування** (**переміщення+обертання)** | Прямокутник |
| 10 | Динаміка фігури: графічна фігура зʼявляється та гасне, зʼявляється в різних частинах графічного вікна | Паралелепіпед |

**ІІІ. Результати виконання лабораторної роботи.**

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

Відповідно до умов задачі синтезовано математичну модель операцій над структурою вхідного графічного об’єкту.

Модель є частково композиційною і реалізує послідовність дій масштабування – (переміщення + обертання).

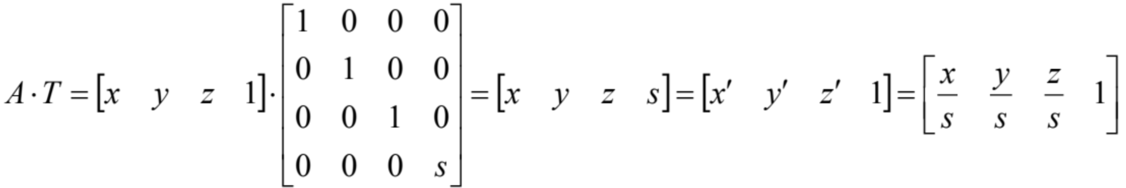
**Переміщення** в точку  реалізується за узагальненою моделлю:

|  |  |
| --- | --- |
| . | (1) |

**Обертання** реалізує суть аксонометричної проекції та здійснюється у відповідності до моделі:

|  |  |
| --- | --- |
| . | (2) |

**Масштабування** реалізується за допомогою діагональних елементів матриці T - diarT[4x4].



(3)

|  |
| --- |
|  |

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

Застосування синтезованих моделей здійснюється у порядку, що відображає суть алгоритму реалізації завдань лабораторної роботи.

|  |
| --- |
|  |
| Рис.1. Блок-схема алгоритмів програми для завдання першої та другої складності відповідно. |

Розглянемо перший алгоритм. Робота розпочинається із формування сегменту початкових даних, що описують прямокутник. Він формується з 8 точок кожна з яких має три координати. Описані дії реалізуються у блоці 1 блок-схеми алгоритму рис.1.

Відповідно до математичної моделі (3) в блоці 2 реалізується масштабування прямокутника. При цьому існує дві матриці масштабування (на непарних ітераціях – обʼєкт збільшується, на парних – зменшується).

В блоці 3 реалізується поворот графічної фігури в формі виразу (2) та одночасно виконується її переміщення (відповідно до математичної моделі (1)).

Результат вищеописаних обрахунків відображається у графічному вікні.

На цьому робота алгоритму завершується.

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

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

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

|  |
| --- |
|  |
| Рис.2. Структура проекту. |

CV\_lab1 – головний каталог проекту

Lab1.py – файл програмного коду лабораторної роботи;

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

Результатом роботи програми є програма, в якій за допомогою кнопок управління користувач має змогу перемістити(вверх, вниз, ліворуч, праворуч), обертати(w, a, s, d, q, e) та масштабувати(z, x) обʼєкт. Кнопками «1» та «2» запускаються відповідні рівні складності лабораторної роботи.

1. Спочатку користувач вводить характеристики паралелепіпеда (його висоту, ширину, глибину):

|  |
| --- |
|  |
| Рис.3. Введення значень для формування паралелепіпеда.    Рис.4. Відображення фігури в центрі екрану. |

2. Переміщення, обертання, та масштабування паралелепіпеда здійснюється за допомогою кнопок з клавіатури. Обертання можна здійснювати відносно осі x, y, z.

|  |
| --- |
|  |
| Рис.5. Зміна положення паралелепіпеду, обертання та масштабування. |

3. Натиснемо клавішу «1» та спостерігатимемо «масштабування – (переміщення + обертання)».

|  |
| --- |
|  |
| Рис.6. Завдання із складністю I. |

4. Натиснемо клавішу «2», щоб спостерігати як фігура обертається на рандомну кількість градусів довкола певної осі і переміщується у рандомну точку. При цьому вона повинна зʼявлятися та зникати.

|  |
| --- |
|  |
| Рис.7. Завдання із складністю II. |

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

**3.5. Програмний код.**

Програмний код послідовно реалізує алгоритми рис.1 для виконання завдання першого та другого рівня складності.

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

Обчислення проводились з розширеною матрицею координат паралелепіпеду.

При цьому використано можливості Python бібліотек: pygame; numpy.

Контексні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

import pygame  
import numpy as np  
  
# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Налаштування\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
# Розміри вікна  
window\_width = 900  
window\_height = 600  
  
# Розміри паралелепіпеда  
print("Фігура: паралелепіпед")  
height = int(input("Введіть висоту: "))  
width = int(input("Введіть ширину: "))  
depth = int(input("Введіть глибину: "))  
  
# Задання початкових координат паралелепіпеда  
initial\_coordinates = np.array([[0, 0, 0, 1],  
 [height, 0, 0, 1],  
 [height, width, 0, 1],  
 [0, width, 0, 1],  
 [0, 0, depth, 1],  
 [height, 0, depth, 1],  
 [height, width, depth, 1],  
 [0, width, depth, 1]])  
  
pygame.init()  
  
screen = pygame.display.set\_mode((window\_width, window\_height))  
pygame.display.set\_caption("Лабораторна робота №1")  
  
# Щоб паралелепіпед при запуску програми був розташований в центрі вікна  
shift\_vector = np.array([window\_width / 2 - height / 2, window\_height / 2 - width / 2, 0, 0])  
initial\_coordinates += shift\_vector.astype(int)  
parallelepiped\_coordinates = initial\_coordinates.copy()  
  
  
# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Функція для матричних перетворень в центрі початку координат, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_а потім зміщення центру обʼєкта у точку, де він знаходився перед цим\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
def transform\_and\_center(coordinates, transformation\_matrix):  
 center = np.mean(coordinates, axis=0)  
 translated\_coordinates = coordinates - center  
 transformed\_coordinates = np.dot(translated\_coordinates, transformation\_matrix)  
 final\_coordinates = transformed\_coordinates + center  
  
 return final\_coordinates  
  
  
# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Малювання та оновлення вікна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
def draw\_and\_update():  
 screen.fill((255, 255, 255))  
 pygame.draw.polygon(screen, (0, 0, 0), parallelepiped\_coordinates[:4, :2], 2)  
 pygame.draw.polygon(screen, (0, 0, 0), parallelepiped\_coordinates[4:, :2], 2)  
 for i in range(4):  
 pygame.draw.line(screen, (0, 0, 0), parallelepiped\_coordinates[i, :2],  
 parallelepiped\_coordinates[i + 4, :2], 2)  
 pygame.display.flip()  
  
  
# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Функції для обертання навколо осей X, Y, Z\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
def rotate\_x(angle):  
 rotation\_matrix = np.array([[1, 0, 0, 0],  
 [0, np.cos(angle), np.sin(angle), 0],  
 [0, -np.sin(angle), np.cos(angle), 0],  
 [0, 0, 0, 1]])  
 return rotation\_matrix  
  
  
def rotate\_y(angle):  
 rotation\_matrix = np.array([[np.cos(angle), 0, -np.sin(angle), 0],  
 [0, 1, 0, 0],  
 [np.sin(angle), 0, np.cos(angle), 0],  
 [0, 0, 0, 1]])  
 return rotation\_matrix  
  
  
def rotate\_z(angle):  
 rotation\_matrix = np.array([[np.cos(angle), -np.sin(angle), 0, 0],  
 [np.sin(angle), np.cos(angle), 0, 0],  
 [0, 0, 1, 0],  
 [0, 0, 0, 1]])  
 return rotation\_matrix  
  
  
# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Функція для масштабування\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
def scale(sx, sy, sz):  
 scaling\_matrix = np.array([[sx, 0, 0, 0],  
 [0, sy, 0, 0],  
 [0, 0, sz, 0],  
 [0, 0, 0, 1]])  
 return scaling\_matrix  
  
  
# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Початок програми\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
running = True  
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\_UP:  
 parallelepiped\_coordinates[:, 1] -= 10 # Зміщення вгору  
 elif event.key == pygame.K\_DOWN:  
 parallelepiped\_coordinates[:, 1] += 10 # Зміщення вниз  
 elif event.key == pygame.K\_LEFT:  
 parallelepiped\_coordinates[:, 0] -= 10 # Зміщення вліво  
 elif event.key == pygame.K\_RIGHT:  
 parallelepiped\_coordinates[:, 0] += 10 # Зміщення вправо  
 elif event.key == pygame.K\_s:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates, rotate\_x(np.radians(30)))  
 elif event.key == pygame.K\_w:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates, rotate\_x(-np.radians(30)))  
 elif event.key == pygame.K\_a:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates, rotate\_y(np.radians(30)))  
 elif event.key == pygame.K\_d:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates, rotate\_y(-np.radians(30)))  
 elif event.key == pygame.K\_q:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates, rotate\_z(np.radians(30)))  
 elif event.key == pygame.K\_e:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates, rotate\_z(-np.radians(30)))  
 elif event.key == pygame.K\_x:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates,  
 scale(1.1, 1.1, 1.1)) # Збільшення  
 elif event.key == pygame.K\_z:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates,  
 scale(0.9, 0.9, 0.9)) # Зменшення  
 elif event.key == pygame.K\_1:  
 parallelepiped\_coordinates = initial\_coordinates.copy()  
 for frame in range(16):  
 for i in range(10):  
 # Перевірка парності циклу: якщо парне – то зменшуємо фігуру, якщо ні – збільшуємо  
 if frame % 2 == 0:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates,  
 scale(0.95, 0.95, 0.95))  
 else:  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates,  
 scale(1.05, 1.05, 1.05))  
 draw\_and\_update()  
 pygame.time.delay(50)  
  
 for i in range(5):  
 # Перевірка циклу, фігура переміщується вверх-вправо-вниз-вліво та одночасно обертається  
 if frame % 4 == 0:  
 parallelepiped\_coordinates[:, 1] -= 10  
 elif (frame - 1) % 4 == 0:  
 parallelepiped\_coordinates[:, 0] += 10  
 elif (frame - 2) % 4 == 0:  
 parallelepiped\_coordinates[:, 1] += 10  
 else:  
 parallelepiped\_coordinates[:, 0] -= 10  
  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates,  
 rotate\_z(-np.radians(15)))  
 draw\_and\_update()  
 pygame.time.delay(50)  
 parallelepiped\_coordinates = initial\_coordinates.copy()  
 break  
 elif event.key == pygame.K\_2:  
 for frame in range(16):  
 # Рандомне переміщення фігури  
 random\_x = int(np.random.uniform(-300, 300))  
 random\_y = int(np.random.uniform(-300, 300))  
 new\_coordinates = parallelepiped\_coordinates.copy()  
 new\_coordinates[:, 1] += random\_x  
 new\_coordinates[:, 0] += random\_y  
  
 # Перевірка, чи не виходить фігура на межі екрану  
 if (0 <= new\_coordinates[:, 0].min() < window\_width) and (  
 0 <= new\_coordinates[:, 1].min() < window\_height):  
 parallelepiped\_coordinates = new\_coordinates  
  
 # Рандомне обертання  
 random\_rot\_x = np.radians(np.random.uniform(0, 360))  
 random\_rot\_y = np.radians(np.random.uniform(0, 360))  
 random\_rot\_z = np.radians(np.random.uniform(0, 360))  
 rotation\_matrix = rotate\_x(random\_rot\_x) @ rotate\_y(random\_rot\_y) @ rotate\_z(random\_rot\_z)  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates, rotation\_matrix)  
  
 # Рандомне масштабування фігури  
 scaling = np.random.uniform(0.5, 1.5)  
 parallelepiped\_coordinates = transform\_and\_center(parallelepiped\_coordinates,  
 scale(scaling, scaling, scaling))  
  
 draw\_and\_update()  
 pygame.time.delay(2000)  
  
 # Очищення екрану  
 screen.fill((255, 255, 255))  
 pygame.display.flip()  
 pygame.time.delay(1000)  
  
 draw\_and\_update()  
  
pygame.quit()

#------------------------------------------------------------------------------

**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.**

У ході виконання лабораторної роботи проведено дослідження особливостей побудови площинних (2d) та просторових - 3d об’єктів з використанням можливостей алгоритмічної мови високого рівня Python. Дослідження довели, що моделювання 2d і 3d об’єктів та обробка просторових форм відбуваються в реальному їх вигляді без змін у геометричній структурі.

Виконала: студент Яковенко Д.О.