**Міністерство освіти і науки України**

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

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №3**

**з навчальної дисципліни «Computer Vision»**

**Тема:**

**ДОСЛІДЖЕННЯ АЛГОРИТМІВ ФОРМУВАННЯ ТА ОБРОБКИ РАСТРОВИХ ЦИФРОВИХ ЗОБРАЖЕНЬ**

**Виконав:**

Студент 3 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Тихонов Ф.С.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**І. Мета роботи:**

Виявити дослідити та узагальнити особливості реалізації алгоритмів формування та обробки векторних цифрових зображень на прикладі застосування алгоритмів інтерполяції, апроксимації та згладжування складних 3D растрових об’єктів та застосування технологій видалення невидимих граней та ребер.

**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 систем з технологіями Computer Vision. Змовниками 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 – максимально 7 балів.**

Здійснити виконання завдання лабораторної роботи із застосуванням алгоритму

інтерполяції для побудови векторного зображення 2D, 3D графічного об’єкту.

|  |  |  |
| --- | --- | --- |
| 28 | Відображення 3D фігури  реалізується з використанням аксонометричної проекції будь-якого типу. Обрати самостійно: бібліотеку, розмір графічного вікна, розмір фігури, динаміку зміни положення фігури, кольорову гамму графічного об’єкту. Всі операції перетворень мають здійснюватися у межах графічного вікна. | Піраміда з чотирикутною основою.  Метод інтерполяції: поліном Лагранжа.  Метод видалення невидимих ліній та поверхонь:  алгоритм плаваючого обрію. |

**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 реалізує операцію побудови векторного зображення 3-вимірної піраміди з чотирикутною основою як ортогональну проекцію на площину XY за допомогою полінома Лагранжа як алгоритму інтерполяції.

**Інтерполяція Лагранжа** – це поліноміальна модель інтерполяції. Для точок даних інтерполяція Лагранджа задана формулою:

Де – це базовий поліном Лагранджа:

В даному випадку, для побудови векторного зображення використовується інтерполяція Лагранжа для побудови ребер чотирикутника у реальному часі, тобто у режимі своєрідної анімації. Анімація йде послідовно по ребер піраміди, тобто в даному випадку інтерполяція Лагранжа використовується як поліном 1 степені, тобто переходить у лінійну інтерполяцію.

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

Блок-схема розв’язку матиме вигляд на рис. 1.

A diagram of a company

Description automatically generated

Програма запускається при виконанні файла у середовищі PyCharm, і виконується автоматично і одноразово. Виконується анімація побудови векторного зображення ортогональної проекції чотирикутної піраміди на площину XY.

**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 функційного програмування програмування та має таку структуру.

|  |
| --- |
|  |

Рис.2. Структура проекту.

lab-3.py – файл з програмою

lab-work\_3.pdf – файл з завданням

Звіт.docx – файл зі звітом

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

Нижче продемонстровано послідовність виконання програми на рис. 3 - 10:

A black rectangular object with a blue line

Description automatically generated

Рис. 3. Побудова нижнього ребра чотирикутної основи

A black square with blue line

Description automatically generated

Рис. 4. Побудова правої побічного ребра чотирикутної основи

A black rectangle with blue lines

Description automatically generated

Рис. 5. Побудова верхнього ребра чотирикутної основи

A blue rectangle on a black background

Description automatically generated

Рис. 6. Побудована чотирикутної основи

A screenshot of a computer

Description automatically generated

Рис. 7. Побудовано першу верхнє ребро

A screenshot of a computer

Description automatically generated

Рис. 8. Побудовано друге верхнє ребро

A screenshot of a computer

Description automatically generated

Рис. 9. Побудовано третє верхнє ребро

A screenshot of a drawing

Description automatically generated

Рис. 10. Побудовано останнє верхнє ребро

**3.5. Програмний код.**

Програмний код послідовно реалізує алгоритм рис.1 та спрямовано на отримання результатів, поданих на рис.2-10.

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: numpy, tkinter.

import tkinter as tk  
import numpy as np  
  
  
class LinearSmoothAnimatedPyramid:  
 def \_\_init\_\_(self, root, canvas\_width=400, canvas\_height=300):  
 self.root = root  
 self.canvas = tk.Canvas(root, width=canvas\_width, height=canvas\_height)  
 self.canvas.pack()  
  
 self.base\_length = 100  
 self.base\_width = 80  
 self.height = 150  
 self.center\_x = canvas\_width // 2  
 self.center\_y = canvas\_height // 2  
  
 self.coordinates = [  
 (self.center\_x - self.base\_length // 2, self.center\_y + self.base\_width // 2), # v0  
 (self.center\_x + self.base\_length // 2, self.center\_y + self.base\_width // 2), # v1  
 (self.center\_x + self.base\_length // 2, self.center\_y - self.base\_width // 2), # v2  
 (self.center\_x - self.base\_length // 2, self.center\_y - self.base\_width // 2), # v3  
 (self.center\_x, self.center\_y - self.height) # v4 (Top of the pyramid)  
 ]  
  
 # Lines of the pyramid  
 self.lines = [  
 (self.coordinates[0], self.coordinates[1]),  
 (self.coordinates[1], self.coordinates[2]),  
 (self.coordinates[2], self.coordinates[3]),  
 (self.coordinates[3], self.coordinates[0]),  
 (self.coordinates[0], self.coordinates[4]),  
 (self.coordinates[1], self.coordinates[4]),  
 (self.coordinates[2], self.coordinates[4]),  
 (self.coordinates[3], self.coordinates[4])  
 ]  
  
 self.line\_draw\_duration\_ms = 1000 # Time to draw each line  
 self.total\_time\_ms = 0 # Total time passed  
  
 self.current\_line\_index = 0  
 self.draw\_next\_line()  
  
 def linear\_interpolation(self, start, end, num\_points):  
 x\_points = np.linspace(start[0], end[0], num\_points)  
 y\_points = np.linspace(start[1], end[1], num\_points)  
 return list(zip(x\_points, y\_points))  
  
 def draw\_next\_line(self):  
 *"""  
 Draw the next line of the pyramid using linear interpolation for smoothness.  
 """* if self.current\_line\_index < len(self.lines):  
 line = self.lines[self.current\_line\_index]  
 start, end = line  
  
 num\_points = self.line\_draw\_duration\_ms // 50  
 interpolated\_points = self.linear\_interpolation(start, end, num\_points)  
  
 for i in range(1, len(interpolated\_points)):  
 segment\_start = interpolated\_points[i - 1]  
 segment\_end = interpolated\_points[i]  
 self.canvas.after(i \* 50, self.draw\_line\_segment, segment\_start, segment\_end)  
  
 self.total\_time\_ms += self.line\_draw\_duration\_ms  
 self.current\_line\_index += 1  
 self.canvas.after(self.total\_time\_ms, self.draw\_next\_line)  
  
 def draw\_line\_segment(self, start, end):  
 self.canvas.create\_line(start[0], start[1], end[0], end[1], fill='blue')  
  
def main():  
 root = tk.Tk()  
 root.title("Linear Smooth Animated Pyramid Drawing")  
 app = LinearSmoothAnimatedPyramid(root)  
 root.mainloop()  
  
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.**

У ході виконання лабораторної роботи проведено дослідження побудови векторних 3д зображень аксонометричної проекції на площину ХY за допомогою полінома Лагранжа в якості алгоритма інтерполяції.

Виконав: студент Тихонов Ф.С.