Übung 2

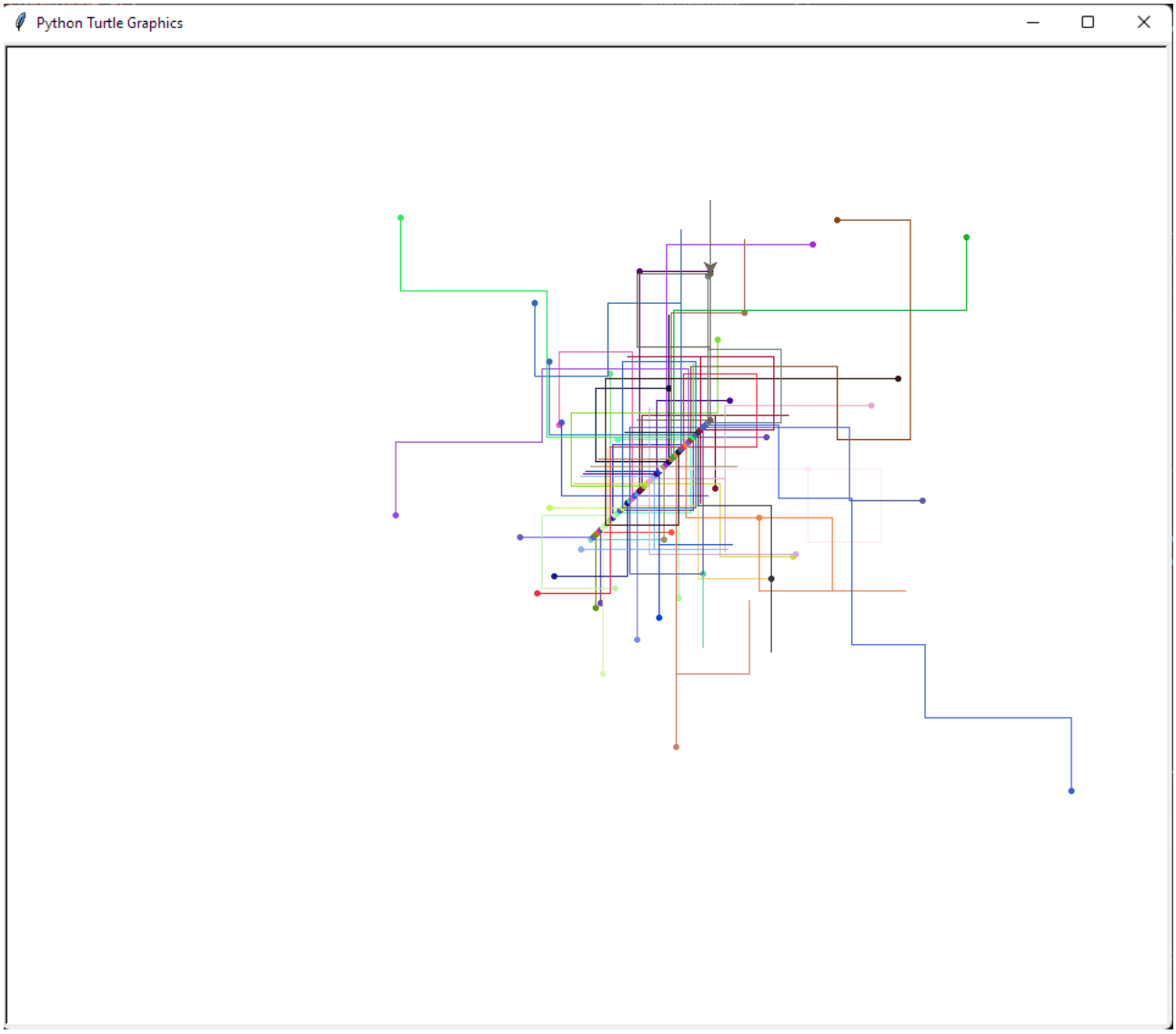
Table of Contents

1. Lösungsidee

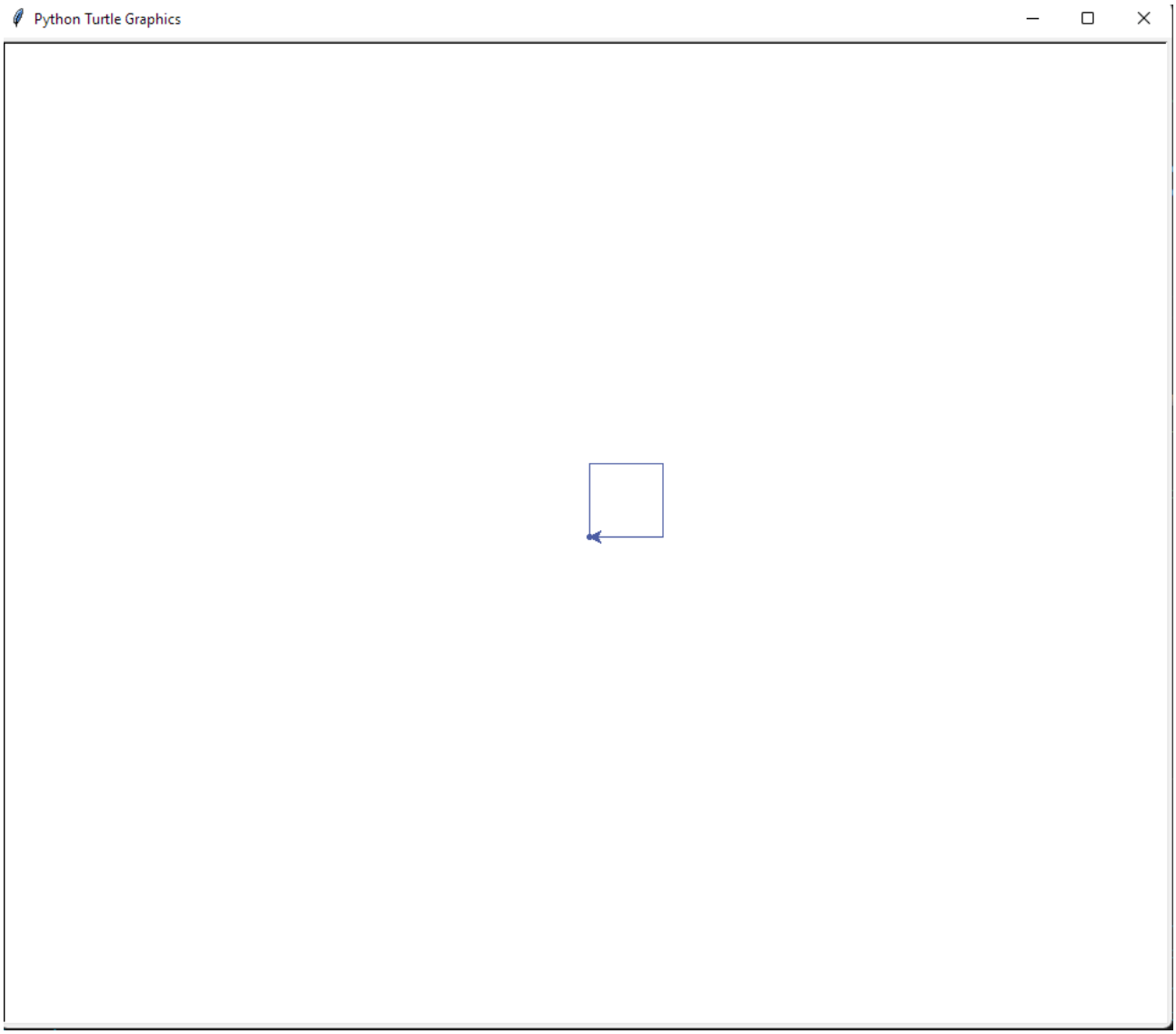
Für die Erstellung der Walk-Visualisierung wurde eine Prozedur **visualize_walk** erstellt. Dieser wird eine Turtle, ein Walk und eine Ausgangsposition übergeben. Anschließend wird für die übergebene Turtle die Ausgangsposition und die Farbe gesetzt. Für die Generierung eindeutiger Farben gibt es eine Methode **generate_unique_colors**, welche mittels matplotlib eine Anzahl an zufälligen Farben generiert. Anschließend wird ein Startpunkt gesetzt und in einer Schleife jeder Richtungseintrag des Walks durchlaufen. Für jeden Eintrag werden die entsprechenden Grad ermittelt und entsprechend rotiert. Damit immer von dem gleichen Gradwert ausgegangen wird, wird der vorherige Gradwert in einer Variable abgespeichert und abgezogen. Nach der Rechtsrotatien wird eine Linie gezogen und Prozess beginnt von vorne. Zum Schluss wird wieder ein Endpunkt gesetzt.

2. Testfälle

Bei einer maximalen Blockanzahl von 10 und 5 Wiederholungen mittels **monte_carlo_walk_analysis** sieht die Visualizierung folgendermaßen aus:



Wenn der Pfad von N, E, S, W visualisiert wird, sieht dies folgendermaßen aus:



3. Quellcode

```
import turtle
import Basic_Library
import matplotlib.pyplot as plt

def get_degree(direction):
    """
    Returns degree depending on direction

    Parameters:
        direction: Current direction

    Returns:
        Rotation in degrees
    """
    if direction == 'N':
        return 270
    if direction == 'E':
        return 0
    elif direction == 'S':
        return 90
    elif direction == 'W':
        return 180
    else:
        raise ValueError("Invalid direction")

def generate_unique_colors(n):
    """
    Generates n unique colors

    Parameters:
        n: Number of unique colors

    Returns:
        List with unique colors
    """
    return plt.get_cmap(lut=n, name="tab20c")


def visualize_walk(t: turtle.Turtle, walk, start_pos: turtle.Vec2D, color):
    """
    Visualizes a given path

    Parameters:
        t: Turtle object
        walk: List with given directions
        start_pos: The position where the drawing should start
    """
    # Set start position
    t.penup()
    t.goto(start_pos)
    t.pendown()
    # Set color and start point
    t.color(color[:-1])
    t.dot()
    # Set initial degree value
```

```
    previous_degree = 0
    for direction in walk:
        degree = get_degree(direction)
        t.right(-previous_degree + degree)
        previous_degree = degree
        t.forward(60)
    # Set end point
    t.dot()

def visualize_walks(t: turtle.Turtle, walks):
    """
    Visualizes given paths

    Parameters:
        walks: List with walks
    """
    t.speed(500)
    (x, y) = t.pos()
    colors = generate_unique_colors(len(walks))
    index = 0

    for walk in walks:
        visualize_walk(t, walk, (x, y), colors(index))
        index += 1
        # Add offset
        x += 2
        y += 2

blocks = 15
walks = [list(Basic_Library.generate_walk(blocks)) for _ in range(10)]
visualize_walks(turtle.Turtle(), walks)

turtle.mainloop()
```