Praktikum 1 CuF

Max Uhl

# Aufgabe 1

**Funktionsdefinition:**

* + get\_points(window: cf.WindowRasterized): Diese Funktion erfasst vier Mausklickpunkte auf einem Fenster und gibt sie zurück. Jeder Punkt wird durch eine Kreismarkierung in einer bestimmten Farbe visualisiert.
  + iteration(start\_point: cf.Point, points: List[cf.Point], window: cf.WindowRasterized) -> cf.Point: Diese Funktion nimmt einen Startpunkt, eine Liste von Punkten und ein Fensterobjekt entgegen. Sie wählt zufällig einen Punkt aus der Liste aus, berechnet den Mittelpunkt zwischen diesem ausgewählten Punkt und dem aktuellen Startpunkt und zeichnet einen Kreis an dieser Position. Der berechnete Punkt wird zurückgegeben.

**Hauptprogramm (if name == 'main'):**

* + Initialisiert die Anzahl der Iterationen (n\_iter), sowie die Dimensionen für das Fenster (i\_range, j\_range).
  + Erstellt ein Fensterobjekt window mit den angegebenen Dimensionen und stellt die Anzeigeskala ein.
  + Zeigt das Fenster an und gibt eine Benutzeranweisung aus, um vier Punkte durch Mausklicks zu setzen.
  + Führt eine bestimmte Anzahl von Iterationen durch, wobei in jeder Iteration die Funktion iteration aufgerufen wird, um einen neuen Punkt zu generieren. Der entstehende Punkt wird durch einen Kreis im Fenster visualisiert.
  + Das Programm zeigt die entstehenden Punkte in einem Fenster an und wartet darauf, dass der Benutzer eine Taste drückt, bevor es endet.

Hierbei sehen wir am Ende ein sogenanntes Sierpinsky Dreieck.

## Aufgabe 2

**Funktionsdefinition:**

* + readLut(filepath: str) -> List[List[str]]: Diese Funktion nimmt den Dateipfad einer CSV-Datei entgegen, liest die darin enthaltene Lookup-Tabelle (LUT) und gibt sie als Liste von Farben zurück.

**Hauptprogramm (if name == 'main'):**

* + Importiert erforderliche Bibliotheken (libcfcg, numpy, csv, os) und definiert den Dateipfad der LUT-Datei.
  + Ruft die Funktion readLut auf, um die LUT aus der CSV-Datei zu lesen und in der Variable lut zu speichern.
  + Initialisiert die Dimensionen für das Fenster (i\_range, j\_range) und erstellt ein Fensterobjekt window mit den angegebenen Dimensionen. Die Anzeigeskala wird ebenfalls eingestellt.
  + Erstellt zwei Arrays x\_arr und y\_arr als Listen von Zahlen von 0 bis i\_range-1 bzw. j\_range-1.
  + Erstellt eine 2D-Liste arr\_final, die die Bitwise-UND-Operation auf den Elementen von x\_arr und y\_arr anwendet.
  + Erstellt eine Kopie von arr\_final als colors und konvertiert dann arr\_final und colors in Tuple-Formate.
  + Erstellt ein Dictionary cmap, das die Punkte aus den Tuples und die zugehörigen Farben enthält.
  + Flacht die Liste von Punkten ab und speichert sie in der Variable points.
  + Durchläuft die Punkte im Fenster und setzt die Farben entsprechend der LUT-Datei.
  + Das Programm zeigt die entstehenden Punkte in einem Fenster an und wartet darauf, dass der Benutzer eine Taste drückt, bevor es endet.

Hierbei entstehen mehrere verschiedenfarbige und sich überlappende Sierpinsky-Dreiecke.

## Aufgabe 3

1. **Funktionsdefinitionen:**
   * get\_pascal\_matrix(dim: Tuple[int]): Diese Funktion erstellt eine modifizierte Pascal'sche Matrix der angegebenen Dimension (dim). Die Matrix wird als Liste von Listen dargestellt.
   * draw\_pascal(arr: list, divider: int, win: cf.WindowRasterized) -> List[Tuple[int, int]]: Diese Funktion nimmt eine Matrix (arr), einen Teiler (divider) und ein Fensterobjekt (win) entgegen. Sie durchläuft die Matrix und sammelt Punkte, die die Bedingung (arr[i][j] % divider != 0) erfüllen, in einer Liste und gibt sie zurück.
2. **Hauptprogramm (if name == 'main'):**
   * Importiert erforderliche Bibliotheken (libcfcg, sys, time, random) und definiert die Dimensionen für das Fenster (i\_range, j\_range) sowie die Dimension der Pascal'schen Matrix (dim).
   * Ruft die Funktion get\_pascal\_matrix auf, um die modifizierte Pascal'sche Matrix zu erhalten, und speichert sie in der Variable arr.
   * Erstellt ein Fensterobjekt window mit den angegebenen Dimensionen und setzt die Anzeigeskala.
   * Ruft die Funktion draw\_pascal auf, um Punkte gemäß der Farbregel zu erhalten und setzt die Farbe dieser Punkte im Fenster auf Rot.
   * Zeigt das Fenster an und wartet auf eine Benutzereingabe, bevor es endet.

Dieses Programm erzeugt ein visuelles Muster im Fenster, indem es Punkte basierend auf einer modifizierten Pascal'schen Matrix und einer Farbregel zeichnet. Die Farbregel besteht darin, Punkte auszuwählen, die nicht durch einen angegebenen Teiler teilbar sind, und diese Punkte in Rot zu färben. Hierbei erhalten wir ein Dreieck, welches aus sich überlappenden Sierpinsky-Dreiecken besteht, wobei die Anzahl und Beschaffung der SP-Dreiecke abhängig von der Anzahl und den Werten aller ganzzahligen Teiler von dem Teiler(divider).