НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
до лабораторної роботи №4
із дисципліни
«Алгоритмічні основи обчислювальної геометрії та комп'ютерної графіки»
із теми
«ДІАГРАМА ВОРОНОГО»

Виконав:
студент групи КМ-11
Пєшков А. Г.

Керівник:
Сирота С. В.

Київ — 2022

ЗМІСТ

ПОСТАНОВКА ЗАДАЧІ

Поставленою задачею була розробка програмного засобу, здібного за даним датасетом знайти центри ваги зв’язних точок з цього датасету, побудови за цими центрами діаграми Вороного та збереження вихідного зображення у графічному форматі. Розміри вихідного полотна - 960x540 пкс.

ІДЕЯ МЕТОДУ

Для побудови діаграми потрібно розбити задану множину точок на неперетинні множини, для цього у програмі запроваджено алгоритм заливки, що використовує стек та замальовує одразу цілі рядки [1].

Оскільки в умові роботи не було сказано, чи вважаються сусідніми точки, що знаходяться на відстані одного пікселю за віссю абсцис та ординат, задля спрощення програми було прийняте рішення не вважати їх такими, але алгоритм можна модифікувати і для цієї мети.

Після розбиття датасету на неперетинні регіони, для кожного регіону знаходяться центри мас як середні арифметичні усіх точок регіону, діаграма Вороного будується методом грубої сили, простим перебором усіх точок площини та порівнянням відстаней до кожної точки. Подібний перебір ефективно робити з використанням графічного процесору для паралельного замальовування сотень маленьких частин полотна, але мова Python не підтримує шейдери.

ОПИС РОБОТИ ПРОГРАМИ

Для виконання задачі було використано бібліотеку PIL мови Python. Модуль складається з классу Voronoi з публічним методом draw, який приймає шлях до датасету, шлях розміщення файлу з центрами областей (якщо присутній, для збереження їх у майбутньому ц метою економії часу) та шлях до розміщення вихідного зображення.

У випадку попереднього використання програми центри мас можуть бути збережені у окремому файлі; якщо такий файл існує, центри мас беруться з нього, інакше центри розраховуються з нуля (рис. 1).

Спочатку точки зчитуються з файлу датасету у множину (рис. 2), після чого, використовуючи описаний раніше алгоритм, з цих точок будуть братися стартові та будуть знаходитися зв’язні області, кожна точка яких буде видалятися з множини, доки усі вони не будуть використані (рис. 3).

У функцію замальовки передаються центри мас знайдених регіонів, після чого, пробігаючи по кожній точці полотна, знаходяться відстані до центрів й вибирається відповідний найближчому колір. На створене зображення наноситься початковий датасет й воно зберігається у відповідному розташуванні (рис. 4).

При запуску програми як виконавчого модуля буде побудовано діаграму Вороного з датасету *DS9.txt* та збережено у *result/output.png*.

```python
def draw(self, dataset_path: str, store_path: str, result_path: str):
    self._read_dots(dataset_path)

    if path.exists(store_path):
        with open(store_path, 'r') as file:
            regions = [
                (*map(int, reg.split()),)
                for reg in file.read().split('|')
            ]
    else:
        regions = self._get_regions()
        self._store_regions(regions, store_path)
    self._draw_regions(regions, result_path)
```

Рисунок 1

```python
def _read_dots(self, dataset_path: str):
    with open(dataset_path, 'r') as file:
        self.dots = {
            (*map(int, line.split()),)
            [::-1] for line in file.readlines()
        }
```

Рисунок 2

```
def _get_regions(self):
    def region_from(dot: tuple[int, int]):
        if dot not in dots:
            return
        region = []
        x, y = dot
        s = [(x, x, y, 1), (x, x, y - 1, -1)]
        while s:
            x1, x2, y, dy = s.pop()
            x = x1
            if (x, y) in dots:
                while (d := (x - 1, y)) in dots:
                    region.append(d)
                    dots.remove(d)
                    x -= 1
            if x < x1:
                s.append((x, x1 - 1, y - dy, -dy))
            while x1 <= x2:
                while (d := (x1, y)) in dots:
                    region.append(d)
                    dots.remove(d)
                    x1 += 1
                    s.append((x, x1 - 1, y + dy, dy))
                    if x1 - 1 > x2:
                        s.append((x2 + 1, x1 - 1, y - dy, -dy))
                x1 += 1
                while x1 < x2 and ((x1, y) not in dots):
                    x1 += 1
                x = x1
        return region

    dots = self.dots.copy()
    regions = []
    while dots:
        dots.add(dot := dots.pop())
        regions.append(region_from(dot))
    return [*map(lambda rg: (*map(self._int_avg, zip(*rg)),), regions)]
```

Рисунок 3

```
def _draw_regions(self, regions, result_path: str):
    def dist_sqr(p1, p2):
        return abs(p1[0] - p2[0]) ** 2 + abs(p1[1] - p2[1]) ** 2

    image = img.new('RGBA', (960, 540), (255, 255, 255))
    cols = {p: self._random_color() for p in regions}
    d = draw.ImageDraw(image)

    image.putdata([
        cols[min(((p, dist_sqr((x, y), p)) for p in regions),
                 key=lambda a: a[1]
                 )[0]]
        for y in range(self.height)
        for x in range(self.width)
    ])

    for p in regions:
        d.ellipse(((p[0] - 2, p[1] - 2), (p[0] + 2, p[1] + 2)), 0, 0, 1)

    for p in self.dots:
        color = tuple(max(a - 25, 0) for a in image.getpixel(p))
        image.putpixel(p, color)
    image.save(result_path)
```

Рисунок 4