Министерство образования Республики Беларусь
Учреждение образования “Белорусский государственный университет информатики и радиоэлектроники”

Факультет информационных технологий и управления
Кафедра интеллектуальных информационных технологий
Дисциплина: Графический интерфейс интеллектуальных систем

Отчёт к лабораторной работе №2

Выполнил: Гафаров М.С.
Группа: 221702
Проверила: Жмырко А.В.

Минск 2024

Лабораторная работа № 2

Алгоритмы построения линий второго порядка

Цель: изучить основные алгоритмы построения линий второго порядка

Задание: разработать элементарный графический редактор, реализующий построение линий второго порядка: окружность, эллипс, гипербола, парабола. Выбор кривой задаётся из пункта меню и доступен через панель инструментов “Линии второго порядка”. В редакторе кроме режима генерации линий второго порядка в пользовательском окне должен быть предусмотрен отладочный режим, где отображается пошаговое решение на дискретной сетке.

Теоретические сведения:

Алгоритм генерации окружности заключается в нахождении такого множества пикселей, которые наилучшим образом аппроксимируют кривую. Главное требование, предъявляемое к алгоритму, - эффективность.

Алгоритм Брезенхема рисует часть кривой в первом октанте – обычно этого достаточно для полной картинки, т.к. кривая может быть просто отражена относительно других квадрантов. Центральным понятием алгоритма является ошибка – разность между центром пикселя и действительным положением кривой. Её вычисление зависит от уравнения, задающего кривую. В лабораторной будут рассмотрены следующие виды кривых второго порядка:

1. Окружность
   Задаётся уравнением:
   $x^2 + y^2 = R^2$
2. Эллипс
   Задаётся уравнением:
   $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
3. Гипербола
   Задаётся уравнением:
   $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$
4. Парабола
   Задаётся уравнением:
   $y^2 = 2px$

Программная реализация:

Окружность:

Основной алгоритм:

```
def circle(center_coord, radius):  3 usages
    x_values, y_values = list(), list()
    x = int(center_coord[0])
    R = int(radius)
    y = R
    limit = int(center_coord[1])
    delta = 2 - 2*R
    x_values.append(x)
    y_values.append(y)
    while y > limit:
        if delta > 0:
            d = 2*delta - 2*x - 1
            if d > 0:
                y -= 1
                delta = delta - 2*y + 1
            else:
                x += 1
                y -= 1
                delta = delta + 2*x - 2*y + 2
            x_values.append(x)
            y_values.append(y)
        elif delta < 0:
            d = 2*delta + 2*y - 1
            if d > 0:
                x += 1
                y -= 1
                delta = delta + 2*x -2*y + 2
```

```
            else:
                x += 1
                delta = delta + 2*x + 1
            x_values.append(x)
            y_values.append(y)
        elif delta == 0:
            x += 1
            y -= 1
            delta = delta + 2*x - 2*y + 2
            x_values.append(x)
            y_values.append(y)
    return x_values, y_values
```

Отрисовка графика:

```python
def graphic(x_values, y_values):
    plt.figure(figsize=(8, 8))
    for x, y in zip(x_values, y_values):
        plt.fill(
            [x, x + 1, x + 1, x],
            [y, y, y + 1, y + 1],
            color='gray'
        )
    plt.axis('equal')
    plt.grid(visible=True, color='lightgray', linestyle='--', linewidth=0.5)
    plt.show()
```

Эллипс:

Основной алгоритм:

```python
def ellips(center_coord, a, b):  3 usages
    x_values, y_values = list(), list()
    a = int(a)
    b = int(b)
    x = int(center_coord[0])
    y = b
    limit = int(center_coord[1])
    x_values.append(x)
    y_values.append(y)
    delta = a**2 + b**2 - 2*b*(a**2)
    while y > limit:
        if delta > 0:
            d = 2 * (delta - x * (b ** 2)) - 1
            if d > 0:
                y -= 1
                delta = delta + (1 - 2 * y) * (a**2)
            else:
                x += 1
                y -= 1
                delta = delta + (2 * x + 1) * (b ** 2) + (1 - 2 * y) * (a ** 2)
            x_values.append(x)
            y_values.append(y)
```

```
    elif delta < 0:
        d = 2 * (delta + y * (a ** 2)) - 1
        if d > 0:
            x += 1
            y -= 1
            delta = delta + (2 * x + 1) * (b ** 2) + (1 - 2 * y) * (a ** 2)
        else:
            x += 1
            delta = delta + (2 * x + 1) * (b ** 2)
        x_values.append(x)
        y_values.append(y)
    elif delta == 0:
        x += 1
        y -= 1
        delta = delta + (2 * x + 1) * (b ** 2) + (1 - 2 * y) * (a ** 2)
return x_values, y_values
```

Отрисовка графика:

```
def graphic(x_values, y_values):
    plt.figure(figsize=(8, 8))
    for x, y in zip(x_values, y_values):
        plt.fill(
            [x, x + 1, x + 1, x],
            [y, y, y + 1, y + 1],
            color='gray'
        )
    plt.axis('equal')
    plt.grid(visible=True, color='lightgray', linestyle='--', linewidth=0.5)
    plt.show()
```

Гипербола:

Основной алгоритм:

```python
def hyperbola(a, b):  3 usages
    a, b = int(a), int(b)
    x, y = a, 0
    limit = 2 * b
    x_values, y_values = list(), list()
    x_values.append(x)
    y_values.append(y)
    delta = (x ** 2) * (b ** 2) - (y ** 2) * (a ** 2) - (a ** 2) * (b ** 2)
    while y < limit:
        if delta > 0:
            d = 2 * (delta - x * (b ** 2)) - 1
            if d > 0:
                y += 1
                delta = delta - (a ** 2) * (2 * y + 1)
            else:
                x += 1
                y += 1
                delta = delta + (b ** 2) * (2 * x + 1) - (a ** 2) * (2 * y + 1)
            x_values.append(x)
            y_values.append(y)
```

```python
        elif delta < 0:
            d = 2 * (delta + y * (a ** 2)) + 1
            if d > 0:
                x += 1
                y += 1
                delta = delta + (b ** 2) * (2 * x + 1) - (a ** 2) * (2 * y + 1
            else:
                x += 1
                delta = delta + (b ** 2) * (2 * x + 1)
            x_values.append(x)
            y_values.append(y)
        elif delta == 0:
            x += 1
            y += 1
            delta = delta + (b ** 2) * (2 * x + 1) - (a ** 2) * (2 * y + 1)
            x_values.append(x)
            y_values.append(y)
    return x_values, y_values
```

Отрисовка графика:

```python
def graphic(x_values, y_values):
    plt.figure(figsize=(8, 8))
    for x, y in zip(x_values, y_values):
        plt.fill(
            [x, x + 1, x + 1, x],
            [y, y, y + 1, y + 1],
            color='gray'
        )
    plt.axis('equal')
    plt.grid(visible=True, color='lightgray', linestyle='--', linewidth=0.5)
    plt.show()
```

Парабола:

Основной алгоритм:

```python
def parabola(p):  3 usages
    p = int(p)
    x, y = 0, 0
    x_values, y_values = list(), list()
    x_values.append(x)
    y_values.append(y)
    while x < 11:
        deltaH = (y ** 2) - (2 * p) * (x + 1)
        deltaV = (y + 1) * (y + 1) - 2 * p * x
        deltaD = (y + 1) * (y + 1) - (2 * p) * (x + 1)
        if ((abs(deltaD) < abs(deltaH) and abs(deltaD) < abs(deltaV)) or
                (abs(deltaD) == abs(deltaH) and abs(deltaD) < abs(deltaV)) or
                (abs(deltaD) == abs(deltaV)) and abs(deltaD) < abs(deltaH)):
            x += 1
            y += 1
            x_values.append(x)
            y_values.append(y)
        elif abs(deltaH) < abs(deltaD) and abs(deltaH) < abs(deltaV):
            x += 1
            x_values.append(x)
            y_values.append(y)
        elif abs(deltaV) < abs(deltaD) and abs(deltaV) < abs(deltaH):
            y += 1
            x_values.append(x)
            y_values.append(y)
    return x_values, y_values
```

Отрисовка графика:

```python
def graphic(x_values, y_values):
    plt.figure(figsize=(8, 8))
    for x, y in zip(x_values, y_values):
        plt.fill(
            [x, x + 1, x + 1, x],
            [y, y, y + 1, y + 1],
            color='gray'
        )
    plt.axis('equal')
    plt.grid(visible=True, color='lightgray', linestyle='--', linewidth=0.5)
    plt.show()
```

Вывод: в результате лабораторной работы изучил и запрограммировал алгоритм Брезенхема для отрисовки окружности. На основании алгоритма были разработаны и запрограммированы алгоритмы для построения частных случаев алгебраических кривых второго порядка; эллипса, гиперболы и параболы.