Лабораторная работа №8: «Программирование в графическом режиме»

Цель работы:

Познакомить студентов на практике с написанием программ для формирования графических изображений с использованием модулей Python.

Постановка задачи

Используя программный код из лабораторной работы №2, Задание и лабораторной работы №2, Задание 2, а также один из графических модулей Python, написать программы, которые иллюстрируют работу кода в графическом виде:

- Задание 1 - Решить обратную задачу - построить график функции, используя программный код, написанный к этому заданию;
- Задание 2 - построить графическое изображение, которое получается в результате генерации точек со случайными координатами. Использовать до 10000 точек. Оценить площадь заштрихованной фигуры методом Монте-Карло и сравнить с реальной. Оценку вывести в процентах по отношению к реальной площади.

Теоретическое введение

В Python разработано несколько модулей, обеспечивающих работу с графикой. Два графических модуля являются частью стандартной библиотеки Python:

- `turtle` (черепашка) - простой графический пакет, который так же может быть использован для создания несложного пользовательского графического интерфейса - Graphical User Interface (GUI);
- `tkinter` - разработан непосредственно для создания графического интерфейса пользователя GUI. Так, интерфейс IDLE построен с использованием `tkinter`.

Существуют и другие модули, из которых мы рассмотрим модуль `matplotlib`, который, по большей части, используется в научной среде и позволяет строить двумерную (2D) и трехмерную (3D) графику. Исходно этот пакет разрабатывался в подражание MATLAB, но является независимым от него проектом.

Другой путь написания графических приложений — это использование кроссплатформенных графических библиотек. Одной из кроссплатформенных графических библиотек является Qt. Эта библиотека используется с такими языками, как C++, Java, Ruby, Delphi, Lazarus, и др. У нее имеется "привязка" и к Python - PyQt.

Компьютерная графика - это довольно обширная и сложная область знания. Она включает знания о технических средствах, позволяющих отображать изображение: растровые и векторные дисплеи, плоттеры и принтеры. Знание о способах формирования цветного изображения, которое воспринимается либо как свечение отдельных точек дисплея, либо как отражение, например, от листа бумаги: аддитивная цветовая модель RGB или субтрактивная - CMY. Обширная область математических и физических знаний помогает решать вопросы перемещения, поворота объекта, формирования второго и

первого планов изображения (сцены), учета рефлексов (вторичных отражений) и еще много чего.

При построении графиков нам потребуются знания о функциях и методах графической системы, которые позволяют:

- формировать окно, в котором будет строиться график (размер, система координат);
- строить графические примитивы (точка, линия, ...);
- задавать ширину и цвет линий, цвет фона, выполнять заливку изображения или его части заданным цветом;
- управлять маркером (указателем), который используется для рисования
- наносить текст.

Модуль `turtle`

Модуль `turtle` входит в стандартную поставку библиотеки Python. Исходно этот модуль позиционируется как средство для обучения компьютерной графике детей.

Модель этого модуля следующая. Имеется прямоугольная поверхность, по которой ползает черепашка. Черепашка может перемещаться на заданное расстояние прямо, назад, под углом или по заданным координатам. Черепашку можно клонировать, создавая группу черепашек. При этом каждая черепашка живет своей жизнью. Для рисования черепашка использует цветовое перо (карандаш), которое может быть поднято или опущено. Если перо опущено, то остается след. Можно изменять цвет и толщину линии.

Черепашка понимает команды, с помощью которых можно нарисовать окружность заданного радиуса и цвета, дугу с заданным углом, залить фигуру определенным цветом, получить текущее состояние настроек или изменить их. Форма черепашки может быть изменена пользователем и использована как штамп, после которого на холсте остается рисунок.

В модуле `turtle` реализованы и интерактивные способы взаимодействия с черепашкой. События, связанные с кликом кнопки мыши или нажатия/отпускания клавиши клавиатуры, могут обрабатываться пользовательскими функциями, привязанными к этим событиям.

Для решения второй задачи (лабораторная работа №2, Задание 2), нам потребуется понимание того, что такое метод Монте-Карло и генератор случайных чисел.

Метод Монте-Карло - численный метод решения математических задач при помощи моделирования случайных величин (метод статистических испытаний).

Одним из простейших механизмов генерации случайных величин является рулетка - основной атрибут игорных домов. Европейский город, знаменитый игорными домами - Монте-Карло, столица княжества Монако. От имени этого города и пошло название метода.

Одной из задач, решаемых методом Монте-Карло, является задача вычисления площади или объема сложной фигуры. Суть метода в следующем. Если фигура изображена на плоскости, то около нее можно описать другую фигуру, площадь которой мы можем вычислить точно. Например, круг или правильный многоугольник. Генерируя N случайных точек, координаты которых будут равномерно распределены по поверхности описанной фигуры, мы можем подсчитать число точек, которые попали в фигуру с неизвестной площадью - Nr. Зная площадь описывающей фигуры S, можно вычислить площадь искомой фигуры с определенной точностью.

Листинг программы (по з. 2.1)

```
# -*- coding: UTF-8 -*-
from math import sqrt
import turtle as tr

def Fun1(x):
    if x <= -6:
        y = 1
    elif -6 < x < -4:
        y = -0.5 * x - 2
    elif -4 <= x <= 0:
        y = sqrt(4 - (x + 2) ** 2)
    elif 0 < x < 2:
        y = -1 * sqrt(1 - (x - 1) ** 2)
    else:
        y = -1 * x + 2
    return y


def Axis(txy, ax='X'):
        a = txy[0]
        b = txy[1]
        tr.up()
        if ax == 'X':
            pb = [a, 0]
            pe = [b, 0]
```

```
        else:
            pb = [0, a]
            pe = [0, b]
        tr.goto(pb)
        tr.down()
        tr.goto(pe)


def Mark(txy, ax='X'):
    a = txy[0]
    b = txy[1]
    tr.up()
    for t in range(a, b):
        if ax == 'X':
            pb = [t, 0]
            pe = [t, 0.2]
            pw = [t, -0.5]
        else:
            pb = [0, t]
            pe = [0.2, t]
            pw = [0.2, t]
        tr.goto(pb)
        tr.down()
        tr.goto(pe)
        tr.up()
        tr.goto(pw)
        tr.write(str(t))


def Arrow(txy, ax='X'):
    a = [0.1, 0, -0.1]
    b = [-0.1, 0.3, -0.1]
    tr.up()
    tr.goto(0, 0)
    tr.begin_poly()
```

```
    for i in range(2):
        tr.goto(a[i], b[i])
    tr.end_poly()
    p = tr.get_poly()
    tr.register_shape("myArrow", p)
    tr.resizemode("myArrow")
    tr.shapesize(1, 2, 1)
    if ax == 'X':
        tr.tiltangle(0)
        tr.goto(txy[1] + 0.2, 0)
        pw = [int(txy[1]), -1.0]
    else:
        tr.tiltangle(90)
        tr.goto(0, txy[1] + 0.2)
        pw = [0.2, int(txy[1])]
    tr.stamp()
    tr.goto(pw)
    tr.write(ax, font=("Arial", 14, "bold"))


def main():
    aX = [-12, 12]
    aY = [-3, 5]
    Dx = 800
    Dy = Dx / ((aX[1] - aX[0]) / (aY[1] - aY[0]))
    tr.setup(Dx, Dy)
    tr.reset()
    Nmax = 1000
    tr.setworldcoordinates(aX[0], aY[0], aX[1], aY[1])
    tr.title("Lab8")
    tr.width(2)
    tr.color("blue", "blue")
    tr.ht()
    tr.tracer(0, 0)
    Axis(aX, 'X')
```

```
    Mark(aX, 'X')
    Arrow(aX, 'X')
    Axis(aY, 'Y')
    Mark(aY, 'Y')
    Arrow(aY, 'Y')
    tr.color("green")
    tr.width(3)
    dx = (aX[1] - aX[0])/Nmax
    x = aX[0]
    y = Fun1(x)
    if (y is None):
        tr.up()
        tr.goto(x, 0)
    else:
        tr.goto(x, y)
        tr.down()
    while x <= aX[1]:
        x = x + dx
        y = Fun1(x)
        if (y is None):
            tr.up()
            continue
        else:
            tr.goto(x, y)
            tr.down()


if __name__ == "__main__":
    main()

tr.mainloop()
```

Описание подпрограмм

Функция Fun1(x)

Считает и возвращает y от введенного x

Функция Axis

Рисует оси

Функция Mark(txy, ax='X')

Рисует отметки на осях

Функция Arrow

Рисует стрелки

Функция main

Тело программы

Листинг программы (по з. 2.2)

```
# -*- coding: UTF-8 -*-
import turtle as tr
from random import uniform


def fun8(x, y):
    R = 6
    if (x < -6) or (x > 6):
        flag = 0
    if (y ** 2 >=2 * R * y - (x + R) ** 2 and -6 <= x <= 0 and 0
<=\ y <= 6) or (y ** 2 <= R ** 2 - x ** 2 and 0 <= x <= 6 and

-6 <= y <= 0):
        flag = 1
    else:
        flag = 0
    return flag


aX = [-8, 8]
aY = [-8, 8]
Dx = 300
Dy = Dx / ((aX[1] - aX[0]) / (aY[1] - aY[0]))
tr.setup(Dx, Dy, 200, 200)
tr.reset()
Nmax = 10000
tr.setworldcoordinates(aX[0], aY[0], aX[1], aY[1])
```

```
tr.title("lab8.2")
tr.width(2)
tr.ht()
tr.tracer(0, 0)
tr.up()
mfun = 0
for n in range(Nmax):
    x = uniform(aX[0], aX[1])
    y = uniform(aY[0], aY[1])
    tr.goto(x, y)
    if fun8(x, y) != 0:
        tr.dot(3, "green")
        mfun += 1
    else:
        tr.dot(3, "#ffccff")
tr.color("blue", "blue")
tr.up()
tr.goto(aX[0], 0)
tr.down()
tr.goto(aX[1], 0)
tr.up()
tr.goto(0, aY[1])
tr.down()
tr.goto(0, aY[0])
tr.up()
for x in range(aX[0], aX[1]):
    tr.goto(x, 0.1)
    tr.down()
    tr.goto(x, 0)
    tr.up()
    tr.sety(-0.4)
    coords = str(x)
    tr.write(coords)

for y in range(aY[0], aY[1]):
```

```
    tr.goto(0, y)
    tr.down()
    tr.goto(0.1, y)
    tr.up()
    tr.setx(0.2)
    coords = str(y)
    tr.write(coords)

poli = [0, 0.1, 0, -0.1, 0]
Arrbeg = int(aX[1])
Xpoli = [Arrbeg, Arrbeg - 0.1, Arrbeg + 0.3, Arrbeg - 0.1,
Arrbeg]
tr.goto(Xpoli[0], poli[0])
tr.begin_fill()
tr.down()
for i in range(1, 5):
    tr.goto(Xpoli[i], poli[i])
tr.end_fill()  # заливаем стрелку
# Надпишем ось X
tr.up()
tr.goto(Arrbeg, -0.7)
tr.write("X", font=("Arial", 14, "bold"))
#
# на ОСИ  y
Arrbeg = int(aY[1])
Ypoli = [Arrbeg, Arrbeg - 0.1, Arrbeg + 0.3, Arrbeg - 0.1,
Arrbeg]
tr.up()
tr.goto(poli[0], Ypoli[0])
tr.begin_fill()
tr.down()
for i in range(1, 5):
    tr.goto(poli[i], Ypoli[i])
tr.end_fill()
# Надпишем ось У
tr.up()
```

```
tr.goto(0.2, Arrbeg)
tr.write("Y", font=("Arial", 14, "bold"))
Sf = (aX[1] - aX[0]) * (aY[1] - aY[0]) * mfun / Nmax
tr.goto(1, 9)
meseg = "N = {0:8d}\nNf = {1:8d}\nSf = {2:8.2f}".format(Nmax,
mfun,\ Sf)
tr.write(meseg, font=("Arial", 12, "bold"))
print(meseg)
#
# Комментировать при работе в IDLE
tr.done()
```

Описание подпрограмм

Функция fun8 (x)

Считает и возвращает y от введенного x

Задание к лабораторной работе №8 «Программирование в графическом режиме»

Постановка задачи

Используя программный код из лабораторной работы №2, Задание и лабораторной работы №2, Задание 2, а также один из графических модулей Python, написать программы, которые иллюстрируют работу кода в графическом виде:

- Задание 1 - Решить обратную задачу - построить график функции, используя программный код, написанный к этому заданию;
- Задание 2 - построить графическое изображение, которое получается в результате генерации точек со случайными координатами. Использовать до 10000 точек. Оценить площадь заштрихованной фигуры методом Монте-Карло и сравнить с реальной. Оценку вывести в процентах по отношению к реальной площади.

Замечание: Эти задания лучше выполнить с модулем Matplotlib, подготовив отдельную лабораторную работу. С модулем Turtle лучше выполнить лабораторную работу 3.

Изображение должно занимать большую часть экрана, сопровождаться заголовком, содержать наименования и градации осей и масштабироваться в зависимости от исходных данных. При любых допустимых значениях исходных данных изображение должно полностью помещаться на экране. Программа не должна опираться на конкретные значения разрешения экрана.

Вывести на экран в графическом режиме графики двух функций на интервале от Хнач до Хкон с шагом dx. Первая функция задана с помощью ряда Тейлора, ее вычисление должно выполняться с точностью ε. Значение параметра b для функции вводится с клавиатуры. Графики должны быть плавными и различаться цветами.

Вариант 1

Вариант 1

Написать программу, которая выводит на экран секторную диаграмму. Диаграмму снабдить заголовком и наименованием для каждого сектора. Исходные данные сформировать в текстовом файле. Количество секторов задавать в программе в виде именованной константы.

Построение секторной диаграммы оформить в виде процедуры. Параметры процедуры: координаты центра диаграммы; радиус; количество секторов; массив процентов; массив наименований. Пример исходных данных см. Таблица 1.

Вариант 2

Написать программу, которая выводит на экран две секторные диаграммы, расположив их рядом. Диаграмму снабдить заголовком и наименованием для каждого сектора. Исходные данные сформировать в текстовом файле. Количество секторов задавать в программе в виде именованной константы.

Построение секторной диаграммы оформить в виде процедуры. Параметры процедуры: координаты центра диаграммы; радиус; количество секторов; массив

процентов; массив наименований. Пример исходных данных см. Таблица 1.

Вариант 3

Написать программу, которая выводит на экран две столбиковые диаграммы. На экране диаграммы расположить рядом, каждую в своих координатных осях. Каждую диаграмму снабдить заголовком и наименованием единиц измерений по осям X и У. Исходные данные сформировать в текстовом файле. Количество столбцов задавать в программе в виде именованной константы. Построение диаграммы оформить в виде процедуры. Пример исходных данных см. Таблица 1.

Вариант 4

Написать программу, которая выводит на экран две столбиковые диаграммы в одной координатной плоскости. Диаграмму снабдить градацией осей и заголовком. Исходные данные сформировать в текстовом файле. Количество столбцов задавать в программе в виде именованной константы. Построение диаграммы оформить в виде процедуры. Пример исходных данных см. Таблица 1.

Вариант 5

Написать программу, которая выводит на экран трехмерную столбиковую диаграмму. Диаграмму снабдить градацией осей л заголовком. Исходные данные сформировать в текстовом файле. Количество столбцов задавать в программе в виде именованной константы. Построение диаграммы оформить в виде процедуры. Пример исходных данных см. Таблица 1.

Вариант 6

Написать программу, которая выводит на экран столбиковую диаграмму, представляющую оптовые и розничные цены на различные наименования кофе. Исходные данные сформировать в текстовом файле.

Построение диаграммы оформить в виде процедуры. Параметры процедуры: количество наименований; массив значений оптовых цен; массив значений розничных цен; массив наименований. Наименования товаров разместить вертикально под осью абсцисс.

Вариант 7

Написать программу, которая выводит на экран столбиковую диаграмму, представляющую максимальную и среднюю норму прибыли при реализации различных сортов шоколада. Исходные данные сформировать в текстовом файле самостоятельно.

Построение диаграммы оформить в виде процедуры. Параметры процедуры: количество наименований; массив значений оптовых цен; массив значений розничных цен; массив наименований. Наименования товаров разместить вертикально под осью абсцисс.

Вариант 8

Написать программу, которая выводит на экран графики динамики изменения максимального, минимального и среднего курса доллара за заданное количество дней. Исходные данные сформировать в текстовом файле самостоятельно.

Построение графика оформить в виде процедуры. Параметры процедуры: массив дат; количество дней; массивы максимальных, минимальных и средних значений.

Вариант 9

Написать программу, которая выводит на экран трехмерную столбиковую диаграмму курса немецкой марки по отношению к рублю за заданное количество дней. Исходные данные сформировать в текстовом файле самостоятельно.

Построение диаграммы оформить в виде процедуры. Параметры процедуры: массив дат; количество дней; массив значений по оси У; код заполнителя.

Пример исходных данных

Таблица 1. Лидеры мирового рынка ПК

Рейтинг 1996 г.	Поставщик	Объем Продаж 1996 г. тыс. шт.	Доля Рынка 1996 г. , %	Объем Продаж 1995 г. тыс. шт.	Доля Рынка 1995 г. , %«	Рост 95/96, %
1	Compaq	7036	10,3	5757	9,8	22
2	IBM	6081	5,9	4785	8,1	27
3	Packard	4247	6,2	4392	7,5	-3
	Bell, NEC	3587	5,2	4627	7,9	22
4	Apple	2995	4,4	2023	3,4	48
5	HP	44459	65,0	37221	63,3	19
6	Другие					

Примечание: Попробуйте обновить информацию, получив необходимые данные из сети Интернет.