**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по лабораторной работе №4**

**по дисциплине «Компьютерная графика»**

**Тема: Исследования алгоритмов отсечения отрезков и многоугольников окнами разного вида.**

**Вариант 4**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 7301 |  | Гарцев Е.А.  Литвинов К.Л. |
| Преподаватель |  | Матвеева И. В. |

Санкт-Петербург

2020 г.

**Цель работы**

Научиться отсекать отрезки и многоугольники при помощи различных алгоритмов.

**Задание**

Обеспечить реализацию алгоритма отсечения массива произвольных отрезков заданным прямоугольным окном с использование метода половинного деления. Вначале следует вывести на экран сгенерированные отрезки полностью, а затем другим цветом или яркостью те, которые полностью или частично попадают в область окна.

**Математическая модель**

Для начала строится необходимое количество отрезков на плоскости. После этого пользователь задаёт размер и координаты левой верхней вершины окна.

Начинаем работу алгоритма с анализа тривиально невидимых и полностью видимых отрезков по отношению к окну при помощи определения 4-х битовыми кодами концов каждого отрезка. Если отрезок является таковым, то алгоритм переходит к анализу следующего отрезка. Если же отрезок пересекается окном, то алгоритм переходит на второй этап своей работы.

Второй этап начинается с нахождения средней точки анализируемого отрезка. Далее при помощи перевода в 4-х битовые коды начинает работу алгоритм деления отрезка пополам, который сравнивает каждую половину, находит невидимую, отбрасывает её и делит пополам видимую часть. Деление осуществляется до тех пор, пока не выйдут на границу окна, либо пока длина отрезка не станет равной или меньше одного пикселя. Результатом работы этого алгоритма будут две видимые точки, по которым можно построить видимый отрезок.

**Ход работы**

Данная работа выполнялась на языке Python с использованием PyQT для реализации интерфейса, pyqtgraph для удобного построения и отображения примитивов на двумерной плоскости, а также системных встроенных библиотек sys, design и math.

В программе реализовано следующее:

1. Возможность задать количество отрезков, строящихся на плоскости, вместе с координатами их вершин.
2. Возможность задать параметры прямоугольного окна: координаты левой верхней вершины, а также высоту и ширину в пикселях.
3. Построение и отображение отрезков и окна на дисплее, а также отсечение отрезков или их участков, находящихся в пределах окна.

**Выводы**

В процессе выполнения данной лабораторной работы был получен опыт отсечения отрезков и многоугольников различного вида при помощи использования различных алгоритмов.

**Листинг кода**

*Файл lab4.py*

import math

def pointToCode(point, window):

    T = []

    T.append(1 if point[0] < window[0] else 0)

    T.append(1 if point[0] > window[1] else 0)

    T.append(1 if point[1] < window[3] else 0)

    T.append(1 if point[1] > window[2] else 0)

    return T

def log\_prod(code1, code2):

    p = 0

    for i in range(4):

        p += code1[i] & code2[i]

    return p

def is\_visible(bar, rect):

    """Видимость - 0 = невидимый

                   1 = видимый

                   2 = частично видимый"""

    # вычисление кодов концевых точек отрезка

    s1 = sum(pointToCode(bar[0], rect))

    s2 = sum(pointToCode(bar[1], rect))

    # предположим, что отрезок частично видим

    vis = 2

    # проверка полной видимости отрезка

    if not s1 and not s2:

        vis = 1

    else:

        # проверка тривиальной невидимости отрезка

        l = log\_prod(pointToCode(bar[0], rect), pointToCode(bar[1], rect))

        if l != 0:

            vis = 0

    return vis

def midPoint(lines, rect, tol):

    result = []

    for line in lines:

        if line[0][0] >= line[1][0]:

            tempX = line[0][0]

            tempY = line[0][1]

            line[0][0] = line[1][0]

            line[0][1] = line[1][1]

            line[1][0] = tempX

            line[1][1] = tempY

        s1 = sum(pointToCode(line[0], rect))

        s2 = sum(pointToCode(line[1], rect))

        p = log\_prod(pointToCode(line[0], rect), pointToCode(line[1], rect))

        if s1 == 0 and s2 == 0:

            result.append(line)

            # break

        elif p != 0:

            break

        elif s1!=0 and s2!=0:

            tempX1 = line[0][0]

            tempY1 = line[0][1]

            tempX2 = line[1][0]

            tempY2 = line[1][1]

            leftBorder = find\_left\_border(line, rect, tol)

            rightBorder = find\_right\_border([[tempX1, tempY1], [tempX2, tempY2]], rect, tol)

            line[0][0] = leftBorder[0][0]

            line[0][1] = leftBorder[0][1]

            line[1][0] = rightBorder[1][0]

            line[1][1] = rightBorder[1][1]

            result.append(line)

        elif s1==0:

            tempX = line[0][0]

            tempY = line[0][1]

            border = find\_left\_border(line, rect, tol)

            line[0][0] = tempX

            line[0][1] = tempY

            line[1][0] = border[1][0]

            line[1][1] = border[1][1]

            result.append(line)

        elif s2==0:

            tempX = line[1][0]

            tempY = line[1][1]

            border = find\_right\_border(line, rect, tol)

            line[1][0] = tempX

            line[1][1] = tempY

            line[0][0] = border[0][0]

            line[0][1] = border[0][1]

            result.append(line)

    return result

def find\_left\_border(line, rect, tol):

    while math.sqrt((line[1][0] - line[0][0]) \*\* 2 + (line[1][1] - line[0][1]) \*\* 2) > tol:

        midLine = [(line[0][0] + line[1][0]) / 2, (line[0][1] + line[1][1]) / 2]

        temp = line[0]

        line[0] = midLine

        p = log\_prod(pointToCode(line[0], rect), pointToCode(line[1], rect))

        if p != 0:

            line[0] = temp

            line[1] = midLine

    return line

def find\_right\_border(line, rect, tol):

    while math.sqrt((line[1][0] - line[0][0]) \*\* 2 + (line[1][1] - line[0][1]) \*\* 2) > tol:

        midLine = [(line[0][0] + line[1][0]) / 2, (line[0][1] + line[1][1]) / 2]

        temp = line[1]

        line[1] = midLine

        p = log\_prod(pointToCode(line[0], rect), pointToCode(line[1], rect))

        if p != 0:

            line[1] = temp

            line[0] = midLine

    return  line

*Файл gui\_lab4.py*

import design

from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton, QMessageBox

from PyQt5.QtGui import QIcon

from PyQt5.QtCore import pyqtSlot

from PyQt5.QtGui import QPainter, QColor, QBrush

from lab4 import \*

from pyqtgraph import PlotWidget, plot, mkPen, mkColor

import pyqtgraph as pg

import sys

class App(QMainWindow, design.Ui\_MainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.setupUi(self)

        # self.dotsButton.clicked.connect(self.dotsPushed)

        # self.bezierButton.clicked.connect(self.bezierPushed)

        self.pushButton.clicked.connect(self.buttonPushed)

        self.linesTableButton.clicked.connect(self.lines\_button\_pushed)

        self.sectionButton.clicked.connect(self.section\_pushed)

        self.graphicsView.setBackground('w')

        self.lines = []

        self.foundLines = []

        self.window = []

        self.rectXLeft.setText(str(10))

        self.rectYLeft.setText(str(20))

        self.widthEdit.setText(str(10))

        self.lengthEdit.setText(str(10))

    @pyqtSlot()

    def lines\_button\_pushed(self):

        try:

            n = int(self.linesEdit.text())

            if n < 0:

                raise ValueError

        except ValueError:

            self.lines.clear()

            msg = QMessageBox()

            msg.setIcon(QMessageBox.Critical)

            msg.setText("Ошибка")

            msg.setInformativeText('Неправильно введено число точек')

            msg.setWindowTitle("Ошибка")

            msg.exec\_()

            return

        self.tableWidget.setRowCount(n)

        self.tableWidget.setColumnCount(4)

        # self.tableWidget.resizeColumnsToContents()

        self.tableWidget.setHorizontalHeaderLabels(['x1', 'y1', 'x2', 'y2'])

        self.tableWidget.setVerticalHeaderLabels(['P'+str(i+1) for i in range(n)])

    @pyqtSlot()

    def buttonPushed(self):

        self.lines.clear()

        try:

            for i in range(self.tableWidget.rowCount()):

                self.lines.append([[float(self.tableWidget.item(i,0).text()),

                                    float(self.tableWidget.item(i,1).text())],

                                   [float(self.tableWidget.item(i,2).text()),

                                    float(self.tableWidget.item(i,3).text())]])

        except Exception:

            self.lines.clear()

            msg = QMessageBox()

            msg.setIcon(QMessageBox.Critical)

            msg.setText("Ошибка")

            msg.setInformativeText('Неправильно введены координаты точек')

            msg.setWindowTitle("Ошибка")

            msg.exec\_()

            return

        try:

            x = float(self.rectXLeft.text())

            y = float(self.rectYLeft.text())

            length = float(self.lengthEdit.text())

            width = float(self.widthEdit.text())

            self.window = [x, x + width, y, y - length]

        except Exception:

            self.lines.clear()

            msg = QMessageBox()

            msg.setIcon(QMessageBox.Critical)

            msg.setText("Ошибка")

            msg.setInformativeText('Неправильно введены координаты точек')

            msg.setWindowTitle("Ошибка")

            msg.exec\_()

            return

        #self.lines = [[[15, 15], [15, 5]]]

        # Window

        # First = Xleft, Second = Xright

        # Third = Yup, Fourth = Ydown

        self.window = [10, 20, 20, 10]

        self.drawScene()

    def drawScene(self, linesPen=mkPen('k')):

        self.graphicsView.clear()

        pen = mkPen('r')

        # Up

        self.graphicsView.plot([self.window[0], self.window[1]], [self.window[2], self.window[2]], pen=pen)

        # Down

        self.graphicsView.plot([self.window[0], self.window[1]], [self.window[3], self.window[3]], pen=pen)

        # Left

        self.graphicsView.plot([self.window[0], self.window[0]], [self.window[2], self.window[3]], pen=pen)

        # Right

        self.graphicsView.plot([self.window[1], self.window[1]], [self.window[2], self.window[3]], pen=pen)

        # Draw lines

        for line in self.lines:

            self.graphicsView.plot([line[0][0], line[1][0]], [line[0][1], line[1][1]], pen=mkPen('k'))

    def section\_pushed(self):

        self.foundLines = midPoint(self.lines, self.window, 0.1)

        self.drawNewLines()

    def drawNewLines(self):

        # Draw lines

        for line in self.foundLines:

            self.graphicsView.plot([line[0][0], line[1][0]], [line[0][1], line[1][1]], pen=mkPen('g'))

if \_\_name\_\_ == '\_\_main\_\_':

    app = QApplication(sys.argv)

    ex = App()

    ex.show()

    sys.exit(app.exec\_())

*Файл design.py*

# -\*- coding: utf-8 -\*-

# Form implementation generated from reading ui file 'design.ui'

#

# Created by: PyQt5 UI code generator 5.14.1

#

# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

from  pyqtgraph import PlotWidget

class Ui\_MainWindow(object):

    def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(823, 597)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

        self.centralwidget.setObjectName("centralwidget")

        self.graphicsView = PlotWidget(self.centralwidget)

        self.graphicsView.setGeometry(QtCore.QRect(5, 21, 421, 481))

        self.graphicsView.setObjectName("graphicsView")

        self.gridLayoutWidget = QtWidgets.QWidget(self.centralwidget)

        self.gridLayoutWidget.setGeometry(QtCore.QRect(440, 20, 376, 481))

        self.gridLayoutWidget.setObjectName("gridLayoutWidget")

        self.gridLayout = QtWidgets.QGridLayout(self.gridLayoutWidget)

        self.gridLayout.setContentsMargins(0, 0, 0, 0)

        self.gridLayout.setObjectName("gridLayout")

        self.rectYLeft = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.rectYLeft.setObjectName("rectYLeft")

        self.gridLayout.addWidget(self.rectYLeft, 7, 1, 1, 1)

        self.pushButton = QtWidgets.QPushButton(self.gridLayoutWidget)

        self.pushButton.setObjectName("pushButton")

        self.gridLayout.addWidget(self.pushButton, 10, 0, 1, 2)

        self.tableWidget = QtWidgets.QTableWidget(self.gridLayoutWidget)

        self.tableWidget.setObjectName("tableWidget")

        self.tableWidget.setColumnCount(0)

        self.tableWidget.setRowCount(0)

        self.gridLayout.addWidget(self.tableWidget, 4, 0, 1, 2)

        self.lengthEdit = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.lengthEdit.setObjectName("lengthEdit")

        self.gridLayout.addWidget(self.lengthEdit, 9, 0, 1, 1)

        self.label\_3 = QtWidgets.QLabel(self.gridLayoutWidget)

        self.label\_3.setObjectName("label\_3")

        self.gridLayout.addWidget(self.label\_3, 6, 1, 1, 1)

        self.label\_2 = QtWidgets.QLabel(self.gridLayoutWidget)

        self.label\_2.setObjectName("label\_2")

        self.gridLayout.addWidget(self.label\_2, 6, 0, 1, 1)

        self.sectionButton = QtWidgets.QPushButton(self.gridLayoutWidget)

        self.sectionButton.setObjectName("sectionButton")

        self.gridLayout.addWidget(self.sectionButton, 11, 0, 1, 2)

        self.label\_6 = QtWidgets.QLabel(self.gridLayoutWidget)

        self.label\_6.setObjectName("label\_6")

        self.gridLayout.addWidget(self.label\_6, 3, 0, 1, 2)

        self.widthEdit = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.widthEdit.setObjectName("widthEdit")

        self.gridLayout.addWidget(self.widthEdit, 9, 1, 1, 1)

        self.label = QtWidgets.QLabel(self.gridLayoutWidget)

        self.label.setObjectName("label")

        self.gridLayout.addWidget(self.label, 5, 0, 1, 2)

        self.label\_4 = QtWidgets.QLabel(self.gridLayoutWidget)

        self.label\_4.setObjectName("label\_4")

        self.gridLayout.addWidget(self.label\_4, 8, 0, 1, 1)

        self.label\_5 = QtWidgets.QLabel(self.gridLayoutWidget)

        self.label\_5.setObjectName("label\_5")

        self.gridLayout.addWidget(self.label\_5, 8, 1, 1, 1)

        self.rectXLeft = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.rectXLeft.setObjectName("rectXLeft")

        self.gridLayout.addWidget(self.rectXLeft, 7, 0, 1, 1)

        self.linesEdit = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.linesEdit.setObjectName("linesEdit")

        self.gridLayout.addWidget(self.linesEdit, 1, 0, 1, 2)

        self.label\_7 = QtWidgets.QLabel(self.gridLayoutWidget)

        self.label\_7.setObjectName("label\_7")

        self.gridLayout.addWidget(self.label\_7, 0, 0, 1, 2)

        self.linesTableButton = QtWidgets.QPushButton(self.gridLayoutWidget)

        self.linesTableButton.setObjectName("linesTableButton")

        self.gridLayout.addWidget(self.linesTableButton, 2, 0, 1, 2)

        MainWindow.setCentralWidget(self.centralwidget)

        self.menubar = QtWidgets.QMenuBar(MainWindow)

        self.menubar.setGeometry(QtCore.QRect(0, 0, 823, 22))

        self.menubar.setObjectName("menubar")

        MainWindow.setMenuBar(self.menubar)

        self.statusbar = QtWidgets.QStatusBar(MainWindow)

        self.statusbar.setObjectName("statusbar")

        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)

        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):

        \_translate = QtCore.QCoreApplication.translate

        MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

        self.pushButton.setText(\_translate("MainWindow", "Построить отрезки и прямоугольник"))

        self.label\_3.setText(\_translate("MainWindow", "Верхний левый угол по у"))

        self.label\_2.setText(\_translate("MainWindow", "Верхний левый угол по х"))

        self.sectionButton.setText(\_translate("MainWindow", "Отсечь"))

        self.label\_6.setText(\_translate("MainWindow", "Введите координаты отрезков"))

        self.label.setText(\_translate("MainWindow", "Введите параметры прямоугольника"))

        self.label\_4.setText(\_translate("MainWindow", "Длина"))

        self.label\_5.setText(\_translate("MainWindow", "Ширина"))

        self.label\_7.setText(\_translate("MainWindow", "Введите количество отрезков"))

        self.linesTableButton.setText(\_translate("MainWindow", "Ввести координаты отрезков"))

from pyqtgraph import PlotWidget