МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

по дисциплине Компьютерная графика

Тема: «Исследование алгоритмов отсечения отрезков и многоугольников окнами различного вида»

Студенты гр. 1307 — Грунская Н.Д., Тростин М.Ю., Голубев М.А.

Преподаватель _____ Матвеева И.В.

Санкт-Петербург
2024

Цель работы:
Практическое закрепление теоретических знаний об алгоритмах отсечения отрезков и многоугольников окнами различного вида

Постановка задачи:
Обеспечить реализацию алгоритма отсечения массива произвольных отрезков заданным прямоугольным окном с использование алгоритма Коэна-Сазерленда. Вначале следует вывести на экран сгенерированные отрезки полностью, а затем другим цветом или яркостью те, которые полностью или частично попадают в область окна

Краткая теоретическая информация:

Схема Коэна-Сазерленда - один из первых алгоримов для быстрого отсечения линий

Этот алгоритм сравнивает концы отрезка с границами отсекателя (некоторой области на экране) и на основе результатов определяет, полностью ли отрезок видим, полностью ли скрыт или пересекается с отсекателем.

Точки классифицируются относительно границ отсекателя по коду в двоичной форме (например, верх, низ, право, лево).

Всего четыре границы окна формируют девять областей, а на рис. 1 перечислены значения двоичного кода во всех этих областях.

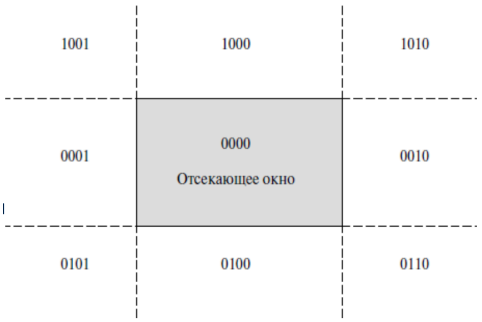

Рис. 1 - Девять областей окна с двоичными кодами

После определения кодов областей для всех конечных точек, определяют линии, которые полностью лежат внутри окна и очевидно лежащие снаружи.

Если конечные точки линий имеют код области 0000, эти отрезки целиком вмещаются в окно, они сохраняются.

Любая линия, конечные точки которой имеют 1 в одинаковых разрядах кода области, лежит полностью за пределами окна, и этот отрезок удаляется.

Если линии с помощью тестов кодов области нельзя отнести к полностью внешним или полностью внутренним, выполняется проверка на пересечения с границами окон.

Реализация алгоритма:
Определение двоичных кодов областей

```
INSIDE = 0   # 0000
LEFT = 1     # 0001
RIGHT = 2    # 0010
BOTTOM = 4   # 0100
TOP = 8      # 1000
```

Функция для определения двоичного кода точки

```
def compute_code(x, y):
    code = INSIDE
    if x < x_min:
        code |= LEFT
    elif x > x_max:
        code |= RIGHT
    if y < y_min:
        code |= BOTTOM
    elif y > y_max:
        code |= TOP
    return code
```

Функция для отображения отрезка с заданными координатами границ

```
def compute_and_draw(x1, y1, x2, y2):
    my_canvas.create_line(x1, y1, x2, y2, fill = 'blue')
    my_canvas.grid(row = 6, column = 0)
    code1 = compute_code(x1, y1)
    code2 = compute_code(x2, y2)
    accept = False
    while True:
        if code1 == 0 and code2 == 0:
            accept = True
            break
        elif (code1 & code2) != 0:
            break
        else:
            x = 1.0
            y = 1.0
            if code1 != 0:
                code_out = code1
            else:
                code_out = code2
            if code_out & TOP:
                x = x1 + ((x2 - x1) / (y2 - y1)) * (y_max
- y1)
                y = y_max
            elif code_out & BOTTOM:
                x = x1 + ((x2 - x1) / (y2 - y1)) * (y_min
- y1)
                y = y_min
            elif code_out & RIGHT:
                y = y1 + ((y2 - y1) / (x2 - x1)) * (x_max
- x1)
                x = x_max
            elif code_out & LEFT:
```

```
                y = y1 + ((y2 - y1) / (x2 - x1)) * (x_min
- x1)
                x = x_min
            if code_out == code1:
                x1 = x
                y1 = y
                code1 = compute_code(x1, y1)
            else:
                x2 = x
                y2 = y
                code2 = compute_code(x2, y2)
    if accept:
        my_canvas.create_line(x1, y1, x2, y2, fill =
'blue',  width = 1)
        my_canvas.grid(row = 6, column = 0)

        second_canvas.create_line(x1, y1, x2, y2, fill =
'green')
        second_canvas.grid(row = 6, column = 1)
    else:
        print("Line rejected”)
```

Пример работы приложения:

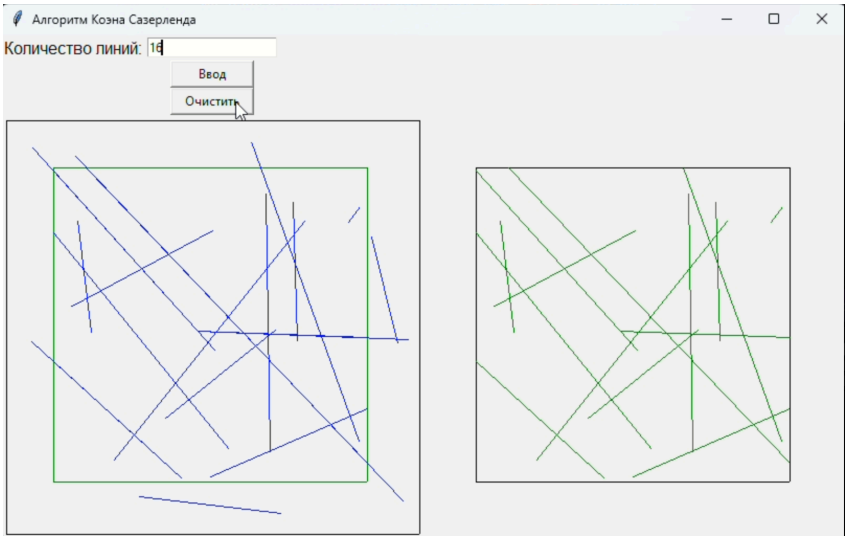

Рис. 2 - Пример работы приложения

Выводы:

В ходе выполнения работы были практически закреплены теоретические знания об алгоритмах отсечения отрезков и многоугольников окнами различного вида

Приложение
Ссылка на видео:

Исходный код программы:

```
import random
from tkinter import *

INSIDE = 0  # 0000
LEFT = 1    # 0001
RIGHT = 2   # 0010
BOTTOM = 4  # 0100
TOP = 8     # 1000

x_max = 350
y_max = 350
x_min = 50
y_min = 50

def compute_code(x, y):
    code = INSIDE
    if x < x_min:
        code |= LEFT
    elif x > x_max:
        code |= RIGHT
    if y < y_min:
        code |= BOTTOM
    elif y > y_max:
        code |= TOP
    return code

def compute_and_draw(x1, y1, x2, y2):
    my_canvas.create_line(x1, y1, x2, y2, fill = 'blue')
    my_canvas.grid(row = 6, column = 0)
    code1 = compute_code(x1, y1)
    code2 = compute_code(x2, y2)
    accept = False
    while True:

        if code1 == 0 and code2 == 0:
            accept = True
            break
        elif (code1 & code2) != 0:
            break
        else:
            x = 1.0
```

```
            y = 1.0
            if code1 != 0:
                code_out = code1
            else:
                code_out = code2

            # y = y1 + slope * (x - x1),
            # x = x1 + (1 / slope) * (y - y1)
            if code_out & TOP:
                x = x1 + ((x2 - x1) / (y2 - y1)) * (y_max
- y1)
                y = y_max
            elif code_out & BOTTOM:
                x = x1 + ((x2 - x1) / (y2 - y1)) * (y_min
- y1)
                y = y_min
            elif code_out & RIGHT:
                y = y1 + ((y2 - y1) / (x2 - x1)) * (x_max
- x1)
                x = x_max

            elif code_out & LEFT:
                y = y1 + ((y2 - y1) / (x2 - x1)) * (x_min
- x1)
                x = x_min

            if code_out == code1:
                x1 = x
                y1 = y
                code1 = compute_code(x1, y1)

            else:
                x2 = x
                y2 = y
                code2 = compute_code(x2, y2)

    if accept:
        my_canvas.create_line(x1, y1, x2, y2, fill =
'blue',  width = 1)
        my_canvas.grid(row = 6, column = 0)

        second_canvas.create_line(x1, y1, x2, y2, fill =
'green')
        second_canvas.grid(row = 6, column = 1)
```

```
    else:
        print("Line rejected")

def generate_lines():
    n = int(entry_n_lines.get())
    for i in range(n):
        x1 = float(random.randint(5, 400))
        y1 = float(random.randint(5, 400))
        x2 = float(random.randint(5, 400))
        y2 = float(random.randint(5, 400))
        compute_and_draw(x1, y1, x2, y2)

def clear_lines():
    my_canvas.delete("all")
    second_canvas.delete("all")

    my_canvas.create_line(5,400,400,400, fill = 'black')
    my_canvas.create_line(400,5,400,400, fill = 'black')
    my_canvas.create_line(5,5,5,400, fill = 'black')
    my_canvas.create_line(5,5,400,5, fill = 'black')

    my_canvas.create_line(50,350,350,350, fill = 'green')
    my_canvas.create_line(350,50,350,350, fill = 'green')
    my_canvas.create_line(50,50,50,350, fill = 'green')
    my_canvas.create_line(50,50,350,50, fill = 'green')

    second_canvas.create_line(50,350,350,350, fill =
'black')
    second_canvas.create_line(350,50,350,350, fill =
'black')
    second_canvas.create_line(50,50,50,350, fill =
'black')
    second_canvas.create_line(50,50,350,50, fill =
'black')

#   ui
my_window = Tk()
my_window.title("Алгоритм Коэна Сазерленда")

#   input
Label(my_window, text = "Количество линий: ", fg =
"black", font = "none 12").grid(row = 1, column = 0,
sticky = W)
entry_n_lines = Entry(my_window, width = 20, bg= "white")
```

```
entry_n_lines.grid (row = 1, column = 0)
#   button
Button(my_window, text = "Ввод", width = 10, command =
generate_lines).grid(row = 4, column = 0)

Button(my_window, text = "Очистить", width = 10, command =
clear_lines).grid(row = 5, column = 0)
#   canvas
my_canvas = Canvas(my_window, width = 400, height = 400)
my_canvas.grid(row = 6, column = 0)
my_canvas.create_line(5,400,400,400, fill = 'black')
my_canvas.create_line(400,5,400,400, fill = 'black')
my_canvas.create_line(5,5,5,400, fill = 'black')
my_canvas.create_line(5,5,400,5, fill = 'black')

my_canvas.create_line(50,350,350,350, fill = 'green')
my_canvas.create_line(350,50,350,350, fill = 'green')
my_canvas.create_line(50,50,50,350, fill = 'green')
my_canvas.create_line(50,50,350,50, fill = 'green')

second_canvas = Canvas(my_window, width = 400, height =
400)
second_canvas.grid(row = 6, column = 1)
second_canvas.create_line(50,350,350,350, fill = 'black')
second_canvas.create_line(350,50,350,350, fill = 'black')
second_canvas.create_line(50,50,50,350, fill = 'black')
second_canvas.create_line(50,50,350,50, fill = 'black')

my_window.mainloop()
```

Приложение:

Ссылка на видео:
https://www.youtube.com/watch?v=kJI2IWfSwnk