**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 7**

Тема: Построение плоских полиномиальных кривых.

Студент: Пищик Евгений Сергеевич

Группа: 80-306

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

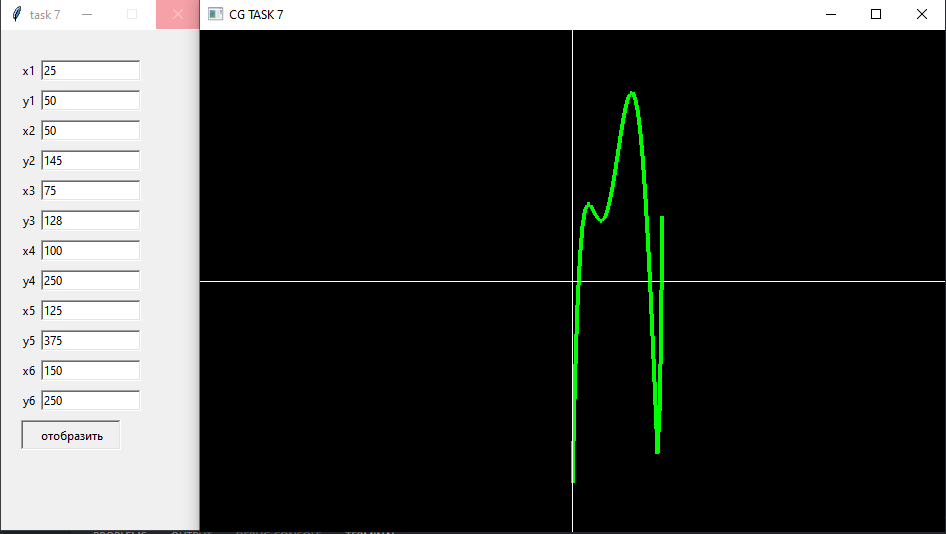
Москва, 2021

1. Постановка задачи

Написать программу, строящую полиномиальную кривую по заданным точкам. Обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.

1. Описание программы

Программа состоит из кода, создающего GUI при помощи библиотеки tkinter для ввода координат 6 пар точек (x,y), кода который при помощи библиотеки PyOpenGL отрисовывает заданную кривую при помощи линий соединяющих пары точек из массива координат.

1. Набор тестов и результаты их выполнения.
2. 
3. 
4. Листинг программы

main.py

import gui

def main():

gui.main()

if \_\_name\_\_ == '\_\_main\_\_':

main()

gui.py

import tkinter

import subprocess

def update(ent1, ent2, ent3, ent4, ent5, ent6, ent7, ent8, ent9, ent10, ent11, ent12):

try:

subprocess.run(f'C:/Users/jenja/anaconda3/python.exe C:\\Users\\jenja\\Downloads\\VSC\\computer\_graphics\\task\_7\\draw.py {ent1.get()} {ent2.get()} {ent3.get()} {ent4.get()} {ent5.get()} {ent6.get()} {ent7.get()} {ent8.get()} {ent9.get()} {ent10.get()} {ent11.get()} {ent12.get()}')

except Exception as \_:

subprocess.run(f'C:\\Users\\SuperPC\\anaconda3\\envs\\p39\\python.exe C:\\Users\\SuperPC\\Downloads\\VSC\\CG\\cg\_exercise\_07\\draw.py {ent1.get()} {ent2.get()} {ent3.get()} {ent4.get()} {ent5.get()} {ent6.get()} {ent7.get()} {ent8.get()} {ent9.get()} {ent10.get()} {ent11.get()} {ent12.get()}')

def main():

window = tkinter.Tk()

window.title("task 7")

window.geometry('200x500')

HL0 = 30

XL1 = 20

WL1 = 15

XL2 = 40

WL2 = 100

Y = [30 + i \* 30 for i in range(13)]

PARR = ((25, 50), (50, 145), (75, 128), (100, 250), (125, 375), (150, 250))

panel = tkinter.Frame(window, width=200, height=500)

panel.place(x=0, y=0, width=200, height=500)

lab1 = tkinter.Label(panel, text='x1')

lab1.place(x=XL1, y=Y[0], width=WL1)

lab2 = tkinter.Label(panel, text='y1')

lab2.place(x=XL1, y=Y[1], width=WL1)

lab3 = tkinter.Label(panel, text='x2')

lab3.place(x=XL1, y=Y[2], width=WL1)

lab4 = tkinter.Label(panel, text='y2')

lab4.place(x=XL1, y=Y[3], width=WL1)

lab5 = tkinter.Label(panel, text='x3')

lab5.place(x=XL1, y=Y[4], width=WL1)

lab6 = tkinter.Label(panel, text='y3')

lab6.place(x=XL1, y=Y[5], width=WL1)

lab7 = tkinter.Label(panel, text='x4')

lab7.place(x=XL1, y=Y[6], width=WL1)

lab8 = tkinter.Label(panel, text='y4')

lab8.place(x=XL1, y=Y[7], width=WL1)

lab9 = tkinter.Label(panel, text='x5')

lab9.place(x=XL1, y=Y[8], width=WL1)

lab10 = tkinter.Label(panel, text='y5')

lab10.place(x=XL1, y=Y[9], width=WL1)

lab11 = tkinter.Label(panel, text='x6')

lab11.place(x=XL1, y=Y[10], width=WL1)

lab12 = tkinter.Label(panel, text='y6')

lab12.place(x=XL1, y=Y[11], width=WL1)

ent1 = tkinter.Entry(panel, bd=2)

ent1.place(x=XL2, y=Y[0], width=WL2)

ent1.insert(0, f'{PARR[0][0]}')

ent2 = tkinter.Entry(panel, bd=2)

ent2.place(x=XL2, y=Y[1], width=WL2)

ent2.insert(0, f'{PARR[0][1]}')

ent3 = tkinter.Entry(panel, bd=2)

ent3.place(x=XL2, y=Y[2], width=WL2)

ent3.insert(0, f'{PARR[1][0]}')

ent4 = tkinter.Entry(panel, bd=2)

ent4.place(x=XL2, y=Y[3], width=WL2)

ent4.insert(0, f'{PARR[1][1]}')

ent5 = tkinter.Entry(panel, bd=2)

ent5.place(x=XL2, y=Y[4], width=WL2)

ent5.insert(0, f'{PARR[2][0]}')

ent6 = tkinter.Entry(panel, bd=2)

ent6.place(x=XL2, y=Y[5], width=WL2)

ent6.insert(0, f'{PARR[2][1]}')

ent7 = tkinter.Entry(panel, bd=2)

ent7.place(x=XL2, y=Y[6], width=WL2)

ent7.insert(0, f'{PARR[3][0]}')

ent8 = tkinter.Entry(panel, bd=2)

ent8.place(x=XL2, y=Y[7], width=WL2)

ent8.insert(0, f'{PARR[3][1]}')

ent9 = tkinter.Entry(panel, bd=2)

ent9.place(x=XL2, y=Y[8], width=WL2)

ent9.insert(0, f'{PARR[4][0]}')

ent10 = tkinter.Entry(panel, bd=2)

ent10.place(x=XL2, y=Y[9], width=WL2)

ent10.insert(0, f'{PARR[4][1]}')

ent11 = tkinter.Entry(panel, bd=2)

ent11.place(x=XL2, y=Y[10], width=WL2)

ent11.insert(0, f'{PARR[5][0]}')

ent12 = tkinter.Entry(panel, bd=2)

ent12.place(x=XL2, y=Y[11], width=WL2)

ent12.insert(0, f'{PARR[5][1]}')

but = tkinter.Button(panel, text="отобразить", command=lambda: update(ent1=ent1, ent2=ent2, ent3=ent3, ent4=ent4, ent5=ent5, ent6=ent6, ent7=ent7, ent8=ent8, ent9=ent9, ent10=ent10, ent11=ent11, ent12=ent12))

but.place(x=XL1, y=Y[12], width=WL2, height=HL0)

window.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()

draw.py

from typing import Tuple

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

import sys

def polyfunc(parr=None, eps=0.1):

x\_curr = 0

x\_lst = []

y\_lst = []

while x\_curr < W / 2:

lagr = 0

for i in range(len(parr)):

y = parr[i][1]

mul = 1.0

for j in range(len(parr)):

if i != j:

mul \*= (x\_curr - parr[j][0]) / (parr[i][0] - parr[j][0])

lagr += y \* mul

x\_lst.append(x\_curr)

y\_lst.append(lagr)

x\_curr += eps

return x\_lst, y\_lst

def draw(MARGIN=50.0, EPS=1.0):

parr = [(X1, Y1), (X2, Y2), (X3, Y3), (X4, Y4), (X5, Y5), (X6, Y6)]

curr\_w = glutGet(GLUT\_WINDOW\_WIDTH)

curr\_h = glutGet(GLUT\_WINDOW\_HEIGHT)

margin\_w = MARGIN / curr\_w

margin\_h = MARGIN / curr\_h

x\_lst, y\_lst = polyfunc(parr=parr, eps=EPS)

x\_lst = [(i / curr\_w) for i in x\_lst if (((i / curr\_w) > (-1.0 + margin\_w)) and ((i / curr\_w) < (1.0 - margin\_w)))]

y\_lst = [(i / curr\_h) for i in y\_lst if (((i / curr\_h) > (-1.0 + margin\_h)) and ((i / curr\_h) < (1.0 - margin\_h)))]

glColor4f(0, 1, 0, 1)

glLineWidth(4)

glBegin(GL\_LINE\_STRIP)

for x, y in zip(x\_lst, y\_lst):

glVertex2fv((x, y))

glEnd()

glColor4f(255, 255, 255, 1)

glLineWidth(1)

glBegin(GL\_LINE\_STRIP)

for x, y in zip([-1 \* W / 2 + margin\_w, W / 2 - margin\_w], [0.0, 0.0]):

glVertex2fv((x, y))

glEnd()

glBegin(GL\_LINE\_STRIP)

for x, y in zip([0.0, 0.0], [-1 \* H / 2 + margin\_h, H / 2 - margin\_h]):

glVertex2fv((x, y))

glEnd()

def display():

MARGIN = 50.0

EPS = 5.0

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glClearColor(0.0, 0.0, 0.0, 1.0)

glLoadIdentity()

draw(MARGIN=MARGIN, EPS=EPS)

glutSwapBuffers()

def main(x1, x2, x3, x4, x5, x6, y1, y2, y3, y4, y5, y6):

global X1, X2, X3, X4, X5, X6, Y1, Y2, Y3, Y4, Y5, Y6, W, H

X1, X2, X3, X4, X5, X6, Y1, Y2, Y3, Y4, Y5, Y6 = x1, x2, x3, x4, x5, x6, y1, y2, y3, y4, y5, y6

W, H = 720, 480

OUR\_W, OUR\_H = 1366, 768

glutInit()

glutInitDisplayMode(GLUT\_RGBA)

glutInitWindowSize(W, H)

glutInitWindowPosition(int((OUR\_W - W) / 2), int((OUR\_H - H) / 2))

glutCreateWindow('CG TASK 7')

glutDisplayFunc(display)

glutIdleFunc(display)

glutMainLoop()

if \_\_name\_\_ == '\_\_main\_\_':

try:

x1, y1, x2, y2, x3, y3, x4, y4, x5, y5, x6, y6 = float(sys.argv[1]), float(sys.argv[2]), float(sys.argv[3]), float(sys.argv[4]), float(sys.argv[5]), float(sys.argv[6]), float(sys.argv[7]), float(sys.argv[8]), float(sys.argv[9]), float(sys.argv[10]), float(sys.argv[11]), float(sys.argv[12])

main(x1, x2, x3, x4, x5, x6, y1, y2, y3, y4, y5, y6)

except Exception as e:

print(str(e))

ЛИТЕРАТУРА

1. Документация OpenGL[Электронный ресурс]. URL: <https://www.khronos.org/registry/OpenGL-Refpages/es3/> (дата обращения: 20.11.2021).

2. Документация tkinter[Электронный ресурс]. URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: 20.11.2021).