Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий  
Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Отчет

По лабораторной работе №2

«Реализация алгоритма удаления невидимых граней»

по дисциплине «Компьютерная графика»

Студент группы ПИ-02 Чередов Р.А.

Преподаватель Андреева А.Ю.

Барнаул 2023

Задание: Вариант 3

Метод Варнока.  
Код:

varnok.py

*"""Пакеты"""*import librosa  
import os  
import numpy as np  
from graphics import \*  
from shapely.geometry.polygon import Polygon as Pol  
from shapely.geometry import Point as Po  
  
  
class MyPoint(object):  
 *"""Точка - """* def \_\_init\_\_(self, x, y=0, z=0):  
 if type(x) is list:  
 self.x = x[0]  
 self.y = x[1]  
 self.z = x[2]  
 else:  
 self.x = x  
 self.y = y  
 self.z = z  
  
  
class MyPolygon(object):  
 *"""Полигон (точки, цвет)"""* def \_\_init\_\_(self, points, color=None):  
 if color is None:  
 self.color = color\_rgb(255, 0, 0)  
 else:  
 self.color = color  
  
 self.points = []  
 for point in points:  
 if type(point) is MyPoint:  
 self.points.append(point)  
 else:  
 self.points.append(MyPoint(point))  
  
  
def is\_inside(\_point, \_polygon):  
 *"""  
 Функция возращает True если координаты (x, y) внутри полигона  
 обозначеного списком вершин [(x1, x2), ... , (xN, yN)]  
 """* x = \_point.x # x  
 y = \_point.y # y  
  
 points = \_polygon.points # список вершин полигона  
 num\_points = len(points) # количество вершин полигона  
 inside\_flag = False # флаг вхождения  
  
 polygon\_x, polygon\_y = [points[0].x, points[0].y] # начальные точки полигона  
  
 for i in range(1, num\_points + 1):  
 point2x, point2y = [points[i % num\_points].x, points[i % num\_points].y]  
  
 if y > min(polygon\_y, point2y):  
 if y <= max(polygon\_y, point2y):  
 if x <= max(polygon\_x, point2x):  
 x\_inters = 0  
 if polygon\_y != point2y:  
 x\_inters = (y - polygon\_y) \* (point2x - polygon\_x) / (point2y - polygon\_y) + polygon\_x  
  
 if polygon\_x == point2x or x <= x\_inters:  
 inside\_flag = not inside\_flag  
  
 polygon\_x, polygon\_y = point2x, point2y  
 return inside\_flag  
  
  
def points\_in\_polygon(\_points, \_polygon):  
 *"""Количество точек в полигоне"""* num\_result = 0  
  
 for point in \_points:  
 if is\_inside(point, \_polygon):  
 num\_result += 1  
  
 return num\_result  
  
  
def split\_check(\_point\_f, \_point\_s, \_polygons, \_win, \_background=None):  
 *"""Проверка разбиения"""* if \_point\_s[0] - \_point\_f[0] > 1:  
 dx = \_point\_s[0] - \_point\_f[0]  
 dy = \_point\_s[1] - \_point\_f[1]  
  
 dx2 = int(dx / 2.)  
 dy2 = int(dy / 2.)  
  
 # Левый-верхний квадрат  
 check(\_point\_f, [\_point\_f[0] + dx2, \_point\_f[1] + dy2], \_polygons, \_win, \_background)  
  
 # Правый-верхний квадрат  
 check([\_point\_f[0] + dx2, \_point\_f[1]], [\_point\_s[0], \_point\_f[1] + dy2], \_polygons, \_win, \_background)  
  
 # Правый-нижний квадрат  
 check([\_point\_f[0] + dx2, \_point\_f[1] + dy2], \_point\_s, \_polygons, \_win, \_background)  
  
 # Левый-нижний квадрат  
 check([\_point\_f[0], \_point\_f[1] + dy2], [\_point\_f[0] + dx2, \_point\_s[1]], \_polygons, \_win, \_background)  
  
 else:  
 # Левый-верхний квадрат  
 check(\_point\_f, \_point\_f, \_polygons, \_win, \_background)  
  
 # Правый-верхний квадрат  
 check([\_point\_s[0], \_point\_f[1]], [\_point\_s[0], \_point\_f[1]], \_polygons, \_win, \_background)  
  
 # Правый-нижний квадрат  
 check(\_point\_s, \_point\_s, \_polygons, \_win, \_background)  
  
 # Левый-нижний квадрат  
 check([\_point\_f[0], \_point\_s[1]], [\_point\_f[0], \_point\_s[1]], \_polygons, \_win, \_background)  
  
  
def get\_z\_coordinate(\_point, \_polygon):  
 *"""Получаем координату z"""* m1 = \_polygon.points[0]  
 m2 = \_polygon.points[1]  
 m3 = \_polygon.points[2]  
  
 vec\_f = [m2.x - m1.x, m2.y - m1.y, m2.z - m1.z]  
 vec\_s = [m3.x - m1.x, m3.y - m1.y, m3.z - m1.z]  
  
 vec\_n = [vec\_f[1] \* vec\_s[2] - vec\_f[2] \* vec\_s[1],  
 vec\_f[2] \* vec\_s[0] - vec\_f[0] \* vec\_s[2],  
 vec\_f[0] \* vec\_s[1] - vec\_f[1] \* vec\_s[0]]  
  
 d = -m1.x \* vec\_n[0] - m1.y \* vec\_n[1] - m1.z \* vec\_n[2]  
  
 vec = [[vec\_n[0], \_point[0]],  
 [vec\_n[1], \_point[1]],  
 [vec\_n[2], 0]]  
  
 sums = [0, 0]  
 for i in range(3):  
 sums[0] += vec[i][0] \* vec\_n[i]  
 sums[1] += vec[i][1] \* vec\_n[i]  
  
 sums[1] += d  
 t = -sums[1] / sums[0]  
  
 x = vec\_n[0] \* t + \_point[0]  
 y = vec\_n[1] \* t + \_point[1]  
 z = vec\_n[2] \* t + 0  
  
 return x, y, z  
  
  
def check(\_point\_f, \_point\_s, \_polygons, \_win, \_background=None):  
 *"""Основной алгоритм Варнока"""* draw\_borders = False  
  
 # если не фона заливаем его цветом  
 if \_background is None:  
 \_background = color\_rgb(255, 255, 0)  
  
 # если точка одна, то проверям полигоны на её вхождение  
 if \_point\_f[0] == \_point\_s[0]:  
 max\_z = -np.inf  
 max\_polygon = None  
  
 # берем полигон  
 for polygon in \_polygons:  
 # если точка входит в полигон  
 if is\_inside(MyPoint(\_point\_f[0], \_point\_f[1]), polygon):  
 # получаем z  
 z = get\_z\_coordinate(\_point\_f, polygon)[2]  
 # если z больше запоминаем новые максимальные значения  
 if z > max\_z:  
 max\_polygon = polygon  
 max\_z = z  
  
 # если нет полигона  
 if max\_polygon is None:  
 # красим фоном  
 \_win.plotPixel(\_point\_f[0], \_point\_f[1], \_background)  
 else:  
 # красим цветом полигона  
 \_win.plotPixel(\_point\_f[0], \_point\_s[1], max\_polygon.color)  
 return  
  
 # квадрат  
 rec = MyPolygon([MyPoint(\_point\_f[0], \_point\_f[1], 0),  
 MyPoint(\_point\_f[0], \_point\_s[1], 0),  
 MyPoint(\_point\_s[0], \_point\_s[1], 0),  
 MyPoint(\_point\_s[0], \_point\_f[1], 0)])  
  
 # Считаем сколько вершин квадрата входят в каждый полигон  
 # и сколько вершин каждого полигона входят в квадрат  
 poly\_in\_rec = []  
 rec\_in\_polygon = []  
 for polygon in \_polygons:  
 rec\_in\_polygon.append(points\_in\_polygon(rec.points, polygon))  
 poly\_in\_rec.append(points\_in\_polygon(polygon.points, rec))  
  
 poly\_in\_rec = np.array(poly\_in\_rec)  
 rec\_in\_polygon = np.array(rec\_in\_polygon)  
  
 # Если в квадрат не внутри никакого полигона и внутри квадрата нет полигона,  
 # то зарисовываем квадрат фоном  
 if (poly\_in\_rec.max() == 0) and (rec\_in\_polygon.max() == 0):  
 rect = Rectangle(Point(\_point\_f[0], \_point\_f[1]), Point(\_point\_s[0] - 1, \_point\_s[1] - 1))  
 rect.setFill(\_background)  
 if not draw\_borders:  
 rect.setOutline(\_background)  
 rect.draw(\_win)  
 return  
  
 # Если внутри квадрата есть хоть 1 вершина полигона, разбиваем квадрат  
 split = False  
 for p in poly\_in\_rec:  
 if p > 0:  
 split = True  
 if split:  
 split\_check(\_point\_f, \_point\_s, polygons, win, \_background)  
 return  
  
 if len(polygons[np.where(rec\_in\_polygon > 0)]) > 1:  
 split\_check(\_point\_f, \_point\_s, polygons, win, \_background)  
 return  
  
 # Выбираем среди полигонов те, в которые квадрат входит полностью  
 # если таких больше 1, то разбиваем квадрат  
 rec\_inside\_polygon = \_polygons[np.where(rec\_in\_polygon == 4)]  
 if len(rec\_inside\_polygon) > 0:  
 rect = Rectangle(Point(\_point\_f[0], \_point\_f[1]), Point(\_point\_s[0] - 1, \_point\_s[1] - 1))  
 rect.setFill(rec\_inside\_polygon[0].color)  
 if not draw\_borders:  
 rect.setOutline(rec\_inside\_polygon[0].color)  
 rect.draw(\_win)  
 return  
  
 # Если хотябы одна вершина квадрата входит в какой-то полигон, то разбиваем квадрат  
 split = False  
 for p in rec\_in\_polygon:  
 if p > 0:  
 split = True  
 if split:  
 split\_check(\_point\_f, \_point\_s, \_polygons, \_win, \_background)  
 return  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 polygons = []  
 win = GraphWin("MyWin", 256, 256)  
 polygons.append(MyPolygon([[50, 150, 0], [150, 200, 00], [150, 100, 60]], color\_rgb(126, 20, 135))) #Фиолетова  
 polygons.append(MyPolygon([[50, 100, 0], [10, 200, 100], [200, 150, 0]], color\_rgb(18, 162, 184))) #Cиняя  
  
  
 polygons = np.array(polygons)  
  
 # Polygon([Point(100, 100), Point(150, 200), Point(50, 200)]).draw(win)  
 # Polygon([Point(150, 100), Point(200, 200), Point(250, 100)]).draw(win)  
  
 check([0, 0], [256, 256], polygons, win, color\_rgb(255, 255, 255))  
 # Polygon([Point(100, 100), Point(150, 200), Point(50, 200)]).draw(win)  
 # Polygon([Point(150, 100), Point(200, 200), Point(250, 100)]).draw(win)  
  
 win.getMouse()  
 win.close()

