**МИНОБРНАУКИ РОССИИ**

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

**Кафедра систем автоматизированного проектирования**

**ОТЧЁТ**

**по лабораторной работе №5  
по дисциплине «Компьютерная графика»**

**Тема: Исследование алгоритмов выявления видимости сложных сцен**

|  |  |
| --- | --- |
| Студенты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Матвеева И.В. |

Санкт-Петербург

2024

**Задание на лабораторную работу**

Вариант 1: Обеспечить реализацию алгоритма выявления видимых граней и ребер для одиночного выпуклого объемного тела.

**Теоретические положения**

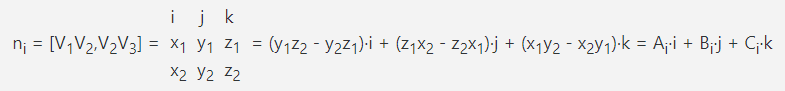
Алгоритм Робертса – представляет собой первое известное решение задачи об удалении невидимых линий. Это математически элегантный метод, работающий в объектном пространстве. Алгоритм прежде всего удаляет из каждого тела те ребра или грани, которые экранируются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, экранируются этими телами. Поэтому вычислительная трудоемкость алгоритма Робертса растет теоретически как квадрат числа объектов.

В алгоритме Робертса требуется, чтобы все изображаемые тела или объекты были выпуклыми. Не выпуклые тела должны быть разбиты на выпуклые части. В этом алгоритме выпуклое многогранное тело с плоскими гранями должно представляться набором пересекающихся плоскостей. Уравнение произвольной плоскости в трехмерном пространстве имеет вид

Если любая точка лежит на плоскости, то

.

Если же S не лежит на плоскости, то знак этого скалярного произведения показывает, по какую сторону от плоскости расположена точка. В алгоритме Робертса предполагается, что точки, лежащие внутри тела, дают положительное скалярное произведение.

Пусть – грани многогранника. Рассмотрим одну из граней. Обозначим вершины, инцидентные грани, через . Найдем вектор нормали к грани, вычислив векторное произведение любых двух смежных ребер этой грани и :

Тогда опорная функция грани имеет вид:

Величина D вычисляется с помощью произвольной точки на плоскости. В частности, если компоненты этой точки на плоскости , то:

Так как многогранник выпуклый, коэффициенты легко выбрать так, чтобы был вектором внешней нормали. Для этого найдем какую-либо внутреннюю точку, например, барицентр многогранника:

Если скалярное произведение уравнения плоскости и этой точки меньше 0, то необходимо поменять знак уравнения этой плоскости, чтобы отразить правильное направление внешней нормали. Остается только вычислить скалярное произведение уравнения плоскости на точку, в которой находится наблюдатель. Если это скалярное произведение меньше 0, то плоскость невидима и необходимо удалить весь многоугольник, лежащий в этой плоскости.

**Ход работы**

Алгоритм Робертса:

– вершины многогранника; – барицентр многогранника; – точка наблюдения

Выделим одну из граней многогранника:

Vec1.X = V1.X - V2.X;

Vec2.X = V3.X - V2.X;

Vec1.Y = V1.Y - V2.Y;

Vec2.Y = V3.Y - V2.Y;

Vec1.Z = V1.Z - V2.Z;

Vec2.Z = V3.Z - V2.Z;

// для этой грани найдем координаты двух векторов, которые лежат в плоскости грани

A = Vec1.Y·Vec2.Z - Vec2.Y·Vec1.Z;

B = Vec1.Z·Vec2.X - Vec2.Z·Vec1.X;

C = Vec1.X·Vec2.Y - Vec2.X·Vec1.Y;

D = -(A·V1.X + B·V1.Y + C·V1.Z);

// вычислим коэффициенты уравнения плоскости

m = -Sign(A·W.X + B·W.Y + C·W.Z + D);

// коэффициент, изменяющий знак плоскости

A = A·m;

B = B·m;

C = C·m;

D = D·m;

// корректируем направление плоскости

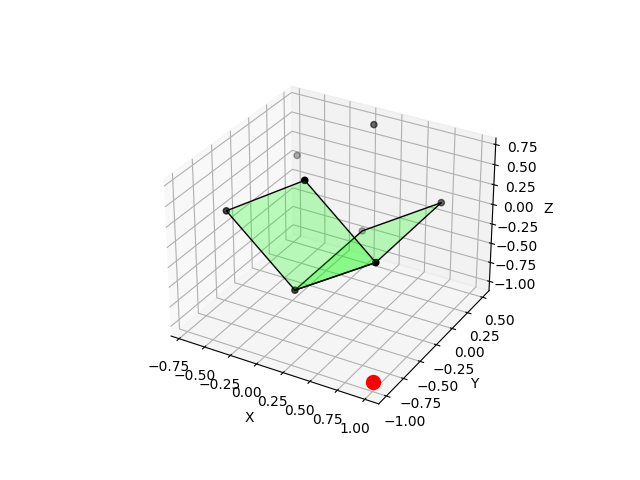
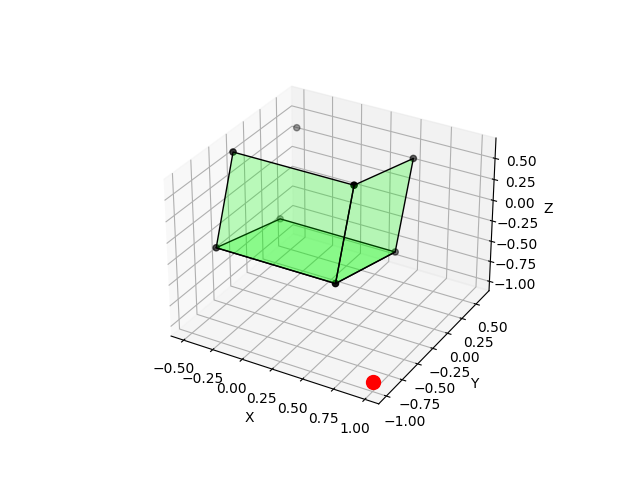
if A·P.X + B·P.Y + C·P.Z + D > 0 then грань видима; отобразить грань

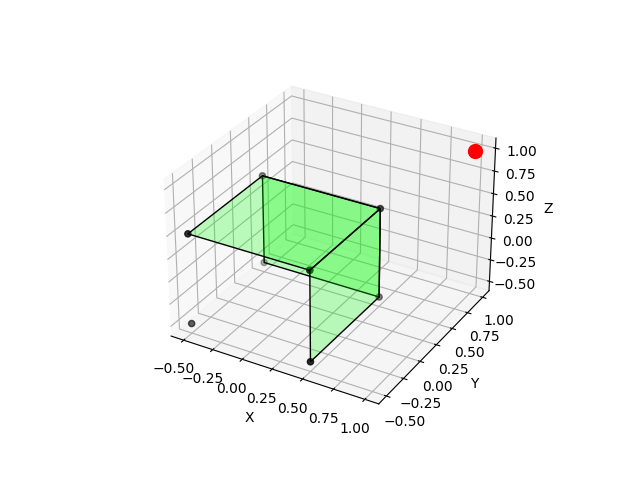
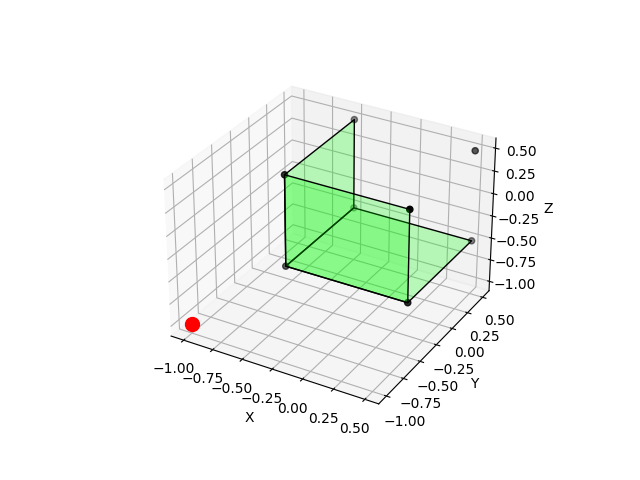
else грань невидима

Если задано только одно тело, то применив этот алгоритм для всех граней тела, поиск невидимых граней завершается. Если в сцене присутствует несколько тел, то следует сформировать приоритетный список этих тел по максимальным значениям координаты z вершин тел. Наибольшим приоритетом будет обладать тело, у которого минимальное значение z. Это тело будет самым удаленным от точки наблюдения, расположенной в бесконечности на оси z.

**Результаты лабораторной работы**

Программа была создана на языке Python с использованием графической библиотеки Tkinter.

Для работы программы нужно ввести с помощью слайдера координаты точки, из которой мы будем смотреть на фигуру (от -1 до 1 по каждой из осей), также можно выбрать ось и угол, на который будет осуществлен поворот фигуры и нажать кнопку “Показать”. 



**Вывод**

Обеспечили реализацию алгоритма выявления видимых граней и ребер для одиночного выпуклого объемного тела. Создали программу на языке Python с интерфейсом Tkinter. Интерфейс предоставляет возможность менять положения точки обзора на фигуру, а также вращение фигуры относительно одной из трех осей X, Y или Z.

**Приложение 1: исходный код программы**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d.art3d import Poly3DCollection

from tkinter import \*

from tkinter.ttk import \*

matrix = [

[0.5, -0.5, -0.5], [0.5, 0.5, -0.5], [-0.5, 0.5, -0.5], [-0.5, -0.5, -0.5],

[0.5, -0.5, 0.5], [0.5, 0.5, 0.5], [-0.5, 0.5, 0.5], [-0.5, -0.5, 0.5]

]

def plt\_show(verts, res\_matrix, eye):

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.scatter3D(res\_matrix[:, 0], res\_matrix[:, 1], res\_matrix[:, 2], color='black')

ax.add\_collection3d(Poly3DCollection(verts, facecolors='lime', linewidths=1, edgecolors='black', alpha=0.25))

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

ax.scatter(eye[0], eye[1], eye[2], color='red', s=100)

plt.show()

def rotate\_figure(matrix, os='x', angle=0):

cos\_angle = np.cos(angle)

sin\_angle = np.sin(angle)

def get\_rotation\_matrix(os):

switcher = {

'x': np.array([[1, 0, 0], [0, cos\_angle, sin\_angle], [0, -sin\_angle, cos\_angle]]),

'y': np.array([[cos\_angle, 0, -sin\_angle], [0, 1, 0], [sin\_angle, 0, cos\_angle]]),

'z': np.array([[cos\_angle, sin\_angle, 0], [-sin\_angle, cos\_angle, 0], [0, 0, 1]])

}

return switcher.get(os, np.eye(3))

rotation\_matrix = get\_rotation\_matrix(os)

res\_matrix = np.dot(matrix, rotation\_matrix)

verts = [[res\_matrix[0], res\_matrix[1], res\_matrix[2], res\_matrix[3]],

[res\_matrix[4], res\_matrix[5], res\_matrix[6], res\_matrix[7]],

[res\_matrix[0], res\_matrix[1], res\_matrix[5], res\_matrix[4]],

[res\_matrix[2], res\_matrix[3], res\_matrix[7], res\_matrix[6]],

[res\_matrix[1], res\_matrix[2], res\_matrix[6], res\_matrix[5]],

[res\_matrix[4], res\_matrix[7], res\_matrix[3], res\_matrix[0]]]

return [verts, res\_matrix]

def Roberts(data, eye):

verts = data[0]

W = np.zeros(3)

P = np.array(eye)

for i in range(2):

max\_value = -10

min\_value = 10

for point in verts:

max\_value1 = max(point[j][0] for j in range(len(point)))

max\_value = max(max\_value, max\_value1)

min\_value1 = min(point[j][0] for j in range(len(point)))

min\_value = min(min\_value, min\_value1)

W[i] = (max\_value + min\_value) / 2

new\_data = []

for vector in verts:

Vec1\_x = vector[0][0] - vector[1][0]

Vec2\_x = vector[2][0] - vector[1][0]

Vec1\_y = vector[0][1] - vector[1][1]

Vec2\_y = vector[2][1] - vector[1][1]

Vec1\_z = vector[0][2] - vector[1][2]

Vec2\_z = vector[2][2] - vector[1][2]

A = Vec1\_y \* Vec2\_z - Vec2\_y \* Vec1\_z

B = Vec1\_z \* Vec2\_x - Vec2\_z \* Vec1\_x

C = Vec1\_x \* Vec2\_y - Vec2\_x \* Vec1\_y

D = -(A \* vector[0][0] + B \* vector[0][1] + C \* vector[0][2])

m = -(A \* W[0] + B \* W[1] + C \* W[2] + D)

A = A \* m

B = B \* m

C = C \* m

D = D \* m

if (A \* P[0] + B \* P[1] + C \* P[2] + D) > 0:

new\_data.append(vector)

return new\_data

def update\_view():

eye = [eye\_x\_slider.get(), eye\_y\_slider.get(), eye\_z\_slider.get()]

angle = angle\_slider.get()

data = rotate\_figure(matrix, axis\_combo.get(), int(np.radians(angle)))

data[0] = Roberts(data, eye)

plt\_show(data[0], data[1], eye)

# Создание окна Tkinter

window = Tk()

window.title("Лаб. работа №4")

window.geometry('500x400')

window.resizable(False, False)

Label(window, text="Выберите ось для вращения", font=("Consolas", 14)).place(x=120, y=10)

axis\_combo = Combobox(window, values=["x", "y", "z"])

axis\_combo.current(1)

axis\_combo.place(x=175, y=40)

Label(window, text="Выберите угол вращения", font=("Consolas", 14)).place(x=135, y=70)

angle\_slider = Scale(window, from\_=0, to=360, orient=HORIZONTAL)

angle\_slider.set(0)

angle\_slider.place(x=200, y=110)

Label(window, text=f"{angle\_slider['from']}", font=("Consolas", 14)).place(x=135, y=110)

Label(window, text=f"{angle\_slider['to']}", font=("Consolas", 14)).place(x=335, y=110)

Label(window, text="Позиция обзора", font=("Consolas", 14)).place(x=175, y=150)

Label(window, text="X", font=("Consolas", 14)).place(x=90, y=200)

eye\_x\_slider = Scale(window, from\_=-1, to=1, orient=HORIZONTAL, length=200)

eye\_x\_slider.set(0)

eye\_x\_slider.place(x=150, y=200)

Label(window, text=f"{eye\_x\_slider['from']}", font=("Consolas", 14)).place(x=120, y=200)

Label(window, text=f"{eye\_x\_slider['to']}", font=("Consolas", 14)).place(x=360, y=200)

Label(window, text="Y", font=("Consolas", 14)).place(x=90, y=250)

eye\_y\_slider = Scale(window, from\_=-1, to=1, orient=HORIZONTAL, length=200)

eye\_y\_slider.set(0)

eye\_y\_slider.place(x=150, y=250)

Label(window, text=f"{eye\_y\_slider['from']}", font=("Consolas", 14)).place(x=120, y=250)

Label(window, text=f"{eye\_y\_slider['to']}", font=("Consolas", 14)).place(x=360, y=250)

Label(window, text="Z", font=("Consolas", 14)).place(x=90, y=300)

eye\_z\_slider = Scale(window, from\_=-1, to=1, orient=HORIZONTAL, length=200)

eye\_z\_slider.set(1)

eye\_z\_slider.place(x=150, y=300)

Label(window, text=f"{eye\_z\_slider['from']}", font=("Consolas", 14)).place(x=120, y=300)

Label(window, text=f"{eye\_z\_slider['to']}", font=("Consolas", 14)).place(x=360, y=300)

Button(window, text="Показать", command=update\_view).place(x=210, y=350)

window.mainloop()