**МИНОБРНАУКИ РОССИИ**

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

**Кафедра систем автоматизированного проектирования**

**ОТЧЁТ**

**по лабораторной работе №3  
по дисциплине «Компьютерная графика»**

**Тема: Формирования различных поверхностей с использованием ее**

**пространственного разворота и ортогонального проецирования на**

**плоскость при ее визуализации (выводе на экран дисплея)**

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

Санкт-Петербург

2024

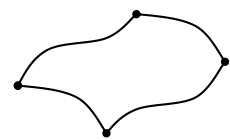
**Задание на лабораторную работу**

Вариант 1: Сформировать билинейную поверхность на основе произвольного задания ее четырех угловых точек. Обеспечить ее поворот относительно осей X и Y.

**Теоретические положения**

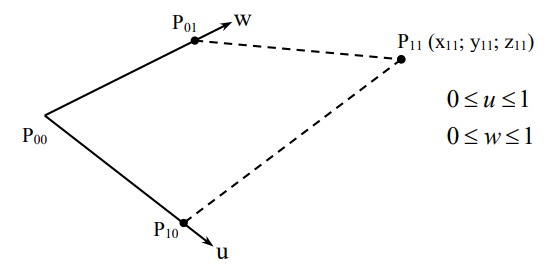
Поверхности задаются параметрически от двух независимых параметров u и w (отдельно по каждому параметру), т.е. можем задавать неоднозначные поверхности (т.е. для одного и того же значения одного параметра второй может иметь несколько значений): – параметрическая зависимость поверхности, позволяющая определить положение координат любой ее точки в функции от значений координат этой поверхности в заданных точках. При этом значение Q(u,w) на промежутках

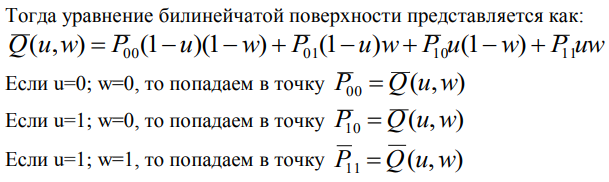
задания параметров u и w может определяться (меняться) непрерывно, а значения P(u,w) задаются для конкретных значений u и w.

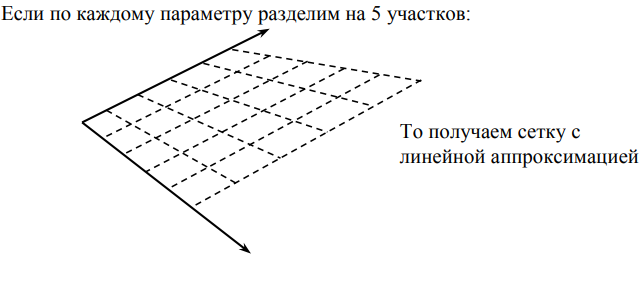
При этом координаты любой точки (X.Y и Z), относящейся к поверхности определяются исходя из соответствующих координат (X.Y и Z) точек задания и задающей функции, которая для всех координат одинаковая, т.е. и т.д.

Простейшими трехмерными поверхностями являются Билинейные

поверхности, их задают на ограниченном участке. Для такого участка поверхности требуется задание в пространстве 4-х угловых точек поверхности.







**Ход работы**

В данной работе мы используем метод билинейной интерполяции для поверхности в трехмерном пространстве. Он принимает на вход массив вершин (verts), который представляет собой матрицу 2x2, где каждый элемент матрицы является вектором в трехмерном пространстве. Далее, функция get\_matrix создает сетку точек (dots) в диапазоне от 0 до 1 с шагом, равным количеству точек N. Затем она преобразует все вершины в двумерный массив all\_verts, где каждая строка содержит координаты x и y для соответствующей точки. Затем в цикле для каждой точки вычисляются значения матриц matrix\_a и matrix\_b, которые используются для билинейной интерполяции. Значения этих матриц умножаются на соответствующие значения матрицы вершин bilinear\_matrix. Результат суммируется и сохраняется в массиве result.

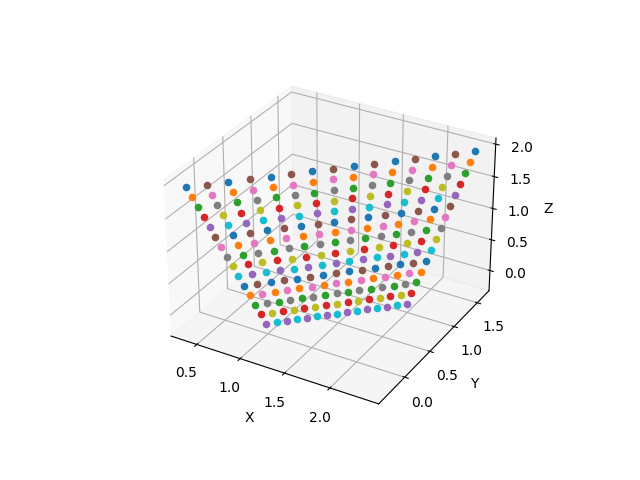
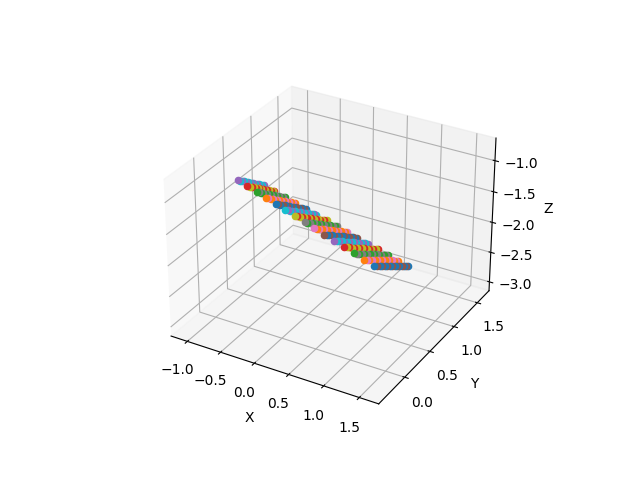
После завершения цикла вызова функции get\_matrix, массив result содержит координаты всех точек на поверхности. Эти точки затем отображаются на графике с помощью библиотеки matplotlib. Создаётся трёхмерный график, и каждая точка отображается в виде точки на графике.

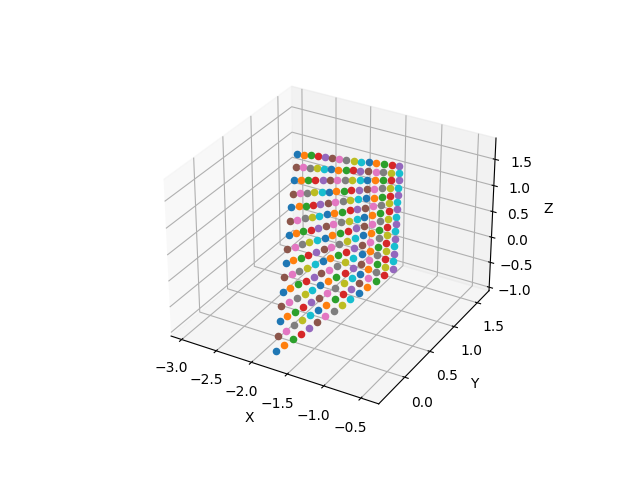
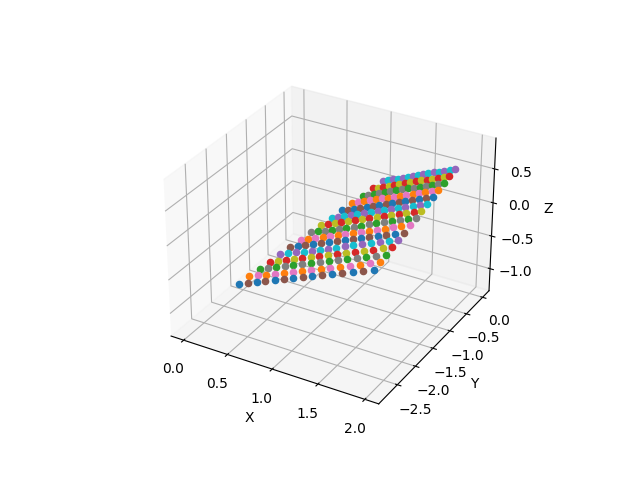
Поворот в нашей программе мы осуществляем за счет функции, которая принимает на вход угол α и использует его для вычисления косинуса (cos(α)) и синуса (sin(α)). Затем функция создаёт матрицу преобразования, используя значения косинуса и синуса в зависимости от выбранного варианта (x или y). После этого функция применяет преобразование к заданным точкам (res\_points) и вызывает функцию bilinear\_surface() для обработки преобразованных точек.

**Результаты лабораторной работы**

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

Для работы программы нужно ввести координаты 4-х угловых точек, которые сформируют билинейную поверхность и нажать кнопку “Построить!”. Для поворота поверхности на заданный угол нужно выбрать ось, относительно которой будет осуществлен поворот, ввести значение угла поворота и нажать кнопку “Повернуть!”.





**Вывод**

В ходе лабораторной работы сформировали билинейную поверхность на основе произвольного задания ее четырех угловых точек. Обеспечили ее поворот относительно осей X и Y. Создали интерфейс для удобного задания нужных координат точек поверхности, а также ее поворот относительно осей X и Y.

**Приложение 1: исходный код программы**

import numpy as np

import matplotlib.pyplot as plt

from tkinter import \*

from tkinter.ttk import \*

N = 15

def bilinear\_surface(verts):

def get\_matrix():

dots = np.linspace(0, 1, N)

all\_verts = np.transpose([np.tile(dots, len(dots)), np.repeat(dots, len(dots))])

for i in range(len(all\_verts)):

matrix\_a = np.array([1 - all\_verts[i][0], all\_verts[i][0]])

matrix\_b = np.array([[1 - all\_verts[i][1]], [all\_verts[i][1]]])

result[i] = np.dot(bilinear\_matrix[0][0], matrix\_a[0] \* matrix\_b[0][0]) + np.dot(bilinear\_matrix[0][1], matrix\_a[0] \* matrix\_b[1][0]) + np.dot(bilinear\_matrix[1][0], matrix\_a[1] \* matrix\_b[0][0]) + np.dot(bilinear\_matrix[1][1], matrix\_a[1] \* matrix\_b[1][0])

result = np.zeros((N \* N, 3))

bilinear\_matrix = np.array([[verts[0], verts[1]], [verts[2], verts[3]]])

get\_matrix()

result = np.array(result)

result\_x, result\_y, result\_z = np.array(result[:, 0]), np.array(result[:, 1]), np.array(result[:, 2])

fig = plt.figure()

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

r = [-1, 1]

np.meshgrid(r, r)

for i in range(len(result)):

ax.scatter(result\_x[i], result\_y[i], result\_z[i])

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

plt.show()

def main\_window():

res\_points = [[0, 0, 1], [1, 1, 1], [1, 0, 0], [0, 1, 0]]

def counts():

res\_points[0][0] = int(x1\_field.get())

res\_points[0][1] = int(y1\_field.get())

res\_points[0][2] = int(z1\_field.get())

res\_points[1][0] = int(x2\_field.get())

res\_points[1][1] = int(y2\_field.get())

res\_points[1][2] = int(z2\_field.get())

res\_points[2][0] = int(x3\_field.get())

res\_points[2][1] = int(y3\_field.get())

res\_points[2][2] = int(z3\_field.get())

res\_points[3][0] = int(x4\_field.get())

res\_points[3][1] = int(y4\_field.get())

res\_points[3][2] = int(z4\_field.get())

def click():

counts()

bilinear\_surface(res\_points)

def rotate():

alfa = int(angle\_field.get())

cos\_alfa = np.cos(alfa)

sin\_alfa = np.sin(alfa)

np.zeros((3, 3))

if os\_combo.get() == 'x':

matrix = np.array([[1, 0, 0], [0, cos\_alfa, -sin\_alfa], [0, sin\_alfa, cos\_alfa]])

else:

matrix = np.array([[cos\_alfa, 0, sin\_alfa], [0, 1, 0], [-sin\_alfa, 0, cos\_alfa]])

for i in range(4):

res\_points[i] = np.dot(res\_points[i], matrix)

bilinear\_surface(res\_points)

window = Tk()

window.title("Лабораторная работа №3")

window.geometry('400x400')

Label(window, text="Координаты 1 точки:", font=("Arial", 14)).place(x=10, y=10)

Label(window, text="x1: ").place(x=10, y=40)

x1\_field = Entry(window, width=10)

x1\_field.place(x=50, y=40)

Label(window, text="y1: ").place(x=140, y=40)

y1\_field = Entry(window, width=10)

y1\_field.place(x=180, y=40)

Label(window, text="z1: ").place(x=270, y=40)

z1\_field = Entry(window, width=10)

z1\_field.place(x=310, y=40)

Label(window, text="Координаты 2 точки:", font=("Arial", 14)).place(x=10, y=70)

Label(window, text="x2: ").place(x=10, y=100)

x2\_field = Entry(window, width=10)

x2\_field.place(x=50, y=100)

Label(window, text="y2: ").place(x=140, y=100)

y2\_field = Entry(window, width=10)

y2\_field.place(x=180, y=100)

Label(window, text="z2: ").place(x=270, y=100)

z2\_field = Entry(window, width=10)

z2\_field.place(x=310, y=100)

Label(window, text="Координаты 3 точки:", font=("Arial", 14)).place(x=10, y=130)

Label(window, text="x3: ").place(x=10, y=160)

x3\_field = Entry(window, width=10)

x3\_field.place(x=50, y=160)

Label(window, text="y3: ").place(x=140, y=160)

y3\_field = Entry(window, width=10)

y3\_field.place(x=180, y=160)

Label(window, text="z3: ").place(x=270, y=160)

z3\_field = Entry(window, width=10)

z3\_field.place(x=310, y=160)

Label(window, text="Координаты 4 точки:", font=("Arial", 14)).place(x=10, y=190)

Label(window, text="x4: ").place(x=10, y=220)

x4\_field = Entry(window, width=10)

x4\_field.place(x=50, y=220)

Label(window, text="y4: ").place(x=140, y=220)

y4\_field = Entry(window, width=10)

y4\_field.place(x=180, y=220)

Label(window, text="z4: ").place(x=270, y=220)

z4\_field = Entry(window, width=10)

z4\_field.place(x=310, y=220)

ok\_button = Button(window, text="Построить!", command=click)

ok\_button.place(x=180, y=260)

Label(window, text="Ось: ", font=("Arial", 14)).place(x=10, y=330)

os\_combo = Combobox(window)

os\_combo["values"] = ("x", "y")

os\_combo.current(0)

os\_combo.place(x=80, y=330)

Label(window, text="Угол:", font=("Arial", 14)).place(x=10, y=370)

angle\_field = Entry(window, width=10)

angle\_field.place(x=80, y=370)

rotate\_button = Button(window, text="Повернуть!", command=rotate)

rotate\_button.place(x=220, y=370)

window.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main\_window()