

**Московский авиационный институт
(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

**Курсовая работа
по курсу «Компьютерная графика»**

Студент:	Обыденкова Ю. Ю.
Группа:	М8О-308Б-18
Вариант:	2
Преподаватель:	Филиппов Г.С.
Оценка:	
Дата:	

Москва, 2020

Билинейная поверхность

Постановка задачи

Составить и отладить программу, обеспечивающую каркасную визуализацию порции поверхности заданного типа. Исходные данные готовятся самостоятельно и вводятся из файла или в панели ввода данных. Должна быть обеспечена возможность тестирования программы на различных наборах исходных данных. Программа должна обеспечивать выполнение аффинных преобразований для заданной порции поверхности, а также возможность управлять количеством изображаемых параметрических линий. Для визуализации параметрических линий поверхности разрешается использовать только функции отрисовки отрезков в экранных координатах.

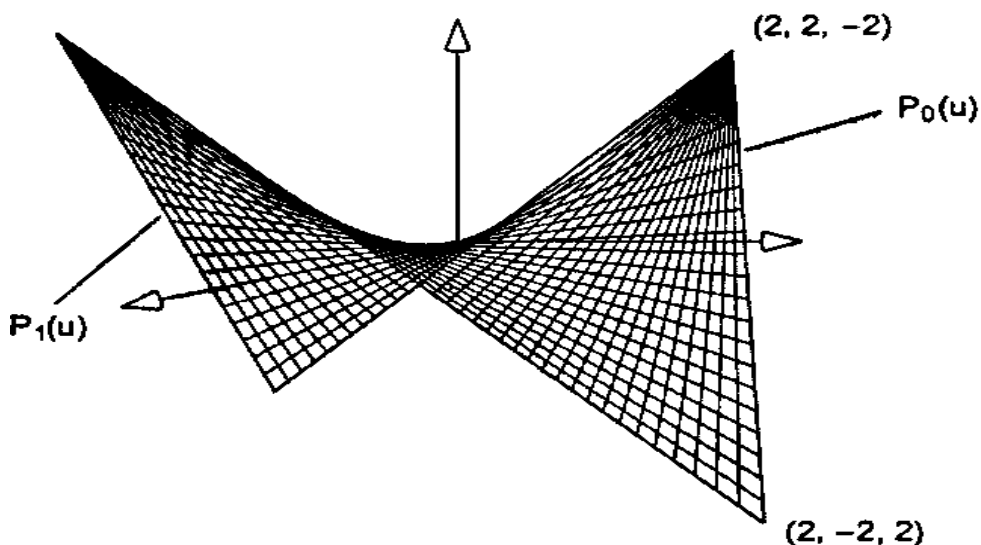
Вариант задания: 2. Билинейная поверхность

Решение задачи

Билинейная поверхность:

Билинейная интерполяция — обобщение линейной интерполяции одной переменной для функций двух переменных.

Обобщение основано на применении обычной линейной интерполяции сначала в направлении одной из координат, а затем в перпендикулярном направлении.



Билинейная поверхность конструируется из четырех угловых точек единичного квадрата в параметрическом пространстве, т.е. из точек $P(0, 0)$, $P(0, 1)$, $P(1, 0)$, $P(1, 1)$. Любая точка на поверхности определяется линейной интерполяцией между противоположными границами единичного квадрата. Любая точка внутри параметрического квадрата задается уравнением

$$Q(u, \omega) = P(0,0)(1-u)(1-\omega) + P(0,1)(1-u)\omega + P(1,0)u(1-\omega) + P(1,1)u\omega$$

В матричном виде:

$$Q(u, \varphi) = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} P(0,0) & P(0,1) \\ P(1,0) & P(1,1) \end{bmatrix} \begin{bmatrix} 1-\varphi \\ \varphi \end{bmatrix}$$

Необходимо, чтобы интерполируемая поверхность удовлетворяла исходным данным. В этом случае легко проверить, что угловые точки принадлежат этой поверхности, т.е. $Q(0, 0) = P(0, 0)$ и т.д.

Руководство по использованию программы

python3 "main.py"

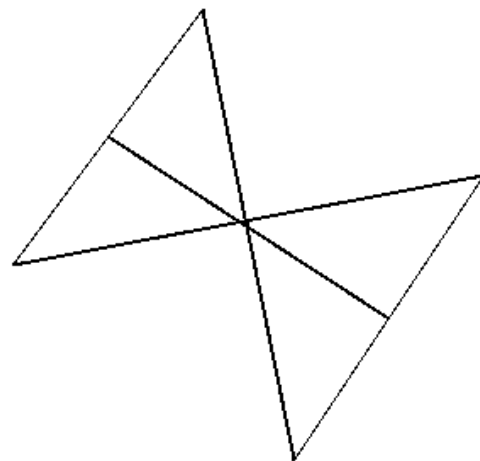
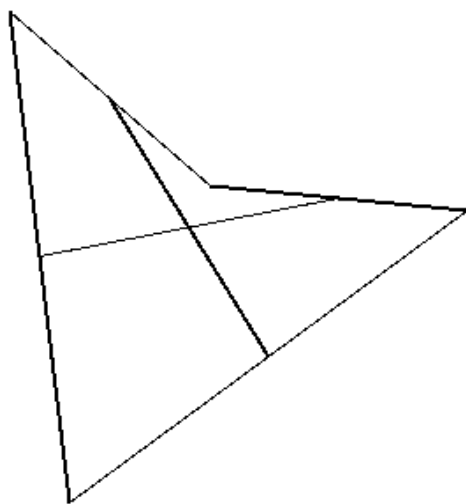
Шаг сетки (от 0 до 1): 0.5

1-ая точка: 0 0 1

2-ая точка: 1 1 1

3-я точка: 1 0 0

4-ая точка: 0 1 0



Код программы

```
import numpy as np
from tkinter import Tk, Canvas

unit_seq = 200

delta = float(input("Шаг сетки (от 0 до 1): "))

p00 = np.array([float(x) for x in input("1-ая точка: ").split()])
p01 = np.array([float(x) for x in input("2-ая точка: ").split()])
p10 = np.array([float(x) for x in input("3-я точка: ").split()])
p11 = np.array([float(x) for x in input("4-ая точка: ").split()])

def rotate_y(angle):
    c = np.cos(angle)
    s = np.sin(angle)
    return np.array([
        [c, -s, 0],
        [s, c, 0],
        [0, 0, 1]
    ])

def rotate_z(angle):
    c = np.cos(angle)
    s = np.sin(angle)
    return np.array([
        [c, 0, s],
        [0, 1, 0],
        [-s, 0, c]
    ])
```

```
def get_point(u, w):  
    return p00 * (1 - u) * (1 - w) + p01 * (1 - u) * w + p10 * u * (1 - w) + p11 * u *  
    w
```

```
def project(point):  
    return 400 + unit_seq * point[2], 400 - unit_seq * point[1]
```

```
def draw(c):  
    u = 0.0  
    while u <= 1.0:  
        x1, y1 = project(get_point(u, 0))  
        x2, y2 = project(get_point(u, 1))  
        c.create_line(x1, y1, x2, y2)  
        u += delta
```

```
    w = 0.0  
    while w <= 1.0:  
        x1, y1 = project(get_point(0, w))  
        x2, y2 = project(get_point(1, w))  
        c.create_line(x1, y1, x2, y2, width=2)  
        w += delta
```

```
def right_arrow(event):  
    global p00  
    global p01  
    global p10  
    global p11
```

```
    p00 = np.dot(rotate_z(0.05), p00)  
    p01 = np.dot(rotate_z(0.05), p01)  
    p10 = np.dot(rotate_z(0.05), p10)  
    p11 = np.dot(rotate_z(0.05), p11)
```

```
canvas.delete('all')
```

```
draw(canvas)
```

```
def left_arrow(event):
```

```
    global p00
```

```
    global p01
```

```
    global p10
```

```
    global p11
```

```
    p00 = np.dot(rotate_z(-0.05), p00)
```

```
    p01 = np.dot(rotate_z(-0.05), p01)
```

```
    p10 = np.dot(rotate_z(-0.05), p10)
```

```
    p11 = np.dot(rotate_z(-0.05), p11)
```

```
canvas.delete('all')
```

```
draw(canvas)
```

```
def up_arrow(event):
```

```
    global p00
```

```
    global p01
```

```
    global p10
```

```
    global p11
```

```
    p00 = np.dot(rotate_y(0.05), p00)
```

```
    p01 = np.dot(rotate_y(0.05), p01)
```

```
    p10 = np.dot(rotate_y(0.05), p10)
```

```
    p11 = np.dot(rotate_y(0.05), p11)
```

```
canvas.delete('all')
```

```
draw(canvas)
```

```
def down_arrow(event):
    global p00
    global p01
    global p10
    global p11

    p00 = np.dot(rotate_y(-0.05), p00)
    p01 = np.dot(rotate_y(-0.05), p01)
    p10 = np.dot(rotate_y(-0.05), p10)
    p11 = np.dot(rotate_y(-0.05), p11)

    canvas.delete('all')
    draw(canvas)

root = Tk()
root.title("Билинейная поверхность")
root.bind('d', right_arrow)
root.bind('a', left_arrow)
root.bind('w', up_arrow)
root.bind('s', down_arrow)
root.bind("<Escape>", exit)
canvas = Canvas(root, width=800, height=800, bg='white')
canvas.pack()
draw(canvas)
root.geometry("800x800")
root.mainloop()
```

Вывод

Выполнив курсовой проект, я научилась реализовывать билинейную поверхность. Выполнить данную работу было довольно интересно и в меру сложно.

Список литературы

1. http://scask.ru/a_book_mm3d.php?id=105
2. https://ru.wikipedia.org/wiki/Билинейная_интерполяция