Министерство высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

(НИЯУ МИФИ)

Лабораторная работа №1

Тема:ВИЗУАЛИЗАЦИЯ 3D ПЛОСКОСТИ

Старший преподаватель кафедры 22 «Кибернетика» НИЯУ МИФИ

Р.В. Душкин

Студент группы Б24-527

А. А. Кокотов

Москва 2025

**Цель работы:**

Изучить базовые математические операции для проекций и использовать их.

**Задачи**

1) понять базовые принципы и описать их

2) реализовать программу

**1.Параметрическая поверхность**

S(u,v)=

где:

- u ∈ [0, 4π] - параметр спирального вращения

- v ∈ [0, 2π] - параметр кручения

- a - радиус центральной оси

- b - радиус поперечного сечения

Геометрическая интерпретация:

- При a → 0 получается классический тор

- При b → 0 получается плоская спираль

- При ненулевых a и b - спираль с тороидальным сечением

**2. Вычисление нормали к поверхности**

Нормаль определяется через векторное произведение касательных векторов:

1. Касательные векторы:

T\_u = ∂S/∂u = [-(a+b·cos(v))·sin(u), (a+b·cos(v))·cos(u),a]

T\_v = ∂S/∂v = [-b·sin(v)·cos(u),-b·sin(v)·sin(u), b·cos(v)]

2. Вектор нормали вычисляется по формуле

N = T\_u × T\_v

*В коде*

*def normal(face):*

*return np.cross(face[1]-face[0], face[2]-face[0])*

**3. Модель освещения Ламберта**

Интенсивность освещения рассчитывается по формуле:

I = N·L / (|N|·|L|)

где:

- N - вектор нормали

- L - вектор направления на источник света

*В коде*

*light = np.array([10,-10,20]) # Положение источника*

*L = unit(light - point) # Направление света*

*N = unit(normal) # Нормализованная нормаль*

*intensity = N.dot(L) # Косинус угла между векторами*

*color = colormap(1-intensity) # Преобразование в цвет*

**4. Проекция 3D→2D**

Используется ортографическая проекция:

Для перевода точки (x,y,z) в 2D-координаты(x`,y`,z`)

используется матрица проекции:

Или в явном виде:

x′=x

y′=y+z

код

*projection = np.array([[1,0,0],[0,1,1]])*

*point\_2D = projection.dot(point\_3D)*

**5. Алгоритм сортировки полигонов**

Для корректного отображения перекрытий используется:

1. Вычисление расстояния до камеры:

d = |P - C|

где P - точка полигона, C - позиция камеры

2. Сортировка по убыванию расстояния:

*код*

*polygons.sort(key=lambda p: -distance(p, camera))*