МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Курсовая работа
по курсу «Компьютерная графика»

Группа: М8О-307Б-19
Студент: Бирюков В. В.
Преподаватель: Морозов А. В.
Оценка:
Дата:

Москва, 2021

Постановка задачи

Тема: Построение полиномиальных поверхностей.

Задание: Составить и отладить программу, обеспечивающую каркасную визуализацию порции поверхности заданного типа. Исходные данные готовятся самостоятельно и переключаются из графического интерфейса. Должна быть обеспечена возможность тестирования программы на различных наборах подготовленных исходных данных и их изменение. 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 или буфер вершин OpenGL. Реализовать возможность отображения опорных точек, направляющих и других данных по которым формируется порция поверхности и отключения каркасной визуализации.

Вариант: NURBS поверхность порядка 4x4

Описание

Элементарная NURBS поверхность задается при помощи 16 опорных точек P_{ij}, $i=0,\ldots,3$, $j=0,\ldots,3$ и описывается следующим уравнением

$$R(u,v)=\frac{\sum_{i=0}^{3}\sum_{i=0}^{3} w_{ij} n_i(u) n_j(v) P_{ij}}{\sum_{i=0}^{3}\sum_{i=0}^{3} w_{ij} n_i(u) n_j(v)},\ 0\le u,v\le 1$$

Функциональные коэффициенты $n_i(u)$ и $n_j(v)$ в котором задаются следующими формулами

$$n_0(t)=\frac{1}{6}(1-t)^3,\ n_1(t)=\frac{1}{6}(3t^3-6t^2+4),\ n_2(t)=\frac{1}{6}(-3t^3+3t^2+3t+1),\ n_3(t)=\frac{t^3}{6}.$$

Неотрицательные числа w_{ij}, сумма которых положительна, называются весами. В случае, если все веса w_{ij} равны между собой, элементарная NURBS поверхность вырождается в элементарную B-сплайновую поверхность.

По заданным параметрам аппроксимации n, m можно сгенерировать $(n+1)\cdot(m+1)$ точек с шагом $\frac{1}{n}$ по u и $\frac{1}{m}$ по v. Затем объединим эти точки в поверхность, состоящую из треугольных полигонов.

Имеется возможность применять к поверхности аффинные преобразования: сдвиг, вращение и масштабирование по любой оси.

Вариантов отображения поверхности несколько: каркасная визуализация, заливка полигонов сплошным цветом, модель затенения Фонга с одним источником света. Цвет и свойства материала объекта, а также параметры и положение света задаются пользователем. Реализована визуализация нормалей вершин трехмерного объекта, текущей ориентации координатных осей, опорных точек и опорной плоскости.

Для просмотра объекта используется перспективная камера. Реализовано управление камерой путем перемещения ее по поверхности сферы с заданным центром, в который постоянно направлен ее вектор взгляда. Также имеется возможность регулировать угол крена камеры.

Для возможности управления опорными точками, необходимо искать точку, находящуюся возле указателя. Для этого переведем координаты точек и координаты указателя в экранную систему координат. Координаты указателя достаточно поделить на длину и ширину OpenGL виджета соответственно, для координат точек же надо провести процесс нормализации, такой же, который выполнятся в шейдерах: умножить на видовую и проекционную матрицы и преобразовать из однородных координат, разделив на четвертую компоненту вектора.

В программе реализовано перемещение точек в пространстве, как перемещение по поверхности плоскости, параллельной проекционной плоскости камеры, а также изменение весового коэффициента точки.

Перемещение точек в пространстве можно производить с заданным шагом, что может упростить управление поверхностью, не давая точке перемещаться по ненужной оси.

Для программы подготовлен ряд тестовых наборов исходных данных, представляющих из себя конфигурационные текстовые файлы с координатами и весами шестнадцати опорных точек. Загрузку и сохранение файлов можно осуществлять непосредственно из интерфейса программы.

Исходный код

Генерация поверхности и функции для получения значения сплайна

```
public static float GetCoefficientPolynom(int i, float t)
{
    return i switch
    {
        0 => (1 - t) * (1 - t) * (1 - t) / 6,
        1 => ((3 * t - 6) * t * t + 4) / 6,
        2 => (((-3 * t + 3) * t + 3) * t + 1) / 6,
        3 => t * t * t / 6,
        _ => 0
    };
}

public Vector3 GetValue(float u, float v)
{
    Vector3 f = Vector3.Zero;
    float g = 0;
    for (int i = 0; i < 4; ++i)
    {
        for (int j = 0; j < 4; ++j)
        {
            float coeff = GetCoefficientPolynom(i, u) *
                          GetCoefficientPolynom(j, v);
            f += coeff * Points[i, j] * Weights[i, j];
            g += coeff * Weights[i, j];
        }
    }

    return f / g;
}

public void GenerateMesh(ref Mesh mesh, int uCount, int vCount,
                         Material material)
{
    mesh.Vertices.Clear();
    mesh.Polygons.Clear();
    uint vertexId = 0;

    float du = 1f / uCount, dv = 1f / vCount, u = 0, v;
    Vector3 value;
    for (int i = 0; i < uCount; ++i)
    {
        v = 0;
        for (int j = 0; j < vCount; ++j)
        {
            value = GetValue(u, v);
            mesh.Vertices.Add(new Vertex(value.X, value.Y, value.Z,
                                         vertexId++));
            v += dv;
        }
        value = GetValue(u, 1);
        mesh.Vertices.Add(new Vertex(value.X, value.Y, value.Z, vertexId++));
        u += du;
    }
    v = 0;
    for (int j = 0; j < vCount; ++j)
    {
```

```
            value = GetValue(1, v);
            mesh.Vertices.Add(new Vertex(value.X, value.Y, value.Z, vertexId++));
            v += dv;
        }
        value = GetValue(1, 1);
        mesh.Vertices.Add(new Vertex(value.X, value.Y, value.Z, vertexId));

        for (int i = 0; i < uCount; ++i)
        {
            for (int j = 0; j < vCount; ++j)
            {
                mesh.Polygons.Add(new Polygon(mesh.Vertices[i * (vCount + 1) + j],
                                              mesh.Vertices[i*(vCount+1) + j + 1],
                                              mesh.Vertices[(i+1)*(vCount+1) + j],
                                              material));
                mesh.Polygons.Add(new Polygon(mesh.Vertices[(i+1)*(vCount + 1) + j],
                                              mesh.Vertices[i*(vCount + 1) + j + 1],
                                              mesh.Vertices[(i+1)*(vCount+1)+j+1],
                                              material));
            }
        }
        mesh.CalculateVerticesNormals();
    }
```

Преобразование точек в экранные координаты и поиск точки

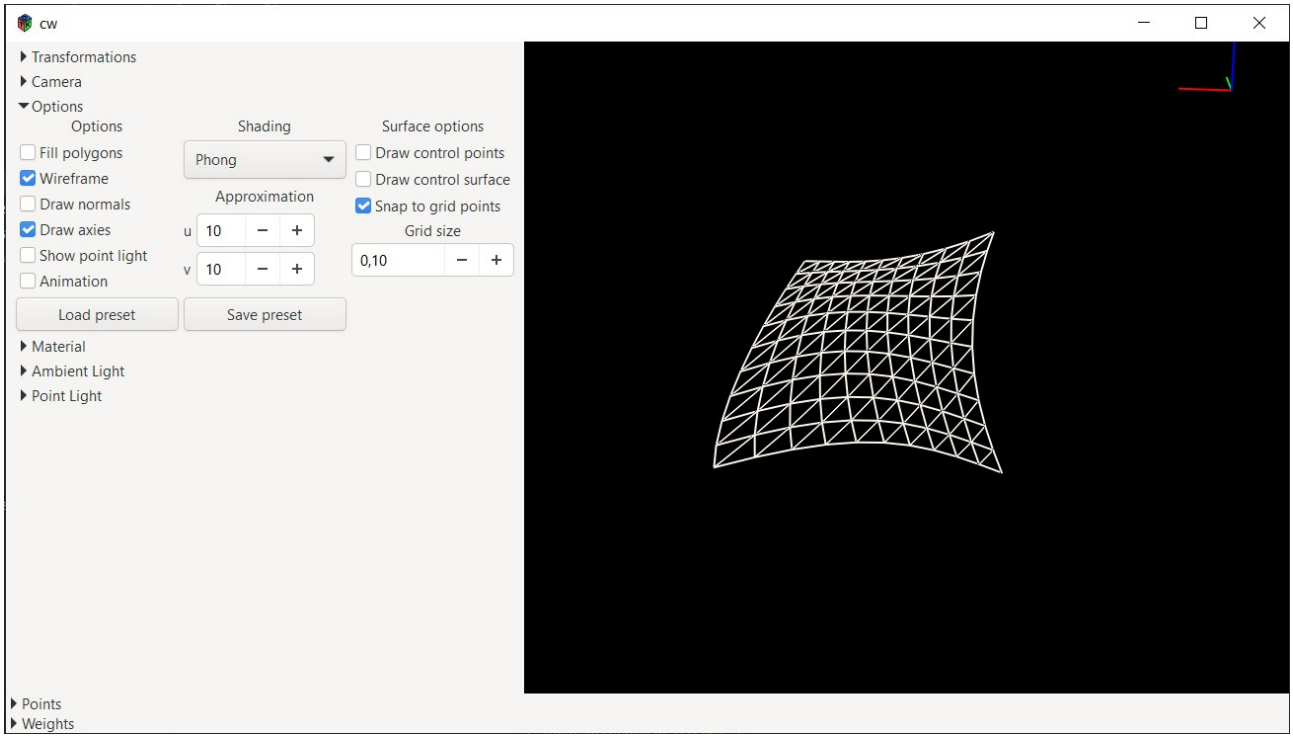
```
public void CalculateClipSpacePoints(Matrix4x4 viewMatrix, Matrix4x4 projMatrix)
{
    Matrix4x4 transformMatrix = Matrix4x4.Transpose(projMatrix * viewMatrix);
    for (int i = 0; i < 4; ++i)
    {
        for (int j = 0; j < 4; ++j)
        {
            Vector4 point = Vector4.Transform(new Vector4(Points[i, j], 1),
                                              transformMatrix);
            ClipSpacePoints[i * 4 + j] = new Vector3(point.X / point.W,
                                                     point.Y / point.W,
                                                     point.Z / point.W);
        }
    }
}

public int FindPoint(Vector2 point)
{
    int id = -1;
    for (int i = 0; i < 16; ++i)
    {
        double length = Math.Sqrt(Math.Pow(ClipSpacePoints[i].X - point.X, 2) +
                                  Math.Pow(ClipSpacePoints[i].Y - point.Y, 2));
        if (length < 3e-2 && Math.Abs(ClipSpacePoints[i].Z) <= 1 &&
            (id < 0 || ClipSpacePoints[i].Z < ClipSpacePoints[id].Z))
        {
            id = i;
        }
    }

    return id;
}
```

Шейдер рисования координатных осей

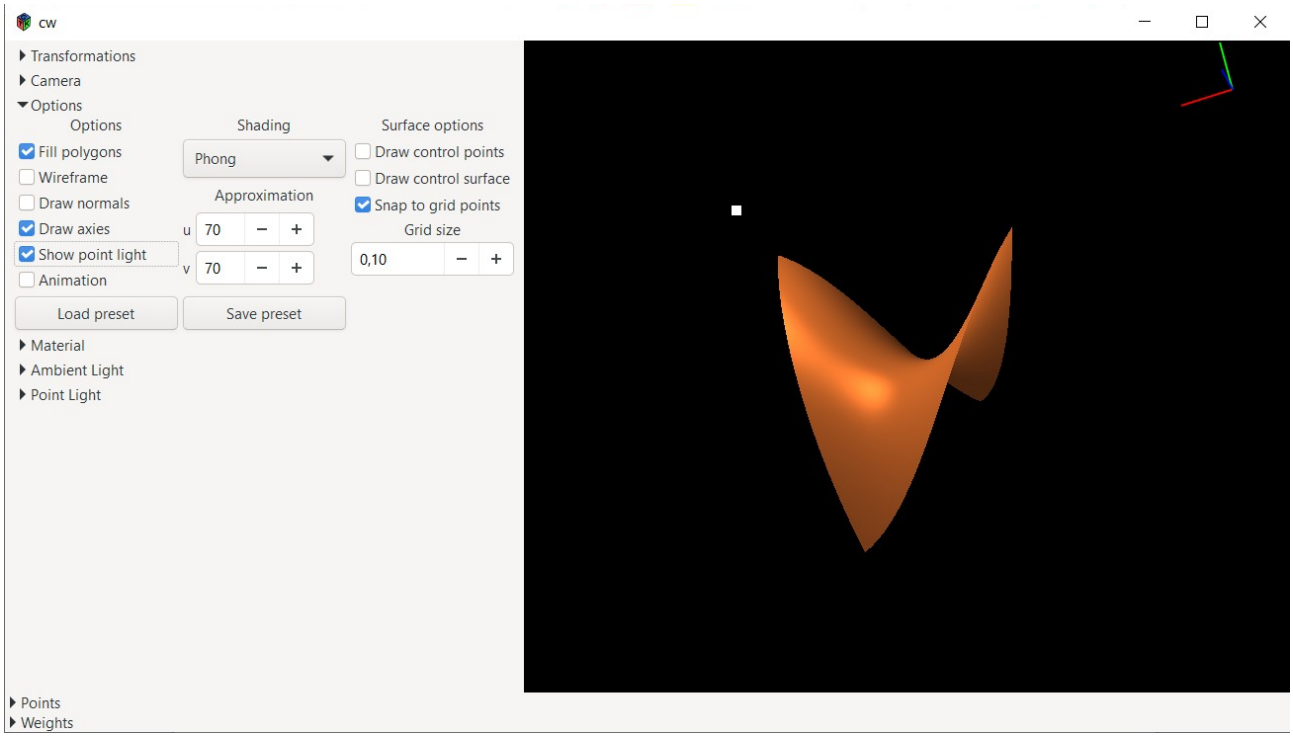
```
#version 330

layout (location = 0) in vec3 position;
layout (location = 1) in vec3 inColor;

out vec3 color;

uniform mat4 view;
uniform float corner;

void main() {
    color = inColor;
    gl_Position = view * vec4(position / 6.0, 0.0);
    gl_Position.w = 1.0;
    gl_Position.z = 0.0;
    gl_Position += vec4(corner, corner, 0.0, 0.0);
}
```

Демонстрация работы программы

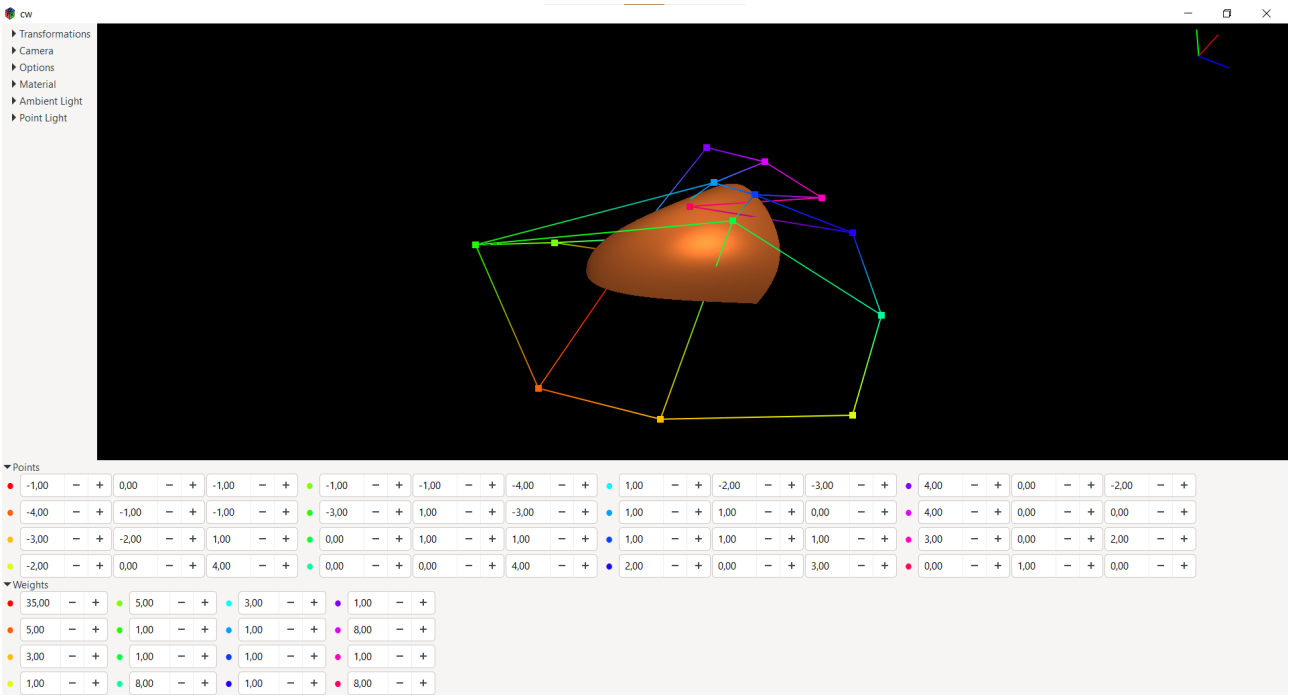
Каркасная визуализация поверхности



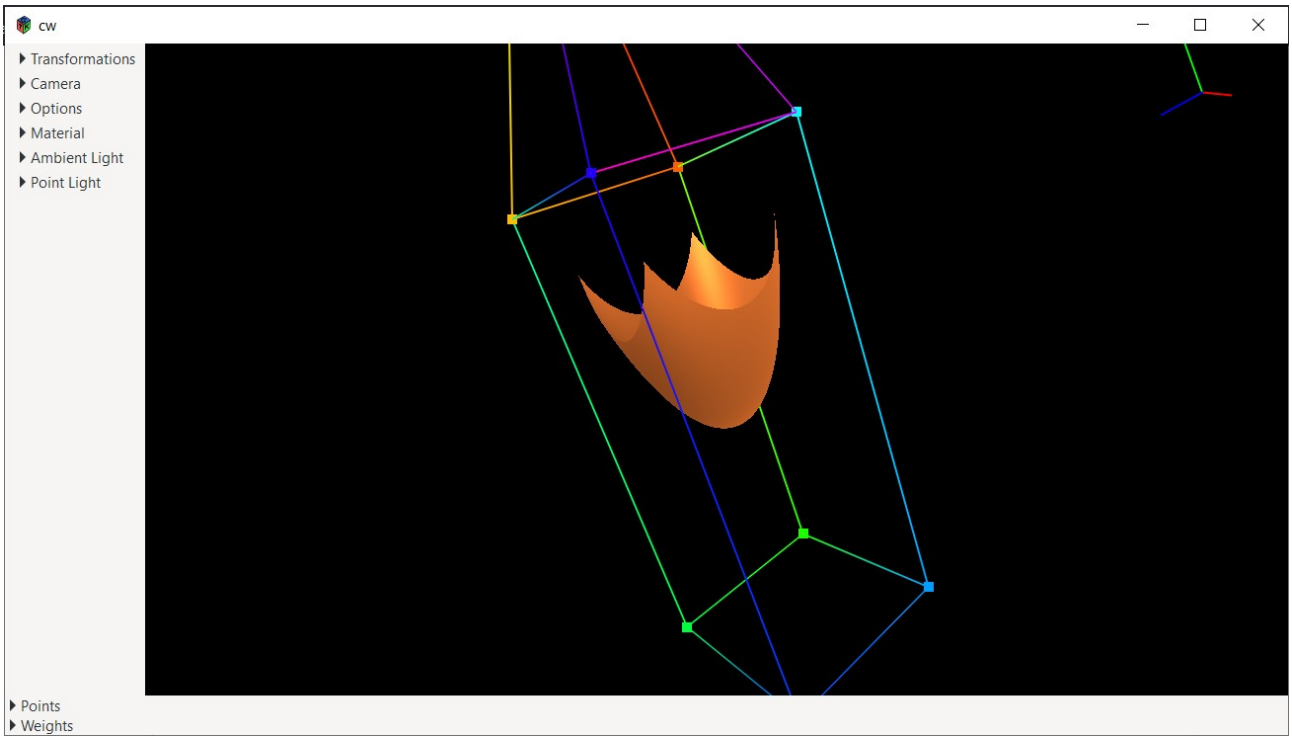
Поверхность, освещенная один источником. Используется модель затенения Фонга

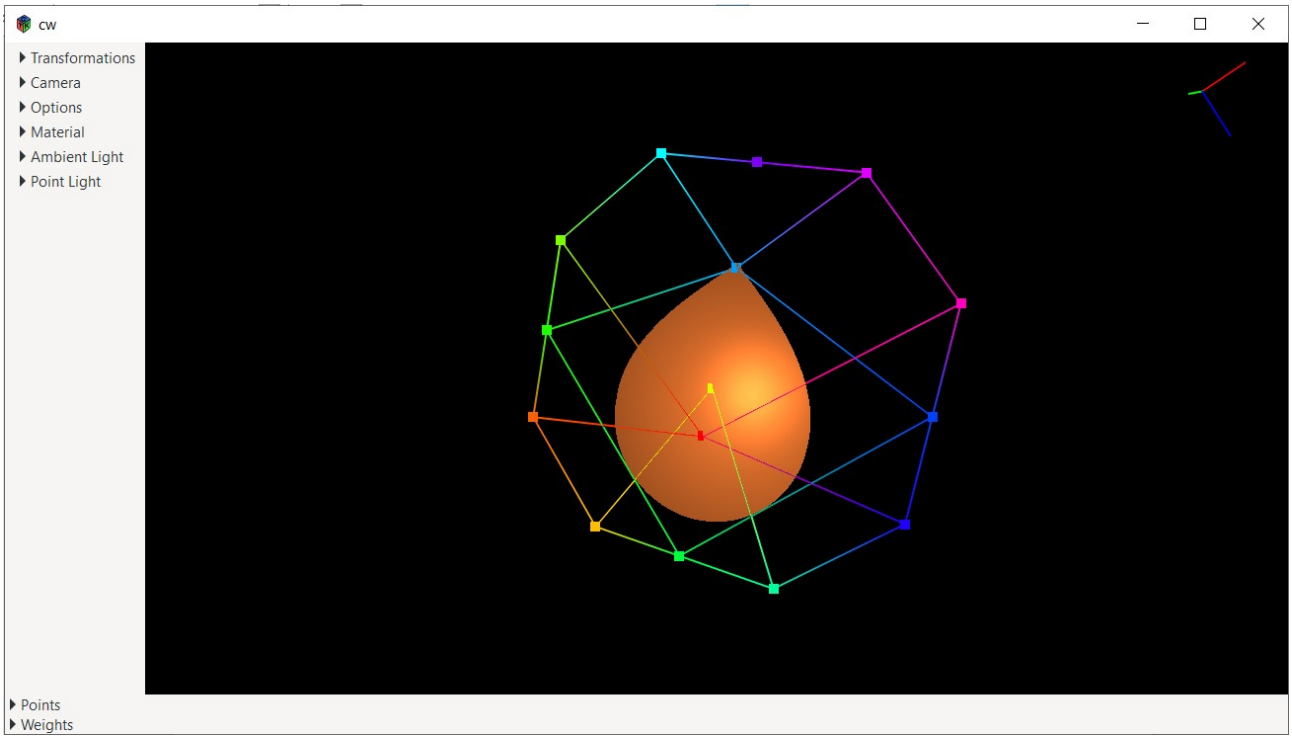


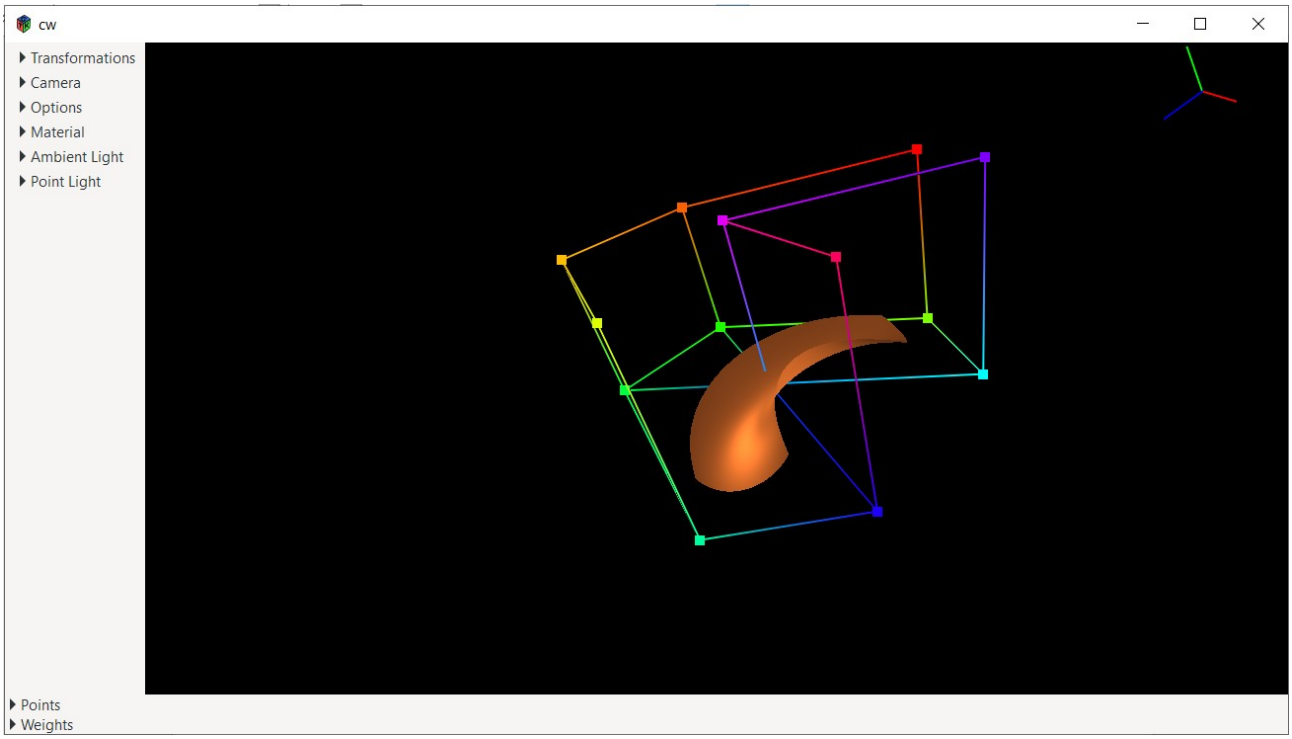
Визуализация опорных точек и опорной плоскости



Некоторые готовые тестовые примеры




cw
Transformations
Camera
Options
Material
Ambient Light
Point Light
Points
Weights


cw
Transformations
Camera
Options
Material
Ambient Light
Point Light
Points
Weights

Используемая литература

1. *Mono Documentation* – URL: http://docs.go-mono.com
2. *GTK Documentation* – URL: https://docs.gtk.org
3. Шикин Е. В., Плис Л. И. *Кривые и поверхности на экране компьютера. Руководство по сплайнам для пользователей.* – М.: ДИАЛОГ-МИФИ, 1996. - 240с.