**Министерство науки и высшего образования Российской Федерации**
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**
**(Университет ИТМО)**

**Факультет программной инженерии и компьютерной техники**

Наименование дисциплины: **Алгоритмы Компьютерной Графики**

# О Т Ч Е Т

по лабораторной работе 1

Выполнил: **Маргиев Давид, P3369**

Преподаватель: **Андреев Артем Станиславович**

Дата: 13.10.2025

г. Санкт-Петербург
2025

# Задание 1:

Взяв любые примеры – OpenGL GLSL, HLSL пример отрисовки, преобразовать вывод звездочки с 5 лучами на иное число лучей – 4, 6.
Поменять цвет фона, заданный в исполняемом файле. Реализовать вместо triangle fan : triangle list и triangle strip.

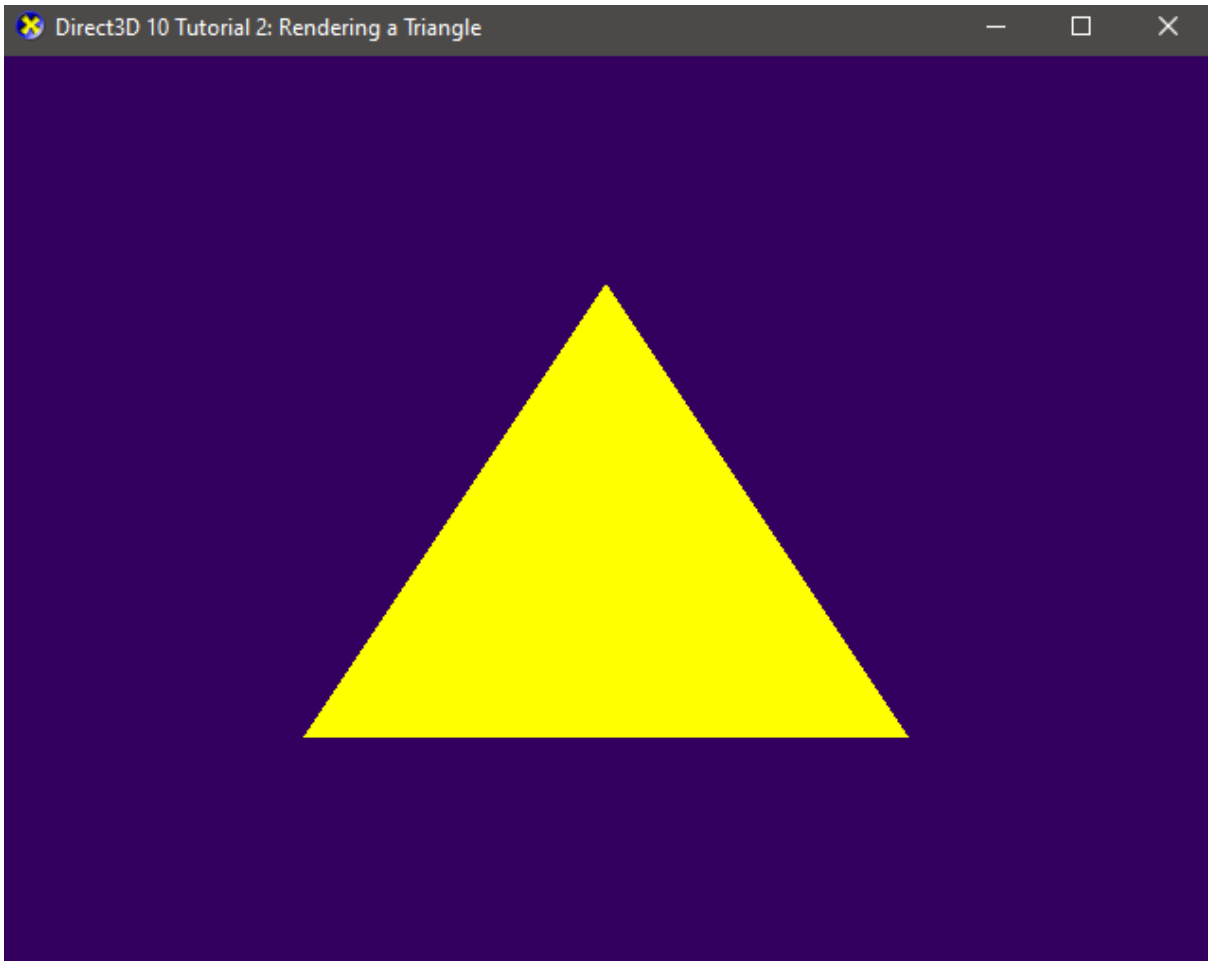
**Выполнение:**

Я решил взять пример из Sample Browser, который предоставляется с DirectX SDK 2010, при этом перейдя с топологии TriangleFan, на TriangleList.
Пример упрощенный, с треугольником, но я не нашел другой, который мог бы подойти (отрисовка треугольника и звезды работают по одному алгоритму: треугольник - звезда с тремя вершинами, поэтому считаю, что этот сэмпл можно использовать)

Сначала, я изменил цвет фона в функции Render():

```
float ClearColor[4] = { 0.2f, 0.0f, 0.375f, 1.0f }; // red,green,blue,alpha
```

Результат - цвет стал фиолетовым после изменения значений красного и синего цветов:



Затем я подключил модули cmath и модуль вектора для дальнейшей работы:

```
#include <vector>
#include <cmath>
```

А также добавил переменную g_VertexCount типа UINT:

```
// … Другие глобальные переменные
ID3D10InputLayout*      g_pVertexLayout = NULL;
ID3D10Buffer*           g_pVertexBuffer = NULL;
UINT                    g_VertexCount = 0;
```

Далее, в функции InitDevice() я поменял алгоритм создания VertexBuffer на следующий:

```
    // Create vertex buffer
    // Параметры звезды
    int numRays = 5;
    float rOuter = 0.5f;   // Внешний радиус
    float rInner = 0.2f;   // Внутренний радиус

    std::vector<SimpleVertex> starVertices;

    float step = 3.1415926535f / numRays; // Шаг угла

    for (int i = 0; i < numRays * 2; i++)
    {
        float angle1 = i * step;
        float angle2 = (i + 1) * step;

        float r1 = (i % 2 == 0) ? rOuter : rInner;
        float r2 = ((i + 1) % 2 == 0) ? rOuter : rInner;

        SimpleVertex vCenter;
        vCenter.Pos = D3DXVECTOR3(0.0f, 0.0f, 0.5f);

        SimpleVertex v1;
        v1.Pos = D3DXVECTOR3(r1 * sin(angle1), r1 * cos(angle1),
0.5f);

        SimpleVertex v2;
```

```
        v2.Pos = D3DXVECTOR3(r2 * sin(angle2), r2 * cos(angle2),
0.5f);

        starVertices.push_back(vCenter);
        starVertices.push_back(v2);
        starVertices.push_back(v1);
    }

    g_VertexCount = (UINT)starVertices.size();

    D3D10_BUFFER_DESC bd;
    bd.Usage = D3D10_USAGE_DEFAULT;
    bd.ByteWidth = sizeof(SimpleVertex) * g_VertexCount;
    bd.BindFlags = D3D10_BIND_VERTEX_BUFFER;
    bd.CPUAccessFlags = 0;
    bd.MiscFlags = 0;

    D3D10_SUBRESOURCE_DATA InitData;
    InitData.pSysMem = &starVertices[0];

    hr = g_pd3dDevice->CreateBuffer(&bd, &InitData,
&g_pVertexBuffer);
    if (FAILED(hr))
        return hr;

    UINT stride = sizeof(SimpleVertex);
    UINT offset = 0;
    g_pd3dDevice->IASetVertexBuffers(0, 1, &g_pVertexBuffer,
&stride, &offset);

    g_pd3dDevice-
>IASetPrimitiveTopology(D3D10_PRIMITIVE_TOPOLOGY_TRIANGLELIST);
```

Затем, мы меняем цикл отрисовки в функции Render():

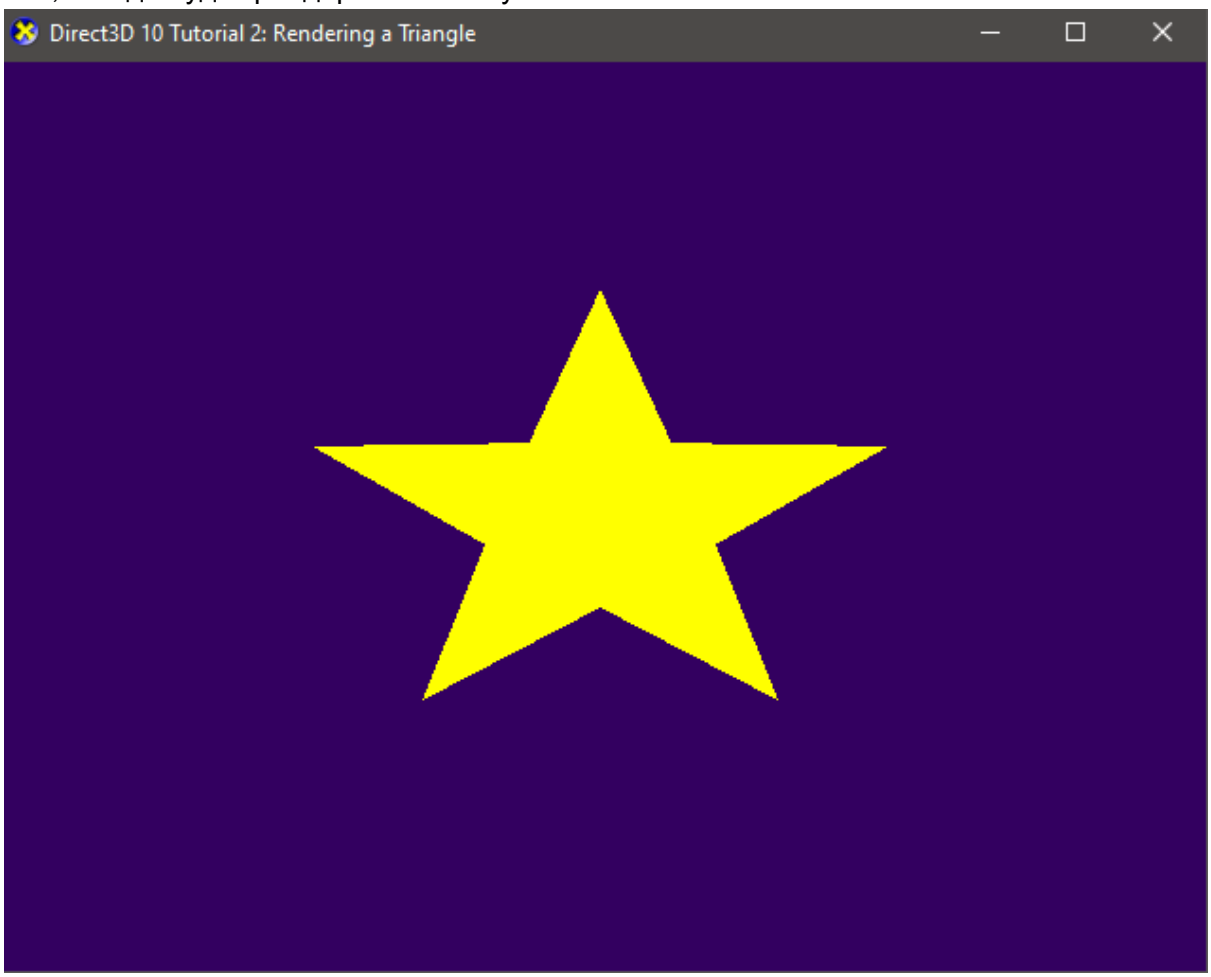
```
    for( UINT p = 0; p < techDesc.Passes; ++p )
    {
        g_pTechnique->GetPassByIndex( p )->Apply( 0 );
        g_pd3dDevice->Draw( g_VertexCount, 0 );
    }
```

И теперь он использует глобальную переменную, которую мы инициализировали ранее.

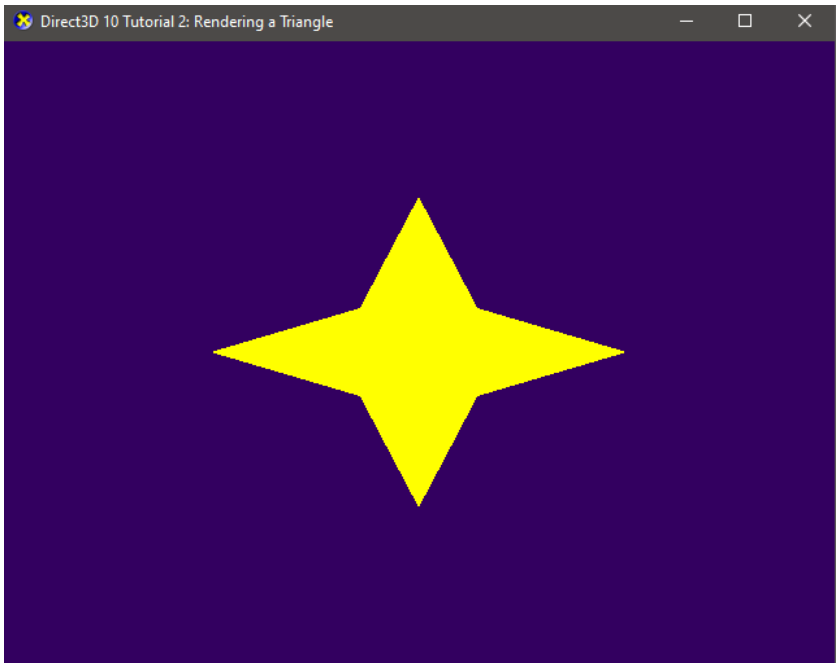
Мы можем поставить количество лучей в куске кода, представленном выше, внутри функции InitDevice():

```
int numRays = 5;
```

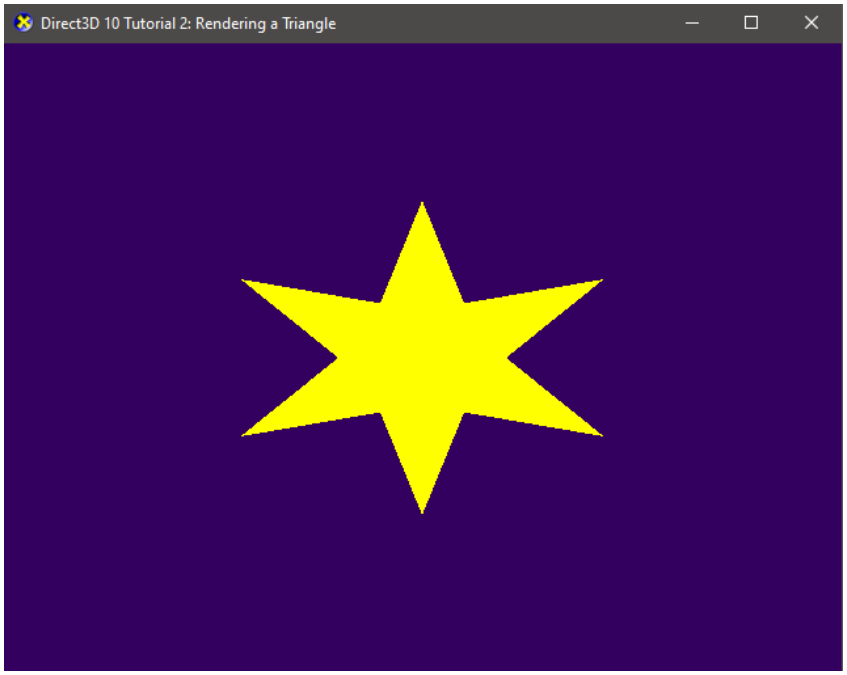
Так, звезда будет рендериться с 5 лучами:



Изменим numRays на 4:




И на 6:



Теперь попробуем использовать Triangle Strip:

Для этого в InitDevice() обозначим:

```
g_pd3dDevice->IASetPrimitiveTopology(D3D10_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP);
```

(До этого мы использовали TriangleList).

Результат (с 6 звездами):


Такой результат получился, потому что геометрия звезды строится от центра (как с TriangleFan). Массив вершин я подготовил для списка отдельных треугольников (List), где центр дублируется.
Режим TriangleStrip пытается соединить эти дубликаты в единую ленту, создавая паразитные треугольники-перемычки между лучами и меняя направление обхода вершин, из-за чего часть геометрии отсекается (backface culling).

**Таким образом, я отрисовал звезды с помощью режимов TriangleList и TriangleStrip. Можно сделать вывод, что второй режим не подходит для рендеринга звезд, и лучше использовать TriangleList.**

**Исходный код измененного проекта можно посмотреть тут:**
**https://github.com/seventwogth/graph-course**