Geometria Intrínseca de Modelos 3D

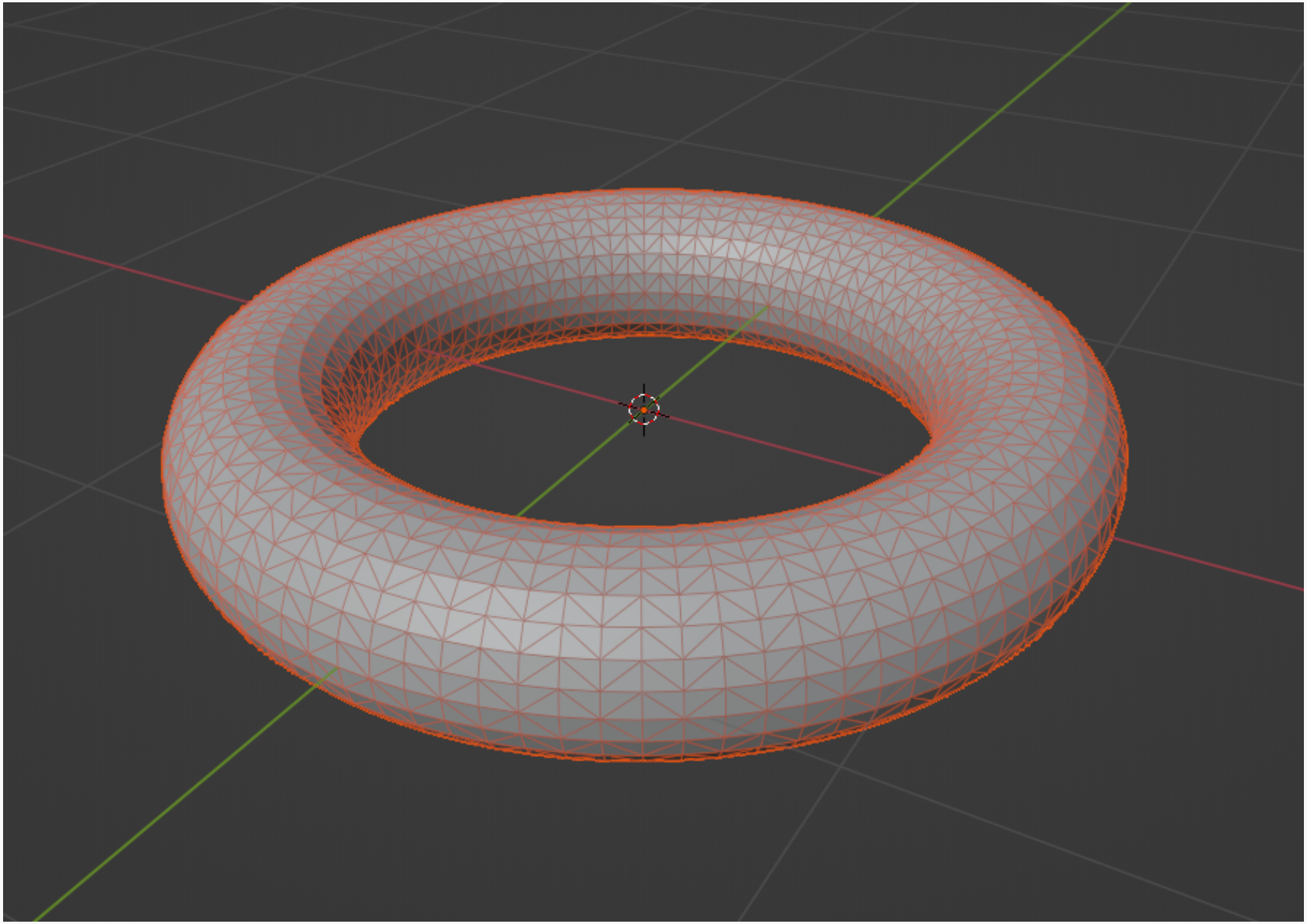
Eduardo Renesto

GD2 UFABC 2022.3

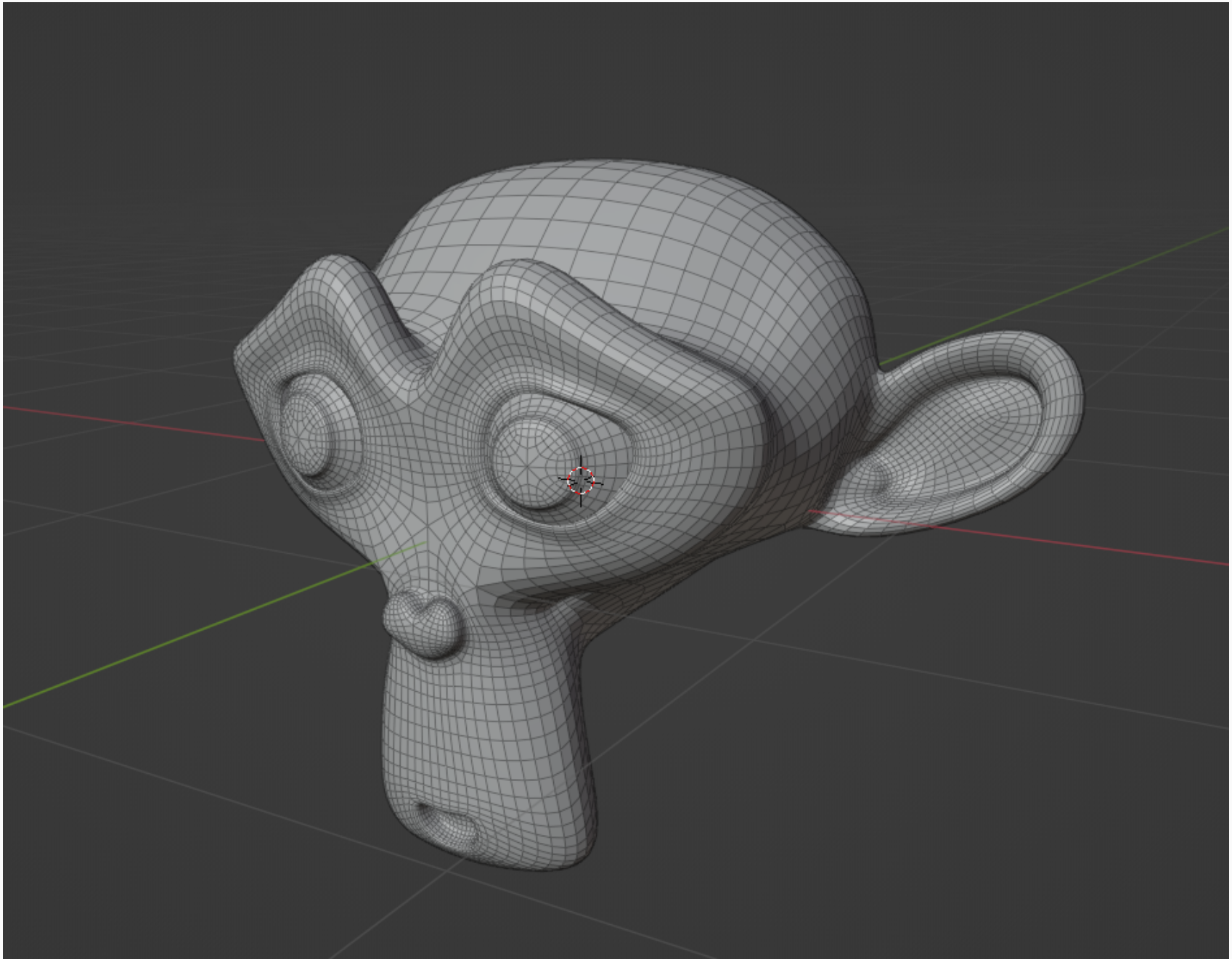
Modelo 3D

- Aproximação de uma superfície por malha triangular
- Representação prática e comum para modelos 3D em CG

Modelo 3D - Toro



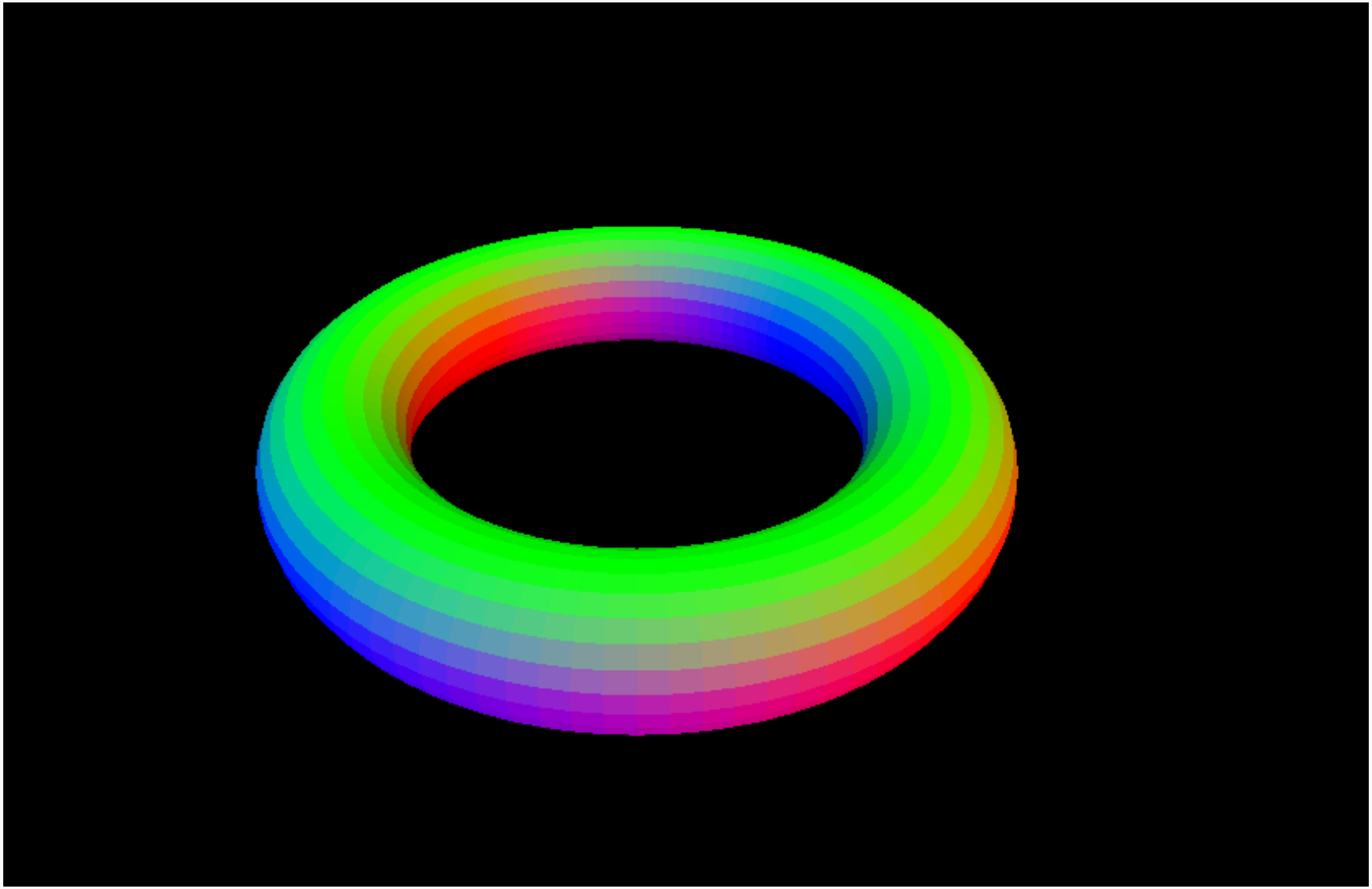
Modelo 3D - Suzanne



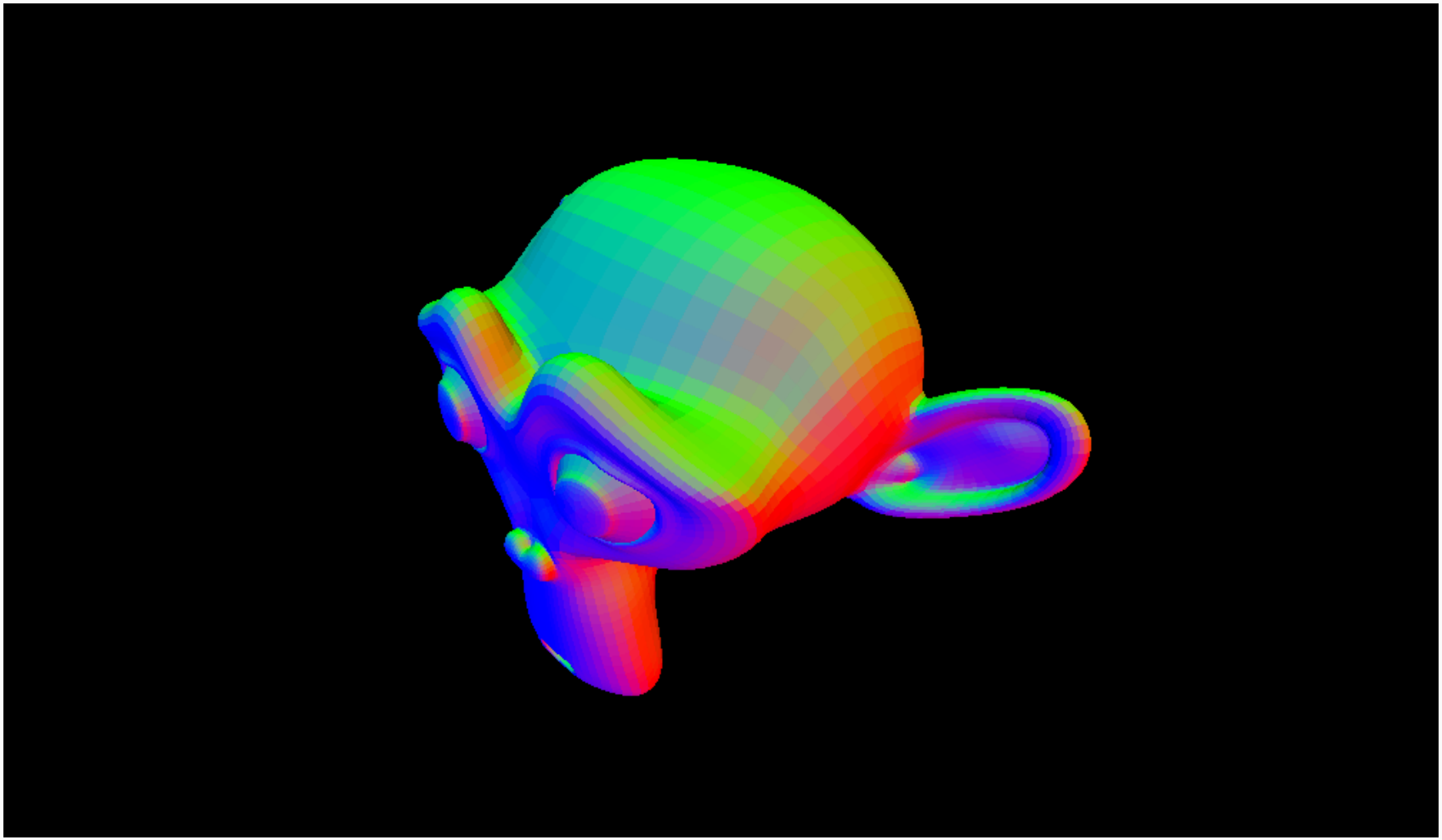
O que quero?

- Calcular parâmetros da geometria intrínseca da superfície representada pelo modelo
  - Curvatura Gaussiana
  - Curvatura Normal
  - Primeira Forma Fundamental

Ideia inicial - vetores normais



Ideia inicial - vetores normais



Ideia inicial - GLSL

```
in vec3 out_position;
in vec3 out_normal;

out vec4 frag_color;

void main() {
    frag_color = vec4(out_normal, 1.0);
}
```

dFdx

GLSL 4
GLSL ES 3

dFdx, dFdy — return the partial derivative of an argument with respect to x or y

Declaration

```
genType dFdx(genType p);
genType dFdy(genType p);
genType dFdxCoarse(genType p);
genType dFdyCoarse(genType p);
genType dFdxFine(genType p);
genType dFdyFine(genType p);
```

Parameters

p

Specifies the expression of which to take the partial derivative.

Description

Available only in the fragment shader, these functions return the partial derivative of expression *p* with respect to the window x coordinate (for `dFdx*`) and y coordinate (for `dFdy*`).

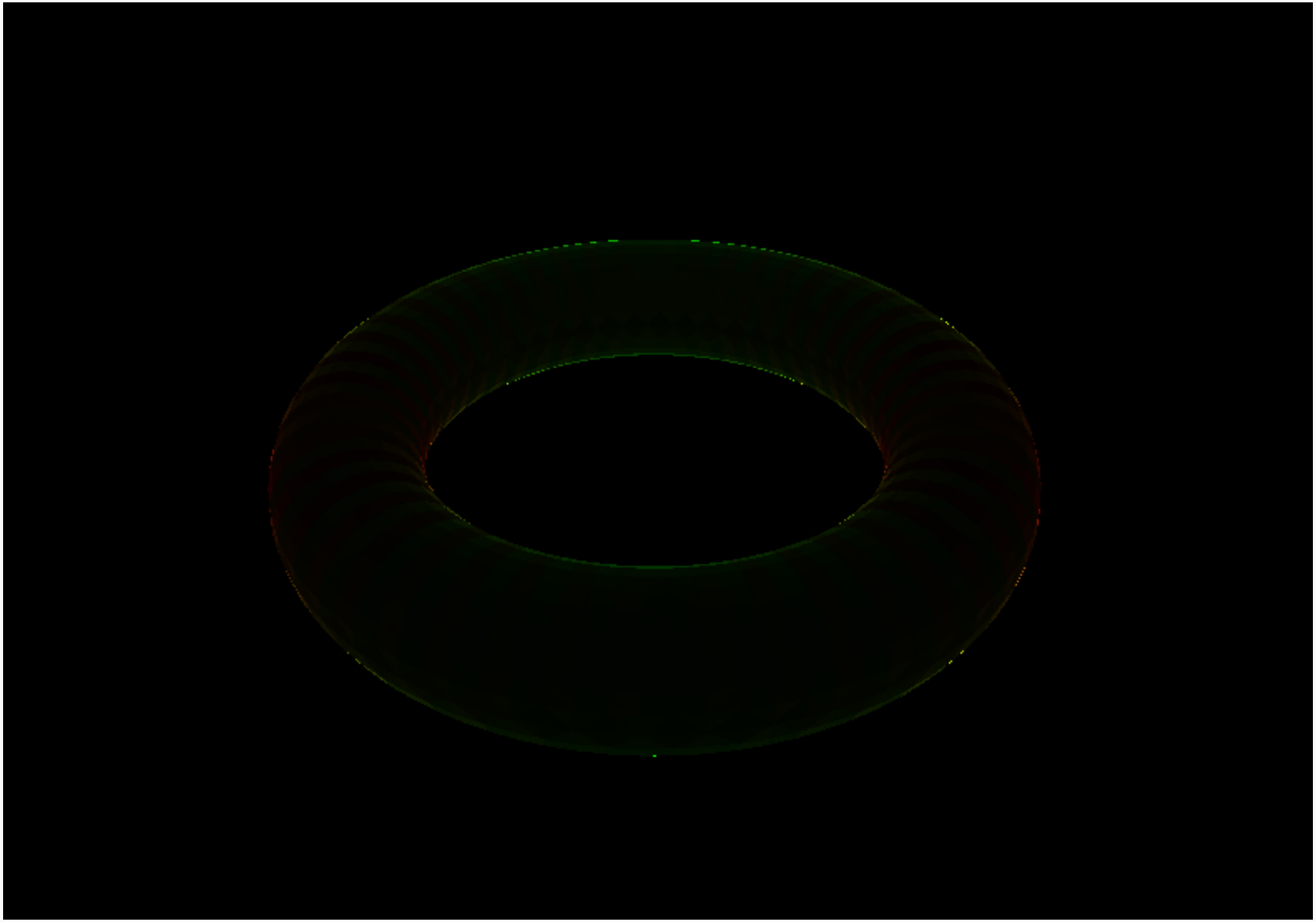
Ideia inicial - GLSL

```
void main() {
    vec3 u = abs(dFdxFine(out_normal));
    vec3 v = abs(dFdyFine(out_normal));

    float lu = length(u);
    float lv = length(v);

    frag_color = vec4(lu, lv, 0.0, 1.0);
}
```

Ideia inicial - GLSL

Inicialmente deu tela preta. . . ..

Ideia inicial - GLSL



Segunda ideia - aproximação

- Inspirado em (1)
- Se sabemos a cara da parametrização, fica fácil...

Ideia - ajustar um paraboloide

$$f(u, v) = \frac{1}{2}\left(au^2 + 2buv + cv^2\right)$$

Ideia - ajustar um paraboloide

Se eu descobrir a, b, c, tenho...

$$S = -\begin{pmatrix} a & b \\ b & c \end{pmatrix} \tag{1}$$

$$K = \det S \tag{2}$$

$$H = \operatorname{tr} S \tag{3}$$

Como?

- Para cada vértice p
- Descobre base de T_pS e completa pra base de $\mathbb{R}^3$ com a normal
- Encontra coordenadas de cada vértice vizinho nessa base
- Pro vizinho p_i, se $p_i = u_i e_1 + v_i e_2 + n_i N$, coloca $f(u_i, v_i) = n_i$
- Resolve por Quadrados Mínimos!

Como?

$$U = \begin{pmatrix} \frac{u_1^2}{2} & u_1 v_1 & \frac{v_1^2}{2} \\ \frac{u_2^2}{2} & u_2 v_2 & \frac{v_2^2}{2} \\ \frac{u_3^2}{2} & u_3 v_3 & \frac{v_3^2}{2} \end{pmatrix}$$

$$X = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$F = \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix}$$

Como?

$$UX = F$$

$$X = U^{-1}F$$

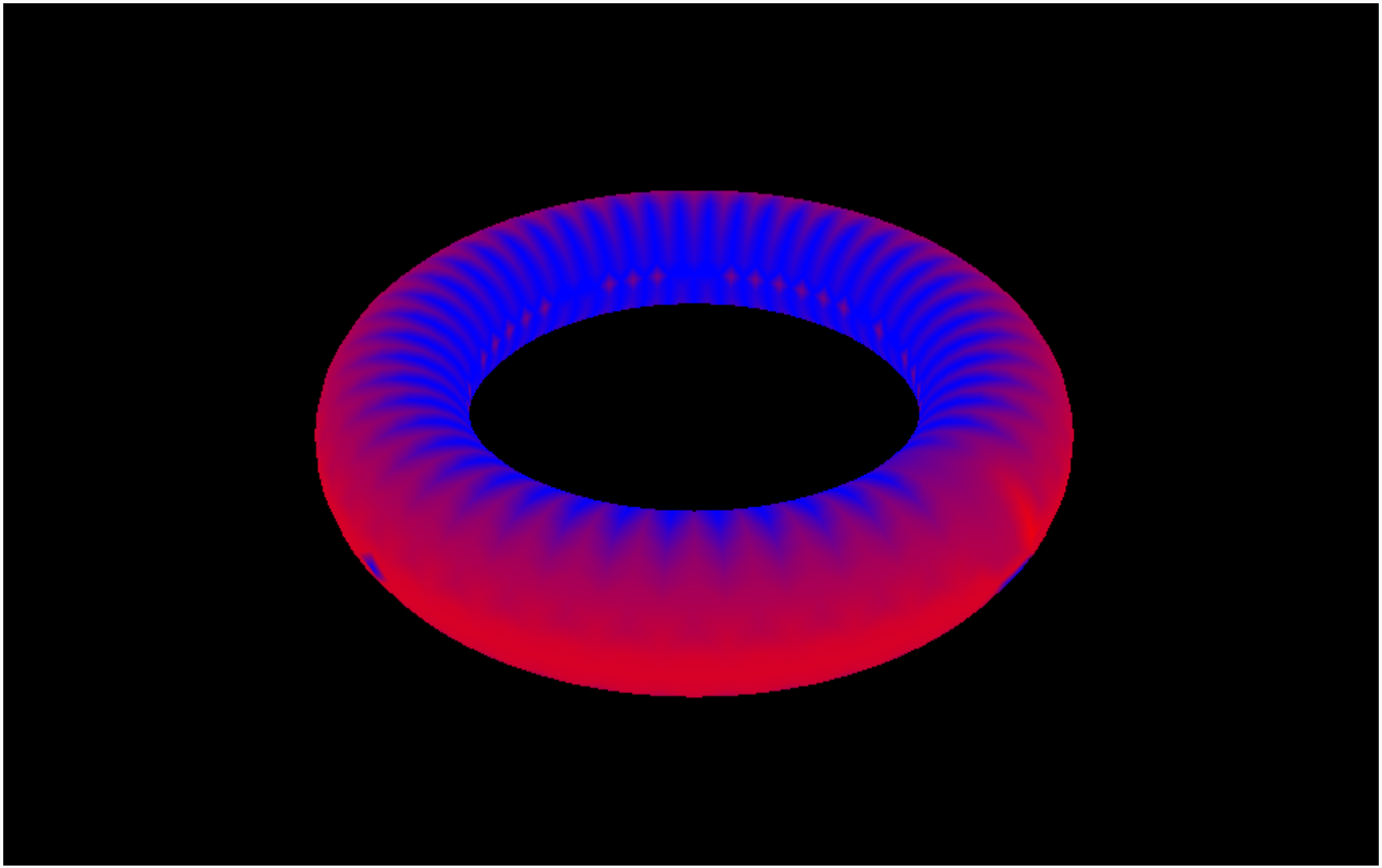
Antes disso

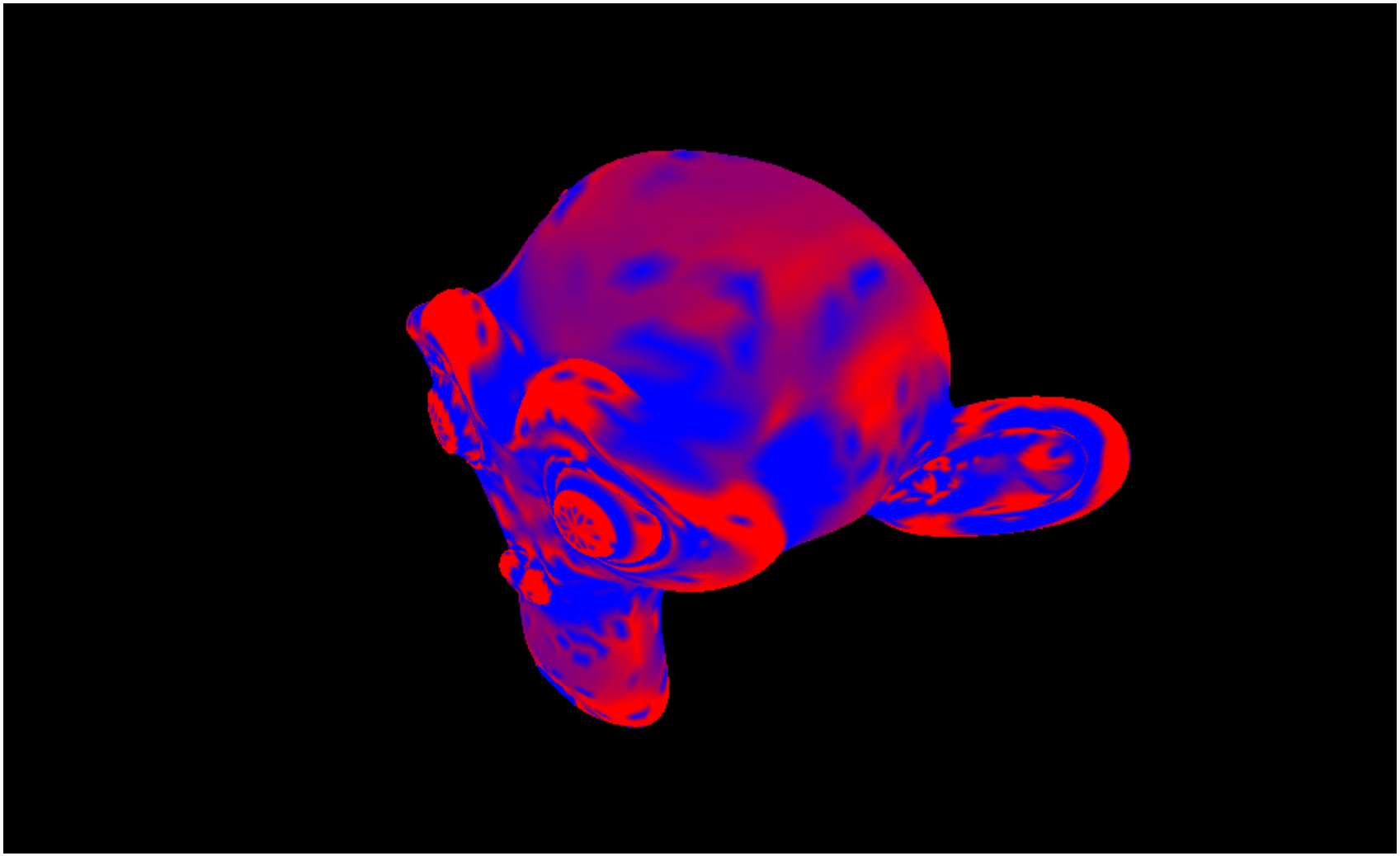
Como encontrar cada T_pS?

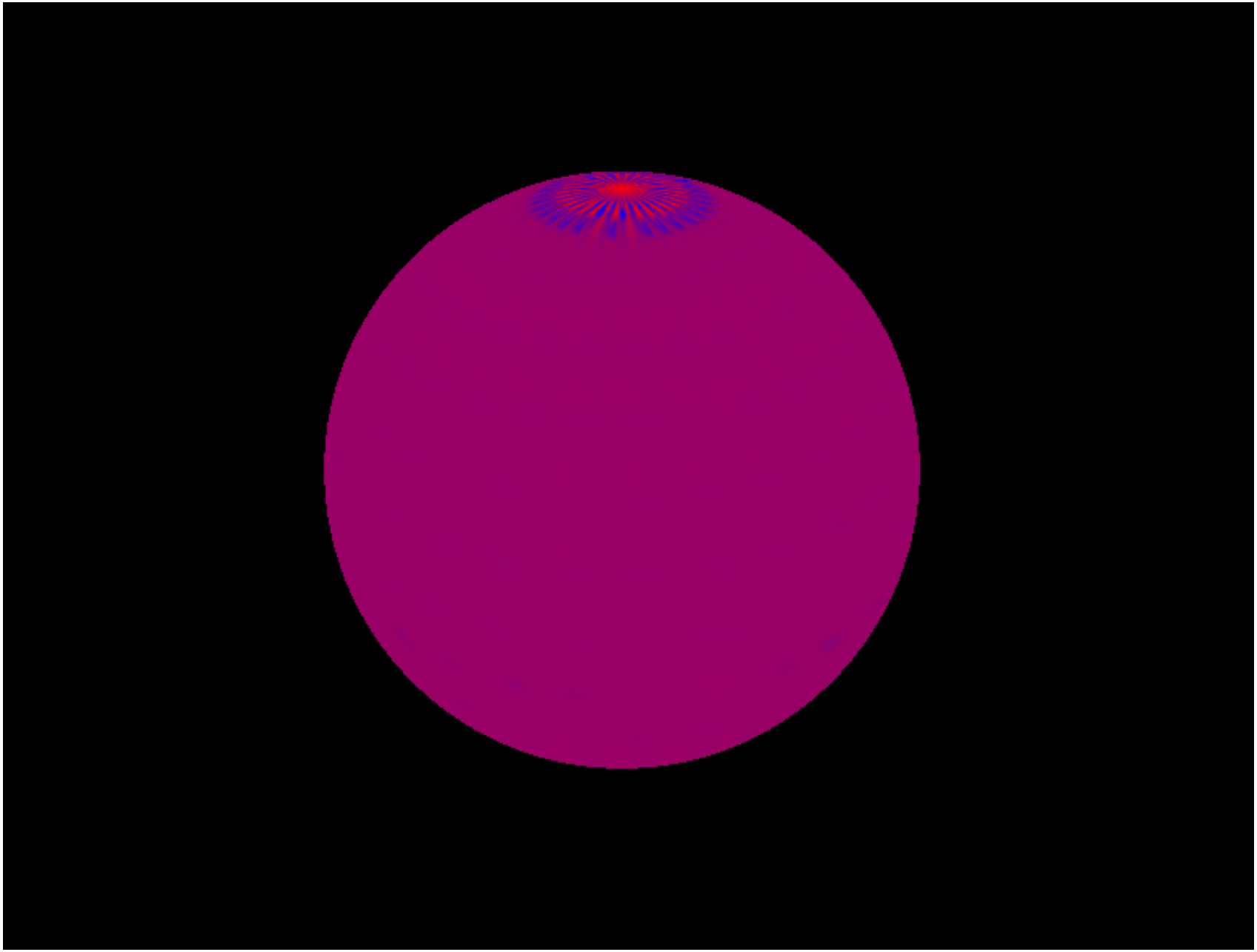
- Encontrar vizinhanças
- Calcular normais médias
- Para cada ponto p
  - Escolher um vetor qualquer a' que não seja paralelo ao T_pS
  - Projetar a' em T_pS e normalizar a projeção para conseguir a
  - $b = \frac{a \times N}{|a \times N|}$
  - $\{a, b\}$ é base de T_pS
  - Monta matriz mudança de base da canônica para $\{a, b, N\}$

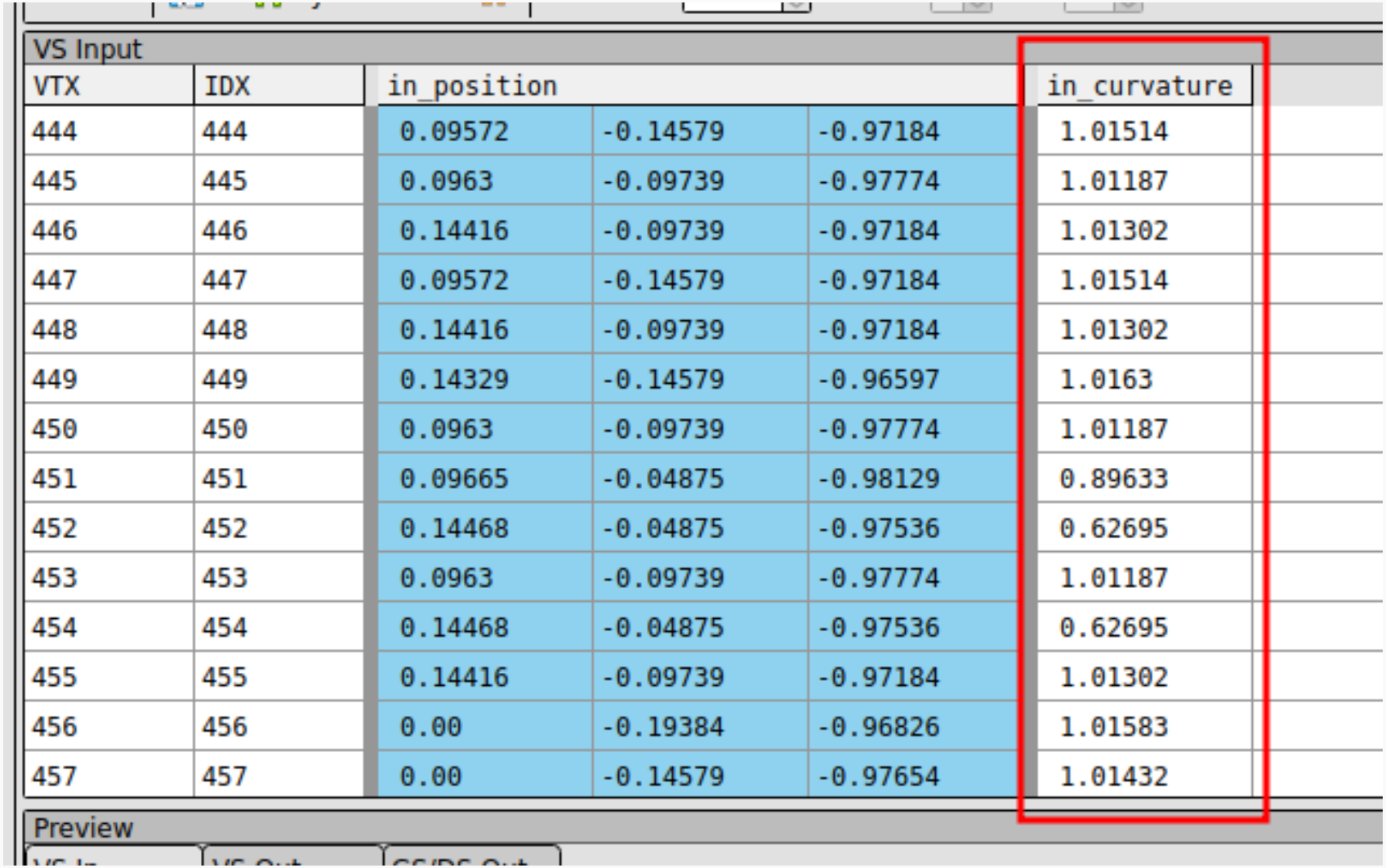
Depois. . .

- Para cada ponto p, escolhe os vizinhos p_1, p_2, p_3
- Encontra as coordenadas na base $\{a, b, N\}$
- Completa as matrizes
- Faz contas
- ????
- Profit





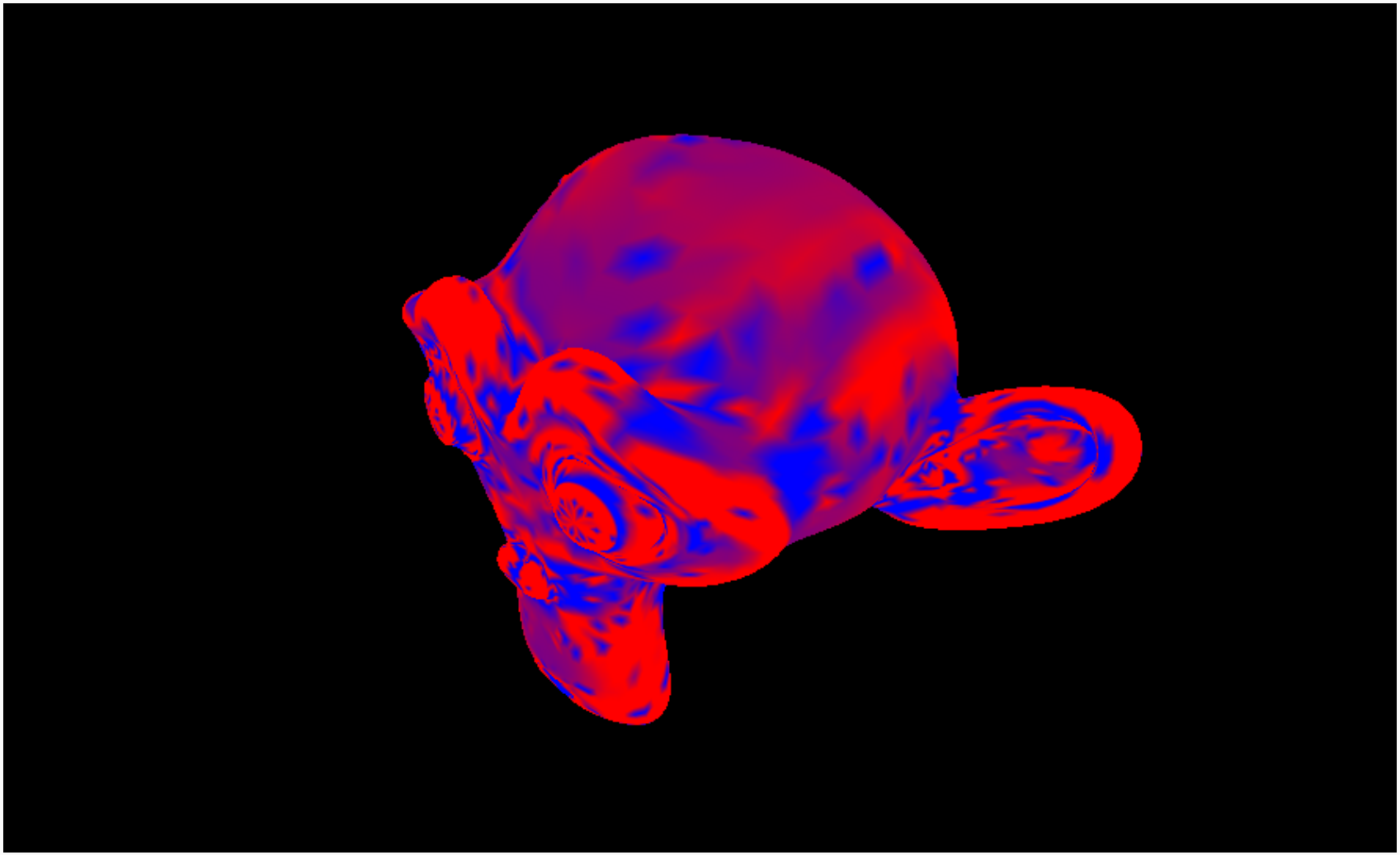




VS Input

VTX	IDX	in_position			in_curvature
444	444	0.09572	-0.14579	-0.97184	1.01514
445	445	0.0963	-0.09739	-0.97774	1.01187
446	446	0.14416	-0.09739	-0.97184	1.01302
447	447	0.09572	-0.14579	-0.97184	1.01514
448	448	0.14416	-0.09739	-0.97184	1.01302
449	449	0.14329	-0.14579	-0.96597	1.0163
450	450	0.0963	-0.09739	-0.97774	1.01187
451	451	0.09665	-0.04875	-0.98129	0.89633
452	452	0.14468	-0.04875	-0.97536	0.62695
453	453	0.0963	-0.09739	-0.97774	1.01187
454	454	0.14468	-0.04875	-0.97536	0.62695
455	455	0.14416	-0.09739	-0.97184	1.01302
456	456	0.00	-0.19384	-0.96826	1.01583
457	457	0.00	-0.14579	-0.97654	1.01432

Preview



Documentação live!

https://edurenesto.github.io/ufabc-gd2-gauss/gauss/geom/index.html

Open source

https://github.com/EduRenesto/ufabc-gd2-gauss

Referências

[1] Bærentzen, J.A., Gravesen, J., Anton, F., Aanæs, H. (2012). Curvature in Triangle Meshes. In: Guide to Computational Geometry Processing. Springer, London. https://doi.org/10.1007/978-1-4471-4075-7_8