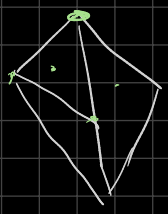
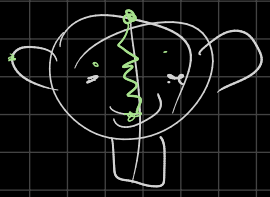
Commence : modèle de base — modèle d'origine



A'



A

But : créer les patchs → associés $v \in A$ à un $f \in A'$ (vertex → face)

Patch P = Hashmap → $K = f \in A'$

→ $V = \text{list}(v \in A)$ (indices des vertices dans model A)

Construction : Soit avec la création du mesh basse
Soit après avec l'utilisation des bords (plus court chemin)

Projection des patchs sur le maillage de base (en considérant on sait sur quelle face est le point à projeter)

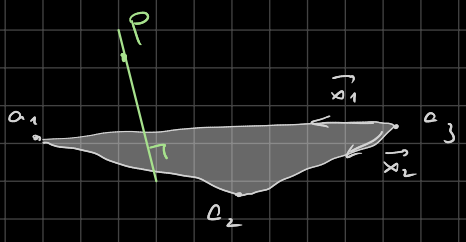
Modèle $M^{(0)}$ ⟶ vertices = Matrice de projection orthogonale

$$\begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = \begin{bmatrix} \text{Proj} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad \begin{matrix} \rightarrow v' \in A' \\ \rightarrow v \in A \end{matrix}$$

!?

associé à vertice ⇒ distance à $v \in A$

Demander à Alice ♡



Soit $p = [x \ y \ z]^T \in \mathbb{R}^3$ et $(a_1, a_2, a_3) \in \mathbb{R}^3$ où le plan défini par $\vec{x_1}$ et $\vec{x_2}$ tq

$x_1 = a_1 - a_3$ et $x_2 = a_2 - a_3$

alors on peut décrire le point p' la projection orthogonale de p comme étant :

$$p' = \underbrace{(p \cdot x_1)}_{\in \mathbb{R}} \vec{x_1} + (p \cdot x_2) \vec{x_2} \quad \in \mathbb{R}^3$$

avec donc =

⚠ en fait on ne garde que : $p' = \begin{pmatrix} p \cdot x_1 \\ p \cdot x_2 \end{pmatrix} \in \mathbb{R}^2$

$$= \begin{bmatrix} e_1^1 \\ e_2^1 \\ e_3^1 \end{bmatrix} \begin{bmatrix} e_1^1 & e_2^1 & e_3^1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_1^2 \\ e_2^2 \\ e_3^2 \end{bmatrix} \begin{bmatrix} e_1^2 & e_2^2 & e_3^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

symétrique → $$= \left[\begin{pmatrix} e_1^1{}^2 & e_1^1 e_2^1 & e_1^1 e_3^1 \\ e_2^1 e_1^1 & e_2^1{}^2 & e_2^1 e_3^1 \\ e_3^1 e_1^1 & e_2^1 e_3^1 & e_3^1{}^2 \end{pmatrix} + \begin{pmatrix} e_1^2{}^2 & e_1^2 e_2^2 & e_1^2 e_3^2 \\ e_2^2 e_1^2 & e_2^2{}^2 & e_2^2 e_3^2 \\ e_3^2 e_1^2 & e_3^2 e_2^2 & e_3^2{}^2 \end{pmatrix} \right] \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$= \begin{bmatrix} (x_1-x_3)^2+(x_2-x_3)^2 & (x_1-x_3)(y_1-y_3)+(x_2-x_3)(y_2-y_3) & (x_1-x_3)(z_1-z_3)+(x_2-x_3)(z_2-z_3) \\ (x_1-x_3)(y_1-y_3)+(x_2-x_3)(y_2-y_3) & (y_1-y_3)^2+(y_2-y_3)^2 & (y_1-y_3)(z_1-z_3)+(y_2-y_3)(z_2-z_3) \\ (x_1-x_3)(z_1-z_3)+(x_2-x_3)(z_2-z_3) & (y_1-y_3)(z_1-z_3)+(y_2-y_3)(z_2-z_3) & (z_1-z_3)^2+(z_2-z_3)^2 \end{bmatrix}$$

une fois qu'on a les coordonnées des points p_i', projections de p_i on peut calculer la distance tels que :

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad p' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

$$d = \sqrt{(x'-x)^2 + (y'-y)^2 + (z'-z)^2}$$

à associer à chaque point projeté

c) coordonnées barycentrique : → $(1/2, 1/2, 0)$ etc. comme en 2A en rendu

donné par a, b, c les 3 sommets d'une face

d'où : $\left[\left(\frac{1}{2}a + \frac{1}{2}b, \frac{1}{2}a + \frac{1}{2}c, \frac{1}{2}b + \frac{1}{2}c, \frac{1}{3}a + \frac{1}{3}b + \frac{1}{3}c \right) \right.$

2) subdivision des triangles

initialisation : Liste L → toutes les faces du mesh

1) calcul de $E(f) = \max\left(\frac{E^1(f)}{B(S^1)}, \frac{E^2(f)}{B(S^2)} \right)$ ← face → 2 mesh on utilise ça ?

avec : $E^1(f) = \max(|v_i - p_i|)$ → v_i = sommets dans le patch (les p')

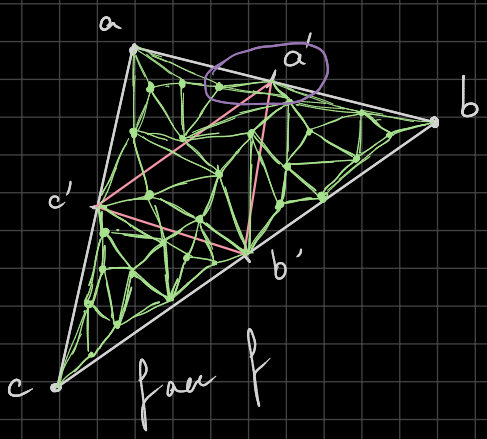
i car on fait pour tous les points du patch → p_i = point de la face pour laquelle la distance est min

$B(S^1)$ = longueur diagonale de la bounding box (permet de normaliser)

↳ on calcule avec les sommets du mesh

→ min (v ∈ S) et max (v ∈ S)

le mesh dérigide (donc pas besoin de recalculer)



si $E(f) < \varepsilon$ → on retire f de L

a) on divise chaque face f en 4 triangles (a', b', c')

on obtient les coordonnées 3D de ces nouveaux points (figure 4) en utilisant les coordonnées barycentriques

⇒ on trouve à quel triangle appartient a' sur le patch

- on trouve les trois sommets (v_1', v_2', v_3')
- on sait qu'ils correspondent à (v_1, v_2, v_3) dans l'input mesh donné:

si $a' = \alpha v_1' + \beta v_2' + \gamma v_3'$ (en 2D)

alors nouv_a' $= \alpha v_1 + \beta v_2 + \gamma v_3$ (en 3D)

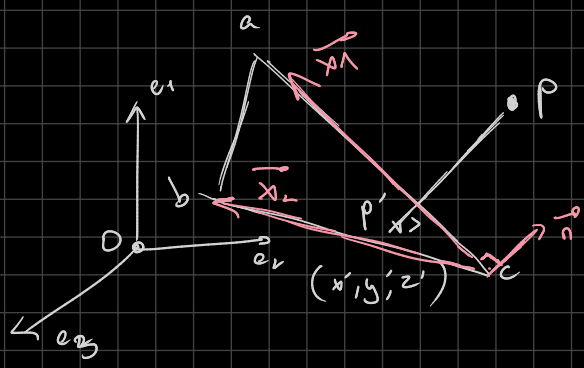
→ pour chaque triangle on résout: $\begin{cases} x' = \alpha x_1' + \beta x_2' + \gamma x_3' \\ y' = \alpha y_1' + \beta y_2' + \gamma y_3' \\ 1 = \alpha + \beta + \gamma \end{cases}$

si $\alpha > 0$ et $\beta > 0$ et $\gamma > 0$
alors on a trouvé le triangle et on peut utiliser ses poids

on supprime la face f et on ajoute les 4 nouvelles

optionnel pour l'instant ← 3) on trouve les T-vertex → si 2 → 4 triangles
→ si 1 → 2 triangles

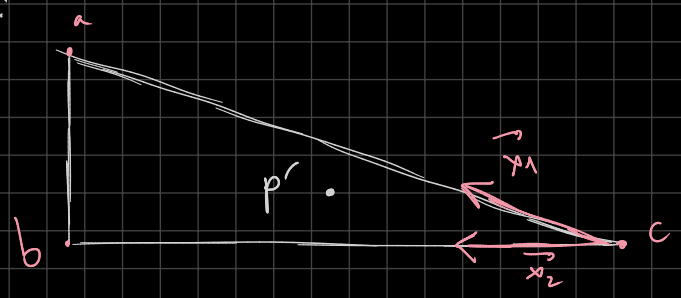
on répète jusqu'à $L = []$



avec $p' = (p \cdot x_1)\vec{x_1} + (p \cdot x_2)\vec{x_2} + 0\vec{n}$

exprimé dans base $(O, \vec{e_1}, \vec{e_2}, \vec{e_3})$

vu d'au dessus:



coordonnée 2D de $p' = \begin{pmatrix} p \cdot x_1 \\ p \cdot x_2 \end{pmatrix}$ avec x_1 x_2 exprimé dans (O, e_1, e_2, e_3)

mais p' ici exprimée dans base $(c, \vec{x_1}, \vec{x_2})$