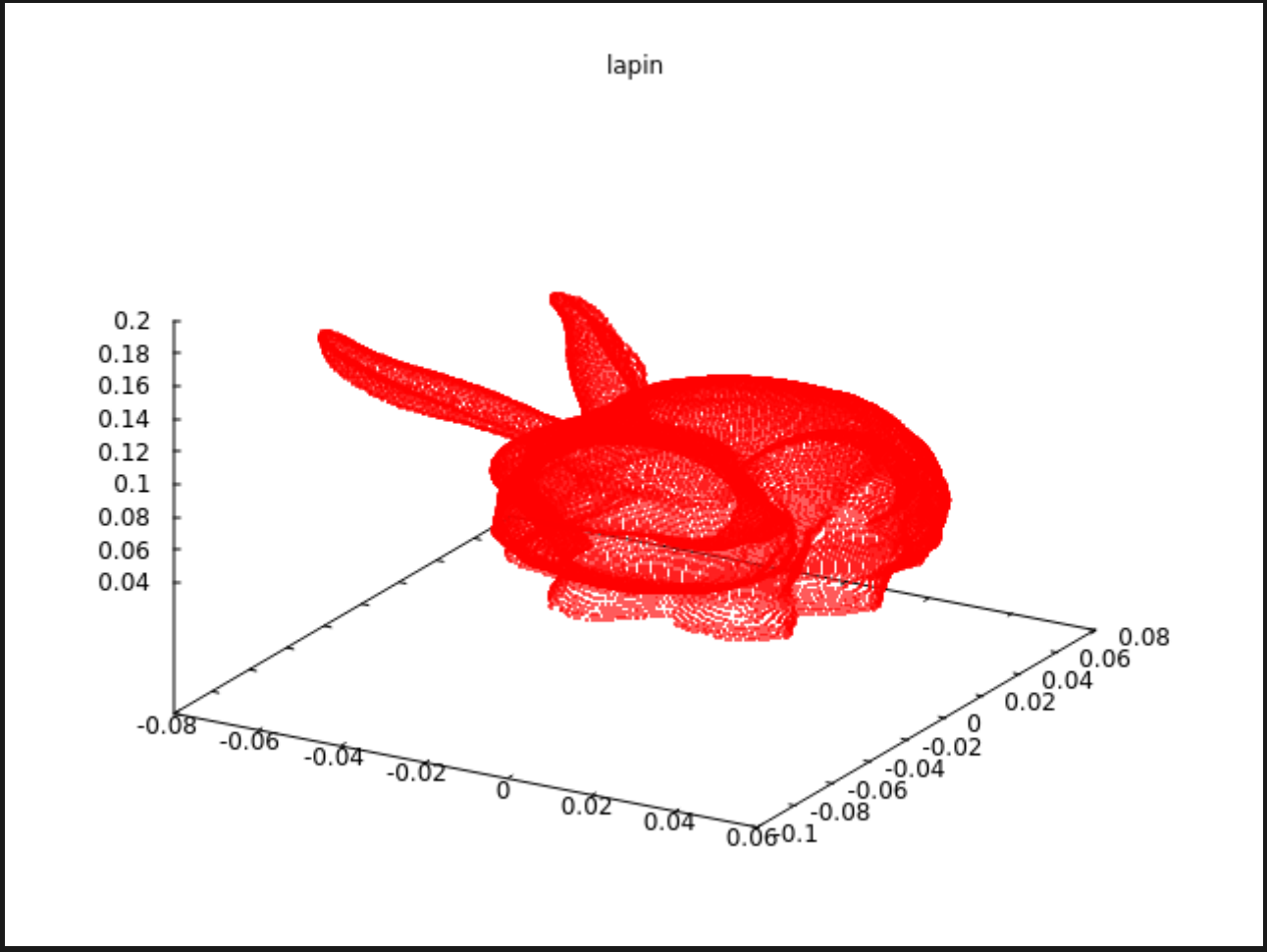
MODULE 201

FARTHEST POINT SEARCH

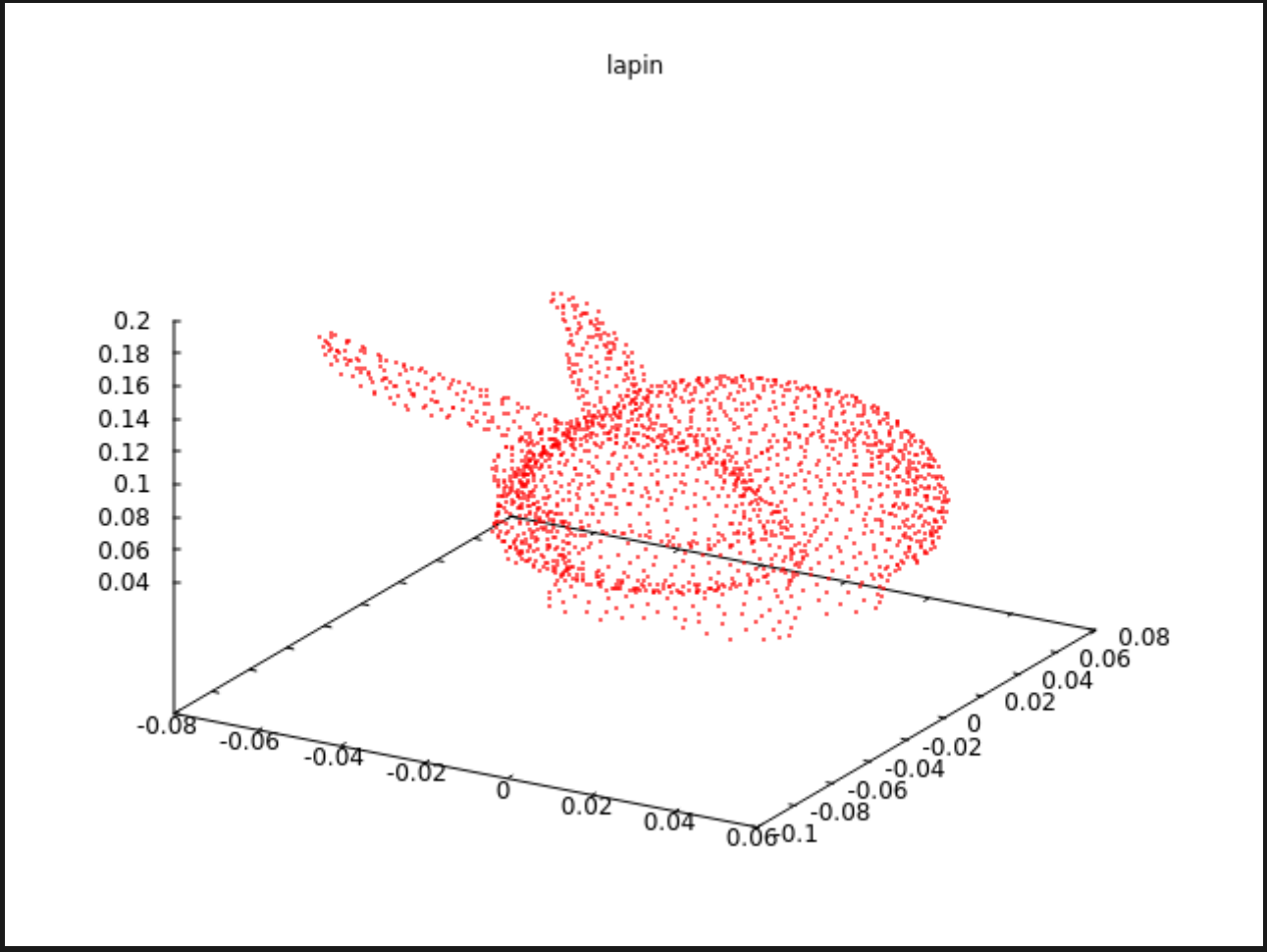
INTRODUCTION

- Algorithme de réduction d'un modèle 3D
  - prendre un ensemble de sommets
  - produire un sous-ensemble
  - garder la structure globale du modèle

INTRODUCTION



INTRODUCTION



EXIGENCES (1/2)

Ecrire un programme pour :

- Prendre en argument un nom de fichier, contenant les sommets 3D
- Lire le fichier
- Calculer le modèle 3D réduit en utilisant l'algorithme FPS
    - utiliser une taille arbitraire pour le résultat : 2048

EXIGENCES (2/2)

- Ecrire dans un fichier la liste des sommets sélectionnés
  - Même format que l'entrée
- Utiliser gnuplot pour afficher les sommets des deux modèles (original et réduit)

VOUS AVEZ

- (cette présentation)
- Un projet cargo initial pré-configuré
- Les jeux de données `data1.txt` et `data2.txt`
- Une implémentation de référence en Python
  - à comparer si l'algo n'est pas clair.

DATASET 1

- fichier texte
- un sommet par ligne (30,500+ sommets)
- coordonnées x, y, z
  - sérialisées comme nombre à virgule flottante
  - séparées par des espaces

DATASET 2

- télécharger ici
- identique au jeu 1, sauf que:
- ce n'est pas un lapin
- 1M+ sommets
- chaque sommet a 3 composantes en plus:
  - octet non signé
  - composantes couleur rouge, vert, bleu.

FARTHEST POINT SEARCH

- Sélectionner aléatoirement un somment
- Tant qu'il y a des sommets à sélectionner
  - En sélectionner un tel que
    - parmis ceux non sélectionnés
    - le plus loin de tout ceux déjà sélectionnés

FARTHEST POINT SEARCH

Utiliser la distance euclidienne:

- Sommet A: Ax, Ay, Az
- Sommet B: Bx, By, Bz

SQRT((Bx-Ax)^2 + (By-Ay)^2 + (Bz-Az)^2)

ROADMAP -- OUTLINE

- Définir une structure pour un sommet 3D
- Récupérer le nom du fichier donné en argument
- Lire le fichier comme un Vec de sommets
- Implémenter l'algorithme FPS.
- Implémenter la sérialisation des résultats
- Utiliser la librairie gnuplot pour afficher les modèles.

ROADMAP -- RUST IDIOMATIQUE

- Utiliser le style fonctionnel autant que raisonnable
- Utiliser la dérivation automatique des traits Copy et Clone
- Utiliser des références

BEST PRACTICES (1/3)

- Ecrire des tests unitaires
  - et les exécuter
- Ecrire de la documentation
  - et générer la doc
- Ecrire du code d'exemple dans la documentation
  - par ex, comment utiliser une fonction
  - ré-exécuter les tests avec cette doc

BEST PRACTICES (2/3)

Utiliser le linter:

`cargo clippy`

BEST PRACTICES (3/3)

- Compiler en modes debug et release
  - comparer les performances
  - comparer avec l'implémentation Python