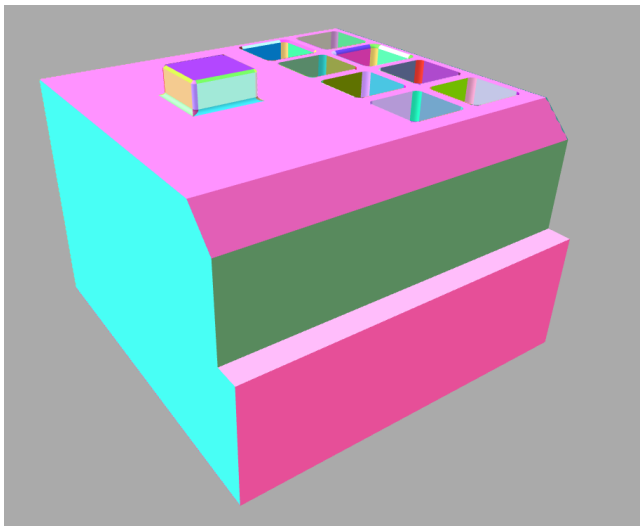
Intro

The purpose of this exercise is to identify sets of entities that may represent a certain type of machining feature given a limited set of structured geometry data. The validity of the analysis results is easiest to digest in a visual 3d format, so you will also be visualizing your results in a client environment.



We will be looking at the above model, which is provided in a mesh format suitable for display in browser environments. Every face is what we call an entity. Each entity has been assigned a unique color so we can map it to a unique entityId that is shared across systems.

You will be given access to a git repository in which there are two directories. The first is a data dump of files described below. The second is a toy Typescript/React/Webpack app that displays the .glb file.

Data Summary

The following files have been provided:

- colored_glb.glb
    - This is a GL transmission format file containing the meshed model
- adjacency_graph.json
    - This is a map where the keys are an entityId and the values are sets of entityIds that are neighbors/touching
- adjacency_graph_edge_metadata.json
    - This is a map where the keys are edges of the adjacency_graph and the values are sets of GraphEdgeType enumerators containing information about whether edges represent concave, convex, and/or tangent neighbor pairs

    - Each edgeId key is of the format “<entityIdOne>-<entityIdTwo>”. Reversal of the ids will also always be present.
- entity_geometry_info.json
    - This is an array of EntityGeometryInfo data. Each entity will have an entry
- rgb_id_to_entity_id_map.json
    - This is a map of rgbId strings to entityId strings. rgbIds correspond to the colors of the each entity in the gltf file
    - An rgbId is of the format “<r>-<g>-<b>”, where r, g, and b are integers from 0-255. Red would be represented as “255-0-0”. Colors are represented as floats from 0-1 in the gltf file.
- data_types.cs
    - Data types for adjacency_graph_edge_metadata and entity_geometry_info

Goal:

- Write an algorithm that uses the above data to identify sets of entity ids that probably represent a pocket
- Extend the toy application code to display your results in a way that can be quickly reviewed by someone who isn’t familiar with your algorithm

Notes/Tips:

- Please reach out if you are blocked for more than an hour!
- There is no language restriction or suggestion on the analysis. It is okay if the integration between your results and the toy app is not pretty.
- threejs provides a useful widget where you can upload a gltf file and examine its contents. Set the Environment to MODELVIEWER under the SCENE tab
    - For the purposes of this exercise, the data has been massaged so that the entityIds map to the name of each mesh that represents a face. If you examine the .glb in the widget, “Product_1_3” represents entityId “3”, “Product_1_12” represents entityId “12”, etc.
    - This visualizer should be enough to get you started on analysis without getting bogged down in the application code
- In EntityGeometryInfo, normals and points are arrays of length 3 corresponding to x/y/z values for that vector. Normal values are all unit vectors
- Entity normals always point away from the solid represented by the faces. Here is vector showing the direction of an entity’s centerNormal: