Software Design Challenge

Mesh Manipulation and Target Packing

Thank you for your interest in the Focused Ultrasound Laboratory at Sunnybrook Research Institute. The following challenge is designed to assess your technical competency in engineering design and communication. Your response will determine whether you are selected for a technical interview with our engineering team. Please return a complete solution within 7 days of receipt.

This challenge will consist of two parts: 3D mesh manipulation and 3D shape packing.The first part of the challenge will require you to provide runable code, while the second part of the challenge will *not* require runnable code, but a short report on how to solve the given problem.

Part 1: 3D Mesh Manipulation

Suppose you would like to perform mesh manipulation on various 3D shapes represented as 3-dimensional meshes of polygons (i.e. the faces) and points (i.e. the vertices). Write a Python program containing functions that have the capabilities listed below. You are strongly encouraged to utilize common mesh representation and manipulation libraries such as `vtk` or `trimesh`

Part 1a: Build 3D Shapes

Write functions that can generate a 3D sphere mesh and a 3D cone mesh, and return their data representation. Ensure that the functions allow the user to control the radius of the generated sphere and cone, as well as the height of the cone.

Part 1b: Scale 3D Shapes

Write a function that takes an input mesh (which would be your 3D shape mesh) and can return an output mesh that is the input mesh but scaled equally along all of its axes. Ensure that the function allows the user to provide an input scaling value.

Part 1c: Compare 3D Shapes

Write a function that takes two input meshes and compares them to each other (assume that the input meshes may be generated from different algorithms / sources / parameters etc.). The function must return a boolean value based on whether the meshes are "the same". Using comments within your code, elaborate on how you define mesh "sameness" or "closeness", and explain the methodology of your implemented approach. If you find multiple approaches for determining sameness, you may cite the alternatives and elaborate on why you chose your approach over the alternatives.

(Bonus: assume the input meshes may belong to different coordinate systems)

Part 1d: Tests

Write tests for the functions you have written in the previous parts. If these tests require test data, please package the test data along with your code so that we can run your tests successfully when you submit your solution.

Endeavour to write code that is clean, modular, and well documented. Try to demonstrate professional practices you have developed in your career so far. We will assess these aspects within your code, in addition to the correctness of the algorithms.

Part 2: 3D Shape Packing

Imagine a non-symmetrical, non-convex arbitrary volume, such as the one shown in Figure 1. We are interested in filling this volume with **very small spherical volumes** such that all the spheres are **within the volume** and packed as optimally as possible by utilizing the hexagonal close-packed arrangement.

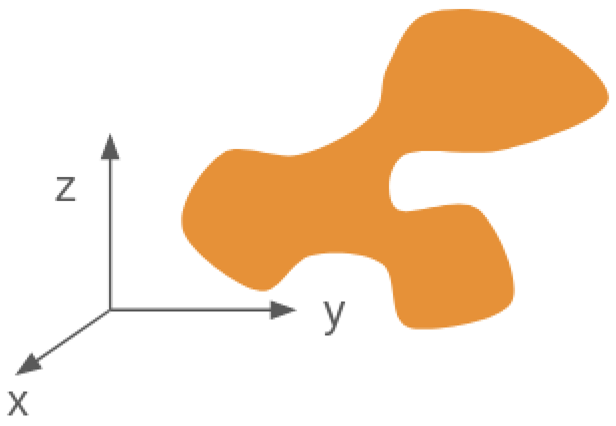

Figure 1. Example of a non-symmetrical, non-convex arbitrary volume.

Write a short report answering the following points:

- Break the problem down into a software design. Provide **pseudocode signatures** for any methods or classes you would use to achieve this.
  - Please provide **algorithmic details** in **pseudocode** for your approach in ensuring that all the spheres are within the volume. No explicit code is necessary, however we would like to see **a step-by-step algorithm** for this.
  - Justify your approach (why did you choose to do it this way?), and cite any possible alternatives and resources you may have consulted.
- Discuss **programming languages** you think are suitable for implementing these algorithms, and any **libraries** that could be helpful, if you are aware of any
- Describe your approach to **testing and validation** for this kind of problem

You should *not* solve this problem with code. We want to assess how you would design the **solution** to the problem, how you would **analyze its requirements and implementation details**, how you would **test the validity of the potential solution**, and how you would **communicate** all of this to others. You are welcome to reference materials to help you solve this problem.