It's all about 3D! - Workshop - Alessandro Fasse - 28.11.2018

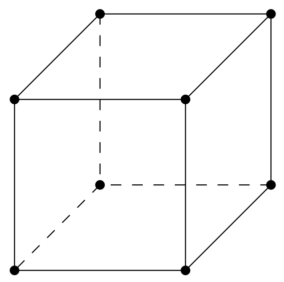
Email: a.fasse@gmx.de

Conventions: For a better notation we assume that we have n vertices and m faces, *i.e.*

$$\texttt{np.shape(vertices)} = (n, 3)$$
$$\texttt{np.shape(faces)} = (m, 3).$$

Exercise 1 - Mesh construction

Write down the list of vertices and faces for a triangulated mesh of a cube with size length one, one vertex is the origin $(0, 0, 0)$ and only positive coordinates. Check your result with the method `print_3d_mesh()` from `tools.py`.



Exercise 2 - Normal vectors

Given is the following triangulated mesh.

$$\texttt{vertices} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \texttt{faces} = \begin{pmatrix} 0 & 1 & 2 \\ 3 & 0 & 1 \\ 3 & 2 & 0 \\ 3 & 2 & 1 \end{pmatrix}.$$

(a) Draw a picture of the mesh on paper (or in your head) and validate your result via the `print_3d_mesh()` method from `tools.py`.

(b) Is the surface closed?

(c) Determine on which triangle of the mesh the normal direction is not chosen with respect to our convention.

(d) Write a python method `compute_face_normals(vertices, faces)` that computes for every triangle the corresponding normal vector (with respect to our convention) and return all of them inside of a single numpy array (matrix).

(e) Write a python method `compute_triangle_areas(vertices, faces)` that computes the area of every triangle in the mesh and returns them as a $m \times 3$ numpy array. *Hint:* Some parts of the computation of the normal vectors and properties of the cross product are helpful!

Exercise 3 - Connectivity

(a) How can you determine if a vertex v is a direct neighbour of another vertex w.

(b) Write down a python method `compute_edges(faces)` that returns the set of all (directed) edges of the mesh.

(c) Write down a python method `compute_neighbors(faces)` that computes for every vertex the set of their direct neighbours. Test your result with the above meshes.

(d) How can you determine the 2-level neighbours of a vertex? (No coding needed)

Exercise 4* - Boundaries

Consider an arbitrary mesh given as a lists `vertices` and `faces`.

(a) Explain why is the boundary of a mesh is independent the coordinates of the vertices, *i.e.* of the list `vertices`.

(b) Determine an algorithm that can detect if an edge of a mesh is a boundary edge (No code implementation needed, just gather some ideas).

(c) Write down a python method `compute_boundary(faces)` that computes the boundary vertices of a mesh. You can use the method `compute_edges(faces)` of Exercise 3 (b).