Architectural Geometry — Worksheet 1

This worksheet is intended to help us understand (a) spherical and cylindrical coordinates, and (b) the standard representation of triangle meshes. We will use the primitives and functions created here in the subsequent worksheets.

You are allowed (and indeed encouraged) to discuss the tasks with other participants, but you are not allowed to hand in solutions produced by somebody else. If you get stuck, please do not hesitate to contact me, and I will do my best to help you get past the obstacle.

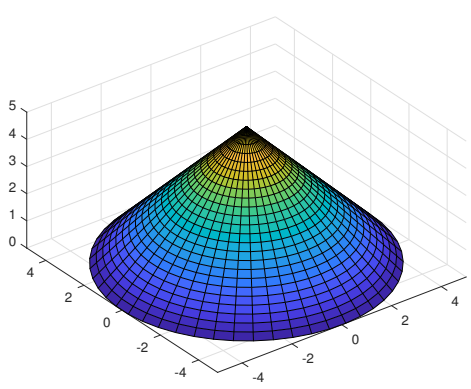
Write up your solutions neatly in a document, and send it together with an archive (e.g. zipped) containing your code to me at `marten.wadenback@chalmers.se` before the lecture on the 10th of April.

Task 1: Study Chapter 3.3 *Platonic Solids*. Study the file `tetrahedron.m` and create corresponding functions for generating cubes and octahedrons. You can view each of the objects using the provided function `renderObject`. Be careful to specify the triangles anticlockwise.

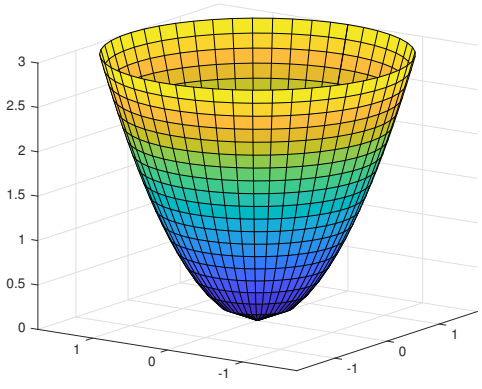
```
% Example usage of colorObject and renderObject
T = tetrahedron;
T = colorObject(T,'red'); % Or try e.g. T = colorObject(T,[1,0.5,0.5]);
renderObject(T);
```

Task 2: Study the section about cylindrical coordinates on p. 11. For each of the surfaces (a)–(d) below, try to figure out what function $r(\varphi, z)$ best describes the radius. Use the provided function `cylinderplot` to experiment with different function expressions until you can reproduce the plots below.

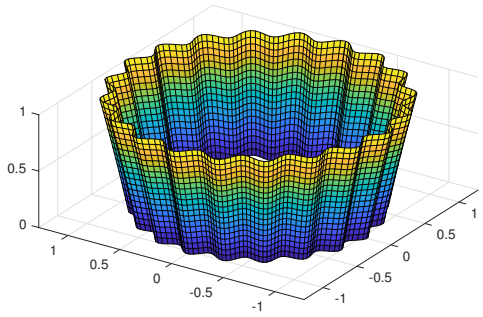
```
% Example usage of cylinderplot
phi = linspace(0,2*pi,40);
z = linspace(0,2,20);
r = @(phi,z) (6+cos(5*phi)).*z./(1+z.^2); % Define the radius function
cylinderplot(r,phi,z);
axis equal;
```



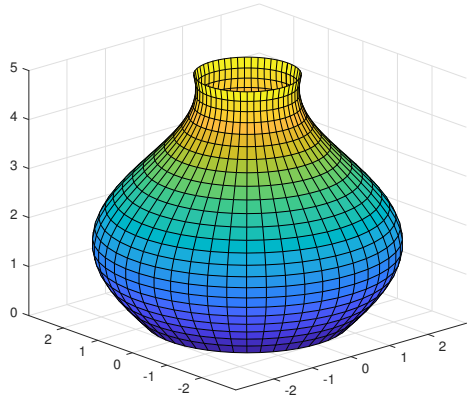
(a)



(b)



(c)



(d)

Task 3: Study the section about cylindrical coordinates on p. 11. Write a function `cylinder` which creates the vertex list and triangle list for (an approximation of) a cylinder with radius $r = 1$. The function should return an object that can be fed into `renderObject`. Let the user specify m and n, and place vertices at $(0, 0, \pm 1)$ and at

$$(x, y, z) = \left(\cos\left(\frac{2k\pi}{n}\right), \sin\left(\frac{2k\pi}{n}\right), \frac{\ell}{m}\right), \tag{1}$$

where $0 \leq k < n$ and $-m \leq \ell \leq m$. What is another word for this kind of solid?

Hint: Do the top and bottom parts last.

Task 4: Create a function `subdivide` (start from the skeleton file `subdivide.m`), which refines the mesh of an object by splitting each triangle into four smaller triangles. Verify that it works as expected on the primitives you have available (e.g. `T = subdivide(cube)`).

Hints:

- If the original mesh has n triangles, the new mesh will have $4n$ triangles. When you split the triangle with index j, you may e.g. place the four new triangles at indices j, $n + j$, $2n + j$, and $3n + j$ in your new triangle list.
- To avoid adding the same midpoint vertex several times, use a `containers.Map` to keep track of which midpoint vertices you have added. For this you need a hashing function (provided for you as `hash = @(x) sort(char(x));` in `subdivide.m`). Try working through the example below to better understand how to use `hash` and `containers.Map` before you start coding.

```
% Example usage of hash and containers.Map
hash = @(x) sort(char(x));
A = containers.Map(); % Creates an empty map
h1 = hash([3 4]); % Hash the index pair [3 4]
h2 = hash([4 3]); % Hash the index pair [4 3]
h3 = hash([5 3]); % Hash the index pair [5 3]
all(h1 == h2) % Returns true
all(h1 == h3) % Returns false
A.isKey(h1) % Returns false, A is still empty
A(h1) = 10; % Add the pair [3 4] to A, associate it with an index
A(h1) % Returns 10 (the index of the midpoint of [3 4])
A(h2) % Also returns 10 (the index of the midpoint of [4 3])
```

```
A(h3) % Error, [5 3] has not yet been added
A(h3) = numel(A.keys)+1; % Add [5 3] and associate it with 2
```

Task 5: Study Chapter 3.7 *Geodesic Spheres*. Try to determine (by hand) the number of triangles and vertices there will be on a geodesic sphere created from an octahedron that has been refined k times. Use your functions `octahedron` and `subdivide` to create geodesic spheres according to the two methods described in the chapter (but starting with an octahedron instead of an icosahedron), and verify (or find the error in) your expected number of triangles and vertices. Hint: The `norm` command is useful here. Can you verify the difference between the results obtained using the two methods?