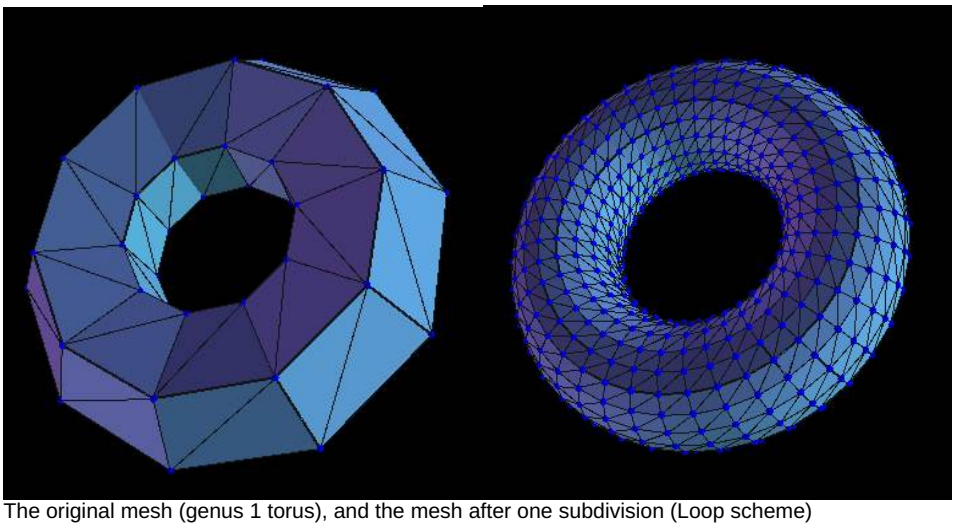
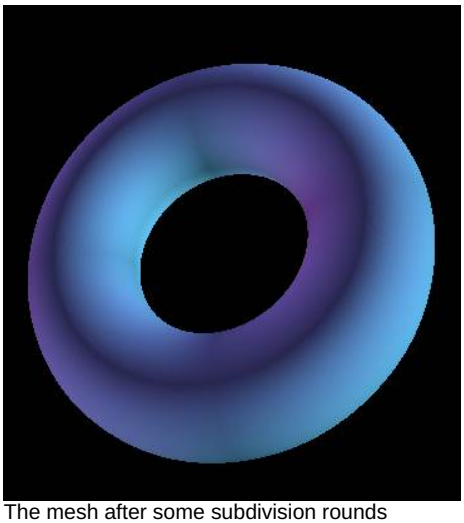
TD 6 - Subdivision surfaces

(Luca Castelli Aleardi)

The goal of this TD session is to implement subdivision schemes for surface meshes



The original mesh (genus 1 torus), and the mesh after one subdivision (Loop scheme)



The mesh after some subdivision rounds

Example of surface subdivision using the Loop scheme.

1. Getting started (a Processing frame for 2D and 3D rendering)

Files to download today

Here are some files to download (and extract to the main folder of your Eclipse project):

- meshes.zip: examples of triangle meshes (OFF format).

Libraries to add (project /LIB):

- Jcg.jar: library for the manipulation of triangle meshes (here are the sources)
- TC.jar: input/output from files.

For the simplification and rendering of meshes:

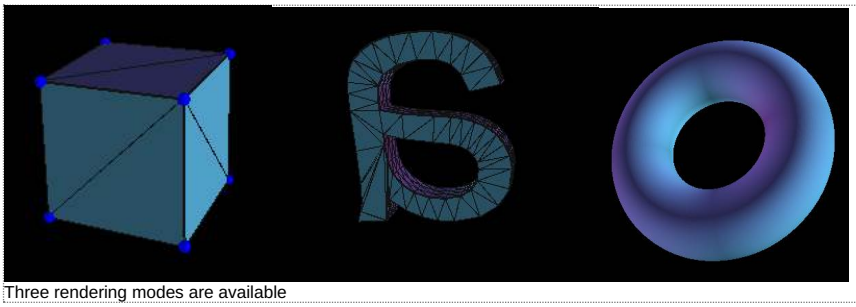
- SurfaceMesh.java: class defining main methods for drawing the mesh
- MeshViewer.java: main drawing class, allowing to rotate the scene in 3D.
- ArcBall.java: simple class allowing to rotate the 3D scene with mouse events.
- MeshSubdivision.java: abstract class defining main methods for surface subdivision
- LoopSubdivision.java: skeleton of the class implementing the Loop subdivision

TD documentation: instructions

Important: here are a few slides illustrating how to implement subdivision schemes for this TD.

class MeshViewer

Class MeshViewer provides simple methods for drawing a surface mesh in a 3D frame; it also provides a simple trackball (allowing to rotate the 3D scene with mouse events).
You can choose three distinct rendering modes (as illustrated below): press **'r'** key (on the keyboard) in order to change rendering mode.
Method setup() allows to select the preferred subdivision scheme: press **'s'** key (on the keyboard) in order to run the subdivision procedure.



Three rendering modes are available

1. Loop subdivision

Goal

Given a triangle mesh (class `Polyhedron_3<Point_3>`), we want to subdivide it, by computing the ***Loop subdivision*** (see slides for this TD).

- complete method `subdivide()` of class `LoopSubdivision` (which extends class `MeshSubdivision`).

Suggestion: in order to implement the Loop subdivision scheme, we suggest to you to follow the steps below (feel you free in choosing and implementing a different solution)

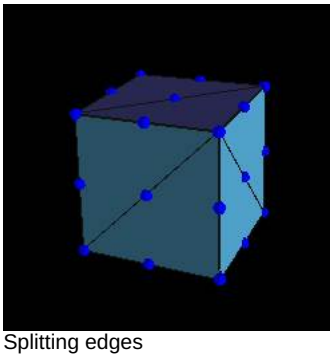
1.1 compute edge point coordinates

Add/complete method `computeEdgePoints()` to class `LoopSubdivision`, which computes the list of edge points (just geometric locations).

1.2 split edges by inserting edge points

Now we want to insert edge vertices: just use `splitEdge(Halfedge<X> h)` operator (already implemented in Jcg).

- add method `splitEdges(...)` to class `LoopSubdivision` which perform the splitting of all original edges.
- complete method `subdivide()` which calls previous method.



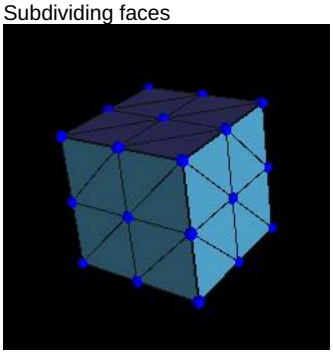
Splitting edges

1.3 subdivide faces

In order to subdivide each face (of degree 3+3) into 4 triangle faces, you can perform three `splitFace()` operations (see slides for this TD).

- add method `subdivideFace(f)`, which perform the subdivision of a face.
- complete method `subdivide()` in order to subdivide all faces (recall that only original faces must to be subdivided).

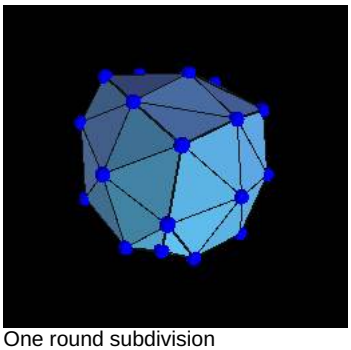
Subdividing faces



1.4 compute new locations for original vertices

To conclude, it only remains to compute the final new locations of vertices of the original mesh.

- add method `computeNewVertexLocation(Vertex<Point_3> v)` in order to compute the new coordinates of original vertices.



One round subdivision

2. Catmull-Clark subdivision

2.0 A subdivision scheme for polygonal meshes

Given a triangle mesh (class `Polyhedron_3<Point_3>`), we want to subdivide it, by computing the ***Catmull-Clark subdivision*** (see slides for TD).

- write a new class `CatmullClarkSubdivision` (which extends class `MeshSubdivision`) and complete its method `subdivide()`.

Suggestion: you can follow the same steps as above, taking a look to the slides for this TD (feel you free in choosing and implementing a different

solution). For testing, uncomment the corresponding line in the class MeshViewer (in the setup function).