Mesh simplification

Vertex clustering

Michał Ciołczyk, Michał Janczykowski

Mesh simplification

For a mesh M=(V, F) find M'=(V', F') that is in some way "simpler".

Usually, we want to reduce the amount of vertices (so we want $|V'|<|V|$).

Why bother?

Many applications, for example:

- when we have oversampled data (e. g. from 3D scanner)
- when we want to render objects with certain level of detail (LOD) - e. g. if we want to have smaller level of detail of objects that are further away from the camera
- when we want to have multi-resolution models for some geometry processing

Mesh simplification techniques

There are several techniques that can be used:

- **vertex clustering,**
- **incremental decimation,**
- resampling algorithms,
- mesh approximation algorithms.

Vertex clustering

- efficient algorithm,
- computational complexity can be linear to the amount of vertices in mesh (it is dependent on the computing complexity of computing clusters' representatives),
- can lead to meshes that are non-manifold or have not satisfying quality.

Algorithm

1. Divide the input mesh into clusters (for example, divide the volume near the mesh into cubes of edge length 2ε)
2. Assign the vertices to the clusters.
3. For each cluster compute its representative.

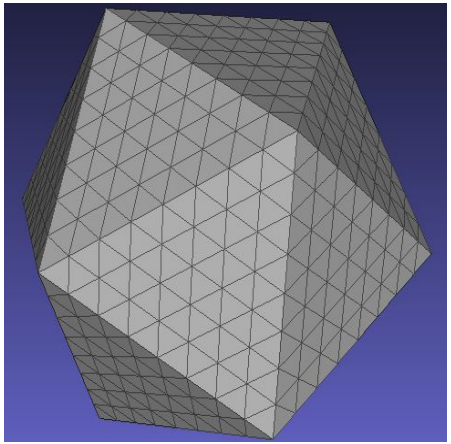
Algorithm (continued)

4. For each face in original mesh:
   a. Get representatives for each of the face's vertex.
   b. If the representatives of the face's vertices make triangle, add this triangle to the output faces.
5. Return mesh made by representatives and output faces.

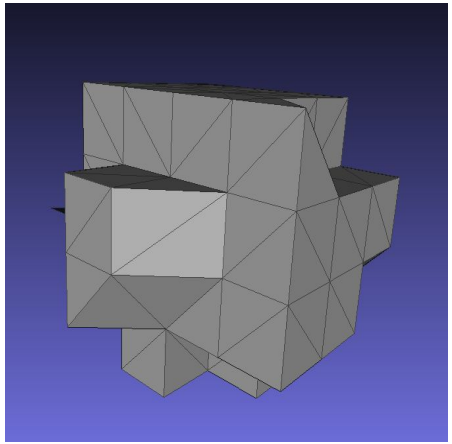
Representative methods

- How we compute the representative of the cluster is crucial for the output mesh quality.
- Several approaches:
  - Cluster center,
  - Mean,
  - Median,
  - Quadric errors (the squared distance from the neighbor faces).

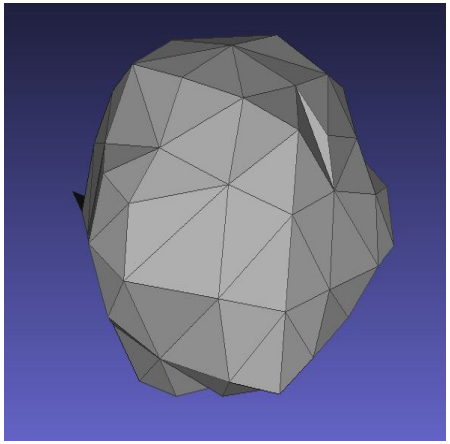
Vertex clustering



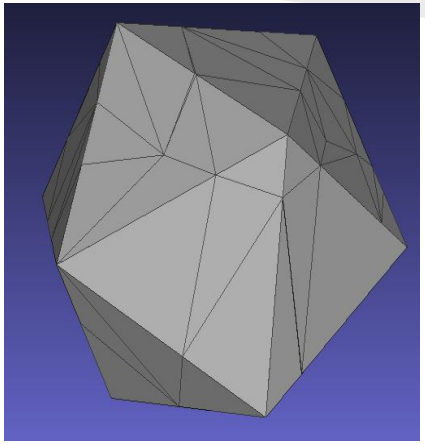
Input mesh



Cluster centers



Mean or median



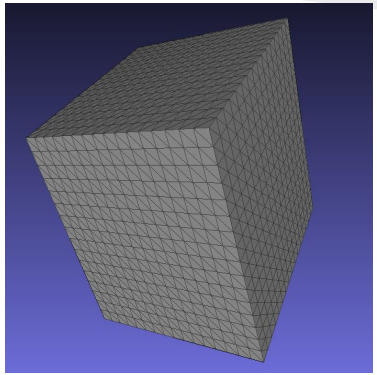
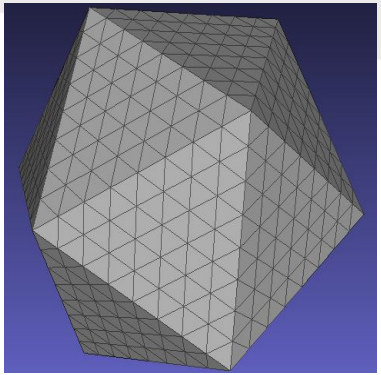
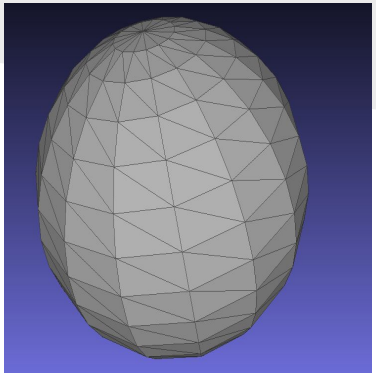
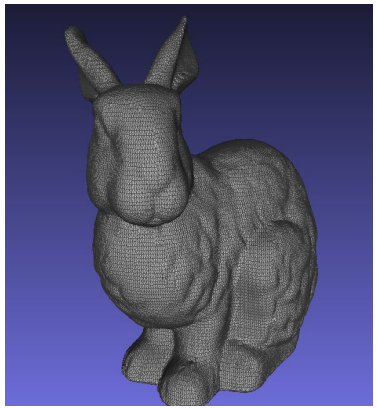
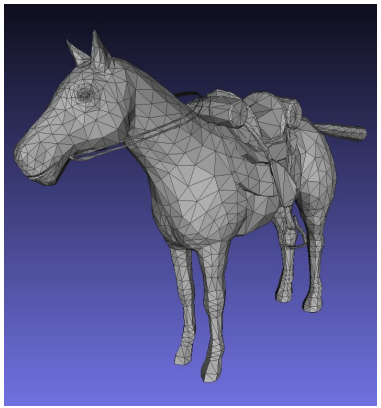
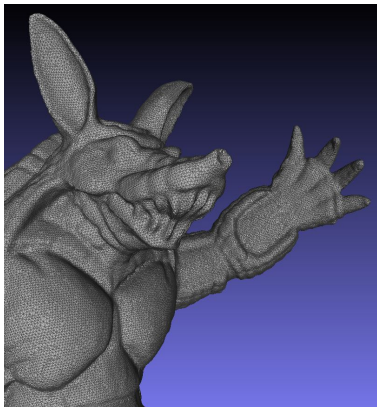
Quadric errors

Implementation

- algorithm written in Python,
- uses CGAL with CGAL Swig bindings for Python,
- available at https://github.com/salceson/modelowanie,
- all previously mentioned representative computation methods implemented.

Test meshes

- Cube
- Icosahedron
- Sphere
- Bunny
- Horse
- Armadillo

Time comparison

Test	Amount of vertices	Mean running time [s]			
		Cluster center	Mean	Median	Quadric errors
cube	**2 116**	0.150	0.175	0.183	0.715
sphere	**242**	0.024	0.029	0.030	0.134
bunny	**35 947**	3.144	3.439	3.568	13.872
horse	**2 831**	0.251	0.287	0.311	1.231
armadillo	**172 974**	16.334	18.611	19.711	77.557

Thank you!

Image: http://pixshark.com/thank-you-slide-images.htm