IFT6113 - Assignment 1
Practice, 40 pts
Deadline: October 3rd, 23:59

Task description: Subdivision is a classical technique that turns a low-resolution mesh (which is easier to create) into a beautiful high-polygonal mesh, ready for rendering. Your task is to implement **two** subdivision schemes of your choice for closed triangular meshes using an existing half-edge data structure. We recommend you to start from implementing subdivision process, and then optimize it with half-edges.

Options of subdivision schemes: **Loop**, **Butterfly**, **Sqrt(3)**.

Useful links: slides UBC, slides Stanford, sqrt(3) subdivision paper.

Code template: We provide code templates for Python and C++, available on github.com/ivanpuhachov/ift6113_2022

Submission

Submit you work via private post in Piazza with zip archive with your code and name (e.g. `hw1_python_john.zip`). Update `readme.md` or provide instruction in your post on how to run your code.

Make sure you understand every line of the code you write.

Requirements

We will deduct points for particularly inefficient solutions.

Half-edge data structure

Files are provided with the template. C++ github.com/yig/halfedge; Python github.com/yig/trimesh.

Usage

Your command-line tool should take the input mesh `.obj` and a number of subdivision operations to perform. Resulting mesh should be stored in `output` subfolder. Provide instructions on how to use your code. We propose the following command line interface:

```
> python loop.py --input ../input/cube.obj -n 3
```

– runs 3 iterations of loop scheme on `cube.obj` and it outputs the result in `output/cube_loop_3.obj`.

Additional visualizations may be useful, but are not required.

Bonus points

You can get extra marks for implementing any of the following options (bonus at our discretion). For the description of each option, please refer to online literature on subdivision.

- Implement vertex projection to the limit surface.
- Implement normal computation using the limit surface

Subdivision schemes

Each iteration of subdivision scheme usually consists of the following steps:

- insert new vertices
- modify mesh connectivity (remove some of the old connections, add new ones)
- relocate “old” (existing before this iteration) vertices, e.g., set each to weighted average of its neighbors.
- relocate “new” vertices.

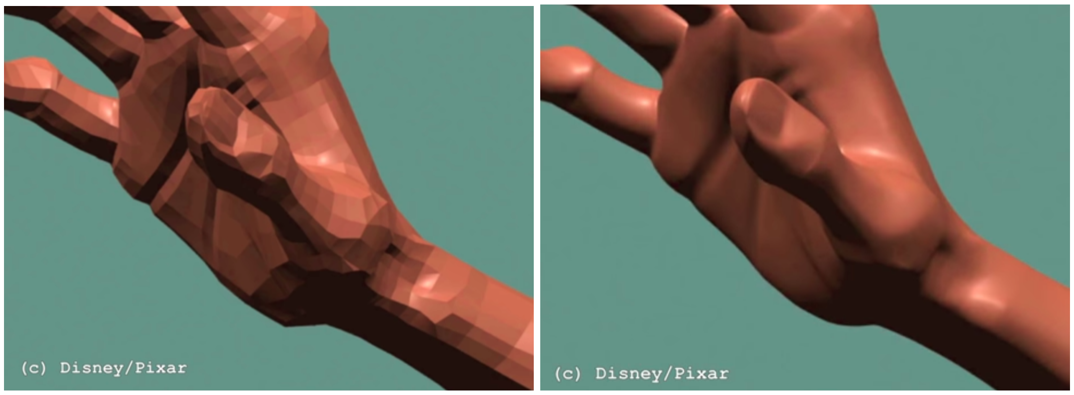


Figure 1: Subdivision example on Pixar's «Geri's Game» . These images were taken from youtube Numberphile video: youtu.be/mX0NB9IyYpU

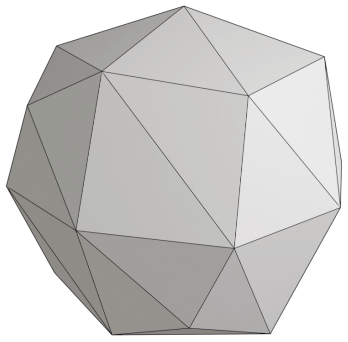
Notations

vertices[1]
stencil — the combination of weights used in a subdivision scheme
$\mathcal{N}(v)$ — neighbour vertices set of vertex v
$n = |\mathcal{N}(v)|$ — valence of a vertex, i.e., number of neighbours.

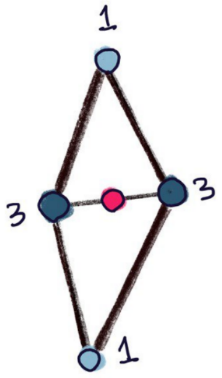
[1]More subdivision schemes (including subdivision for quad meshes) here

Loop scheme

New vertices are initially inserted in the middle of each edge, so each face splits into 4. We then re-calculate new positions for old and new vertices.



(a) Subdivided cube (1 step)



(b) Computational stencil for new vertex (pink) position

Figure 2: Loop subdivision

Old vertices update

Old vertex v with n "old" neighbors is replaced to a new position, derived as a weighted average:

$$w_n = \frac{64 \cdot n}{40 - (3 + 2\cos(2\pi/n))^2} - n$$

$$\hat{P}(v_i) = \frac{1}{n + w_n}[w_n \cdot P(v_i) + \sum_{k \in \mathcal{N}(v_i)} 1 \cdot P(v_k)]$$

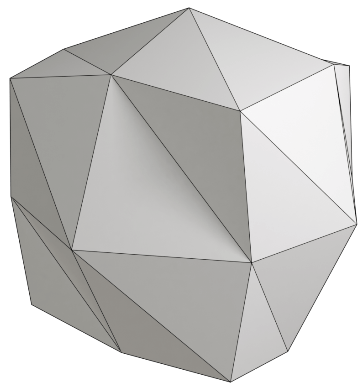
New vertices update

For a vertex $\hat{v}$ that was inserted in the middle of edge v_1v_2, that is shared between triangle faces $F_1 = \Delta v_1v_2v_3$ and $F_2 = \Delta v_1v_2v_4$, we compute new position as weighted average (see Fig.2) of its neighbours:

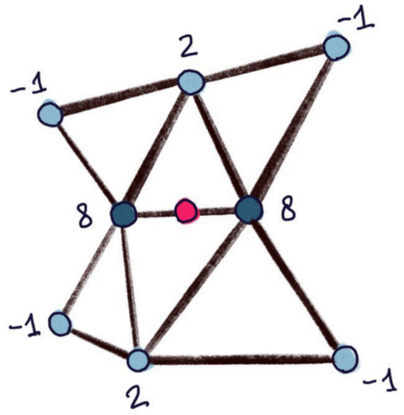
$$\hat{P}(\hat{v}) = \frac{1}{8}[3 \cdot P(v_1) + 3 \cdot P(v_2) + P(v_3) + P(v_4)] \tag{1}$$

Butterfly scheme

Same as in Loop scheme, new vertices are associated with the edge midpoints.



(a) Subdivided cube (1 step)



(b) Computational stencil for new vertex (pink) position

Figure 3: Butterfly subdivision

Vertex update

Old vertices don't move. New vertices: We take into account 1-ring and 2-ring face neighbourhood of the edge when calculating new position for midpoint. See Fig. 3 and equation 1 for reference.

Sqrt(3)

Corresponding paper: graphics.rwth-aachen.de. Note that new vertices are inserted the centre of each face.

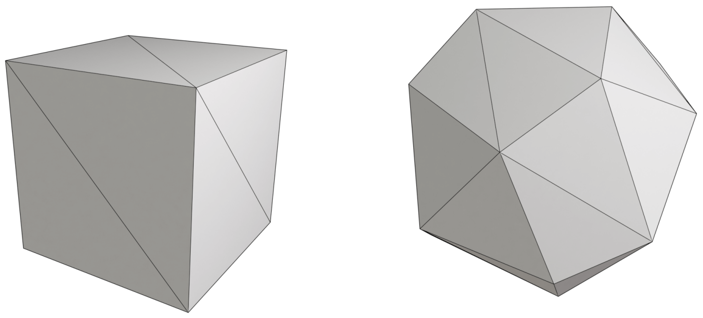


Figure 4: Sqrt(3) subdivision applied to cube

Note: you will need to implement edge flipping to get smooth results.

New vertex update

New vertices are set to the barycenter of associated face $F = \Delta v_1 v_2 v_3$:

$$\hat{P}(\hat{v}) = \frac{1}{3}[P(v_1) + P(v_2) + P(v_3)]$$

Old vertex update

For an "old" vertex v that has n "old" neighbours $(v_1, v_2, ...v_n)$ we set its position to be:

$$\hat{P}(v) = (1 - \alpha_n)P(v) + \frac{\alpha_n}{n}\sum_{i=1}^{n} P(v_i)$$

where $\alpha_n = \frac{4 - 2\cos(2\pi/n)}{9}$.