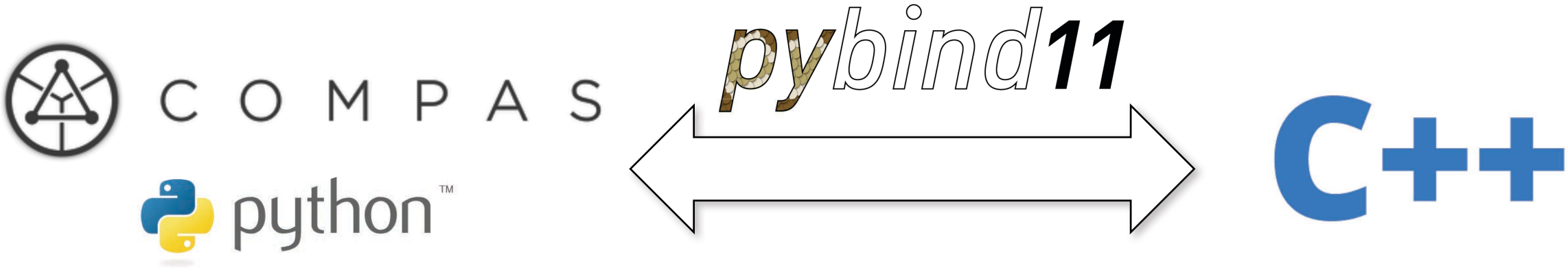
COMPAS

COMPAS Workshop: Interoperability between COMPAS and C++

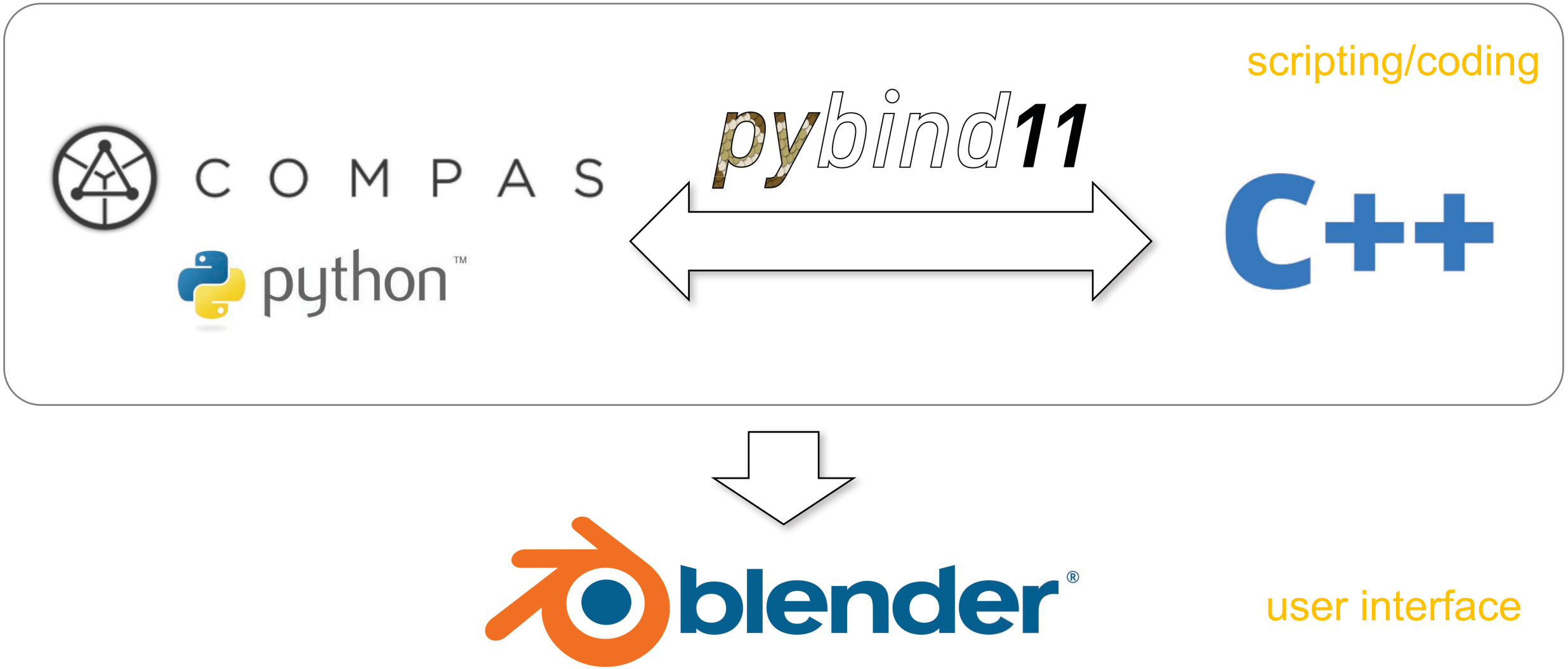
Ziqi[1] Peng[1] Tom[2]

[1] EPFL [2] ETH zürich

Goal: Interoperability between COMPA and C++



Goal: Interoperability between COMPA and C++



Goal: Interoperability between COMPA and C++

pybind11 is a lightweight header-only library that exposes C++ types in Python and vice versa, mainly to create Python bindings of existing C++ code.

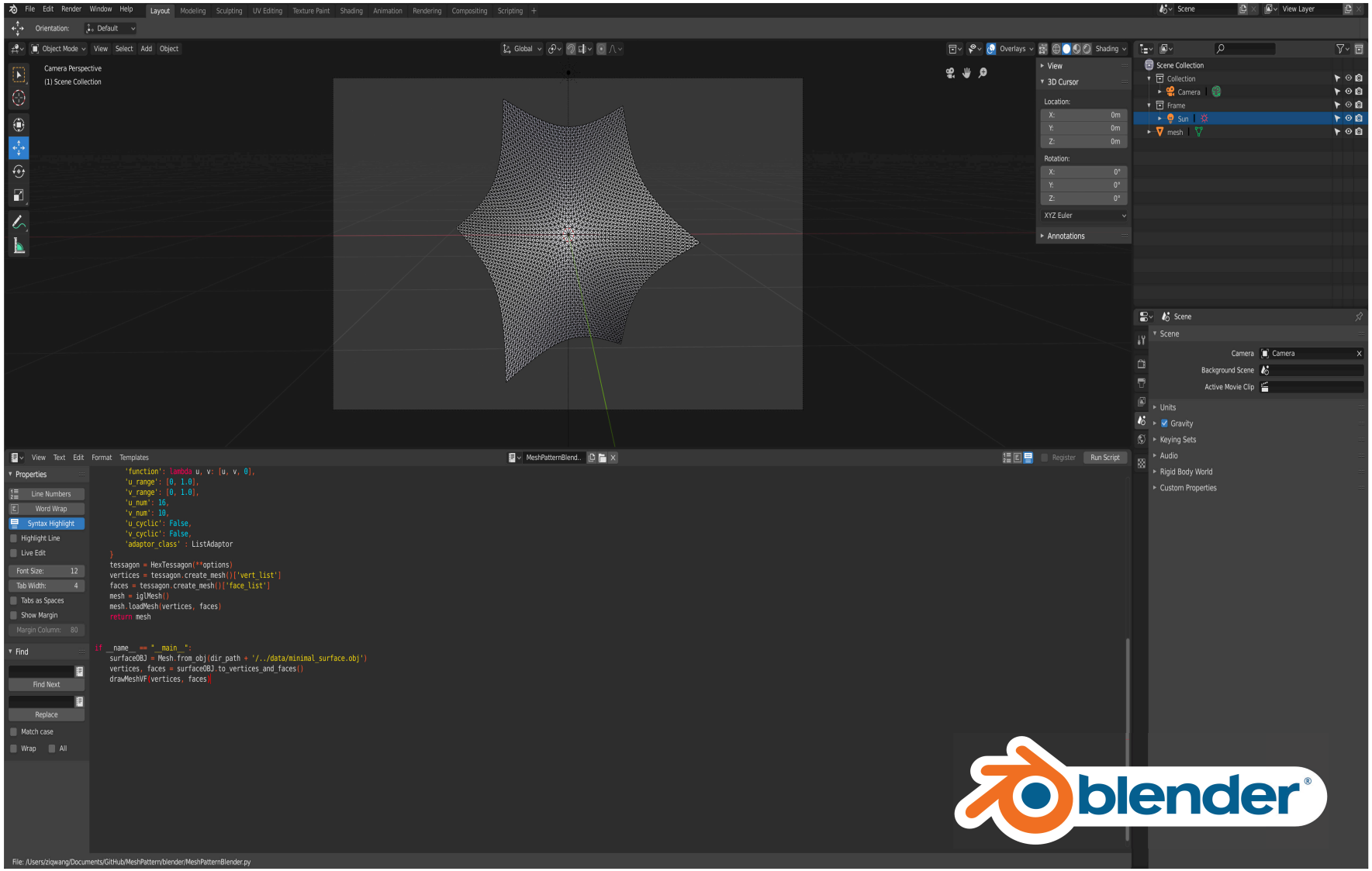
Core features:

- Functions accepting and returning custom data structures
- Instance methods and static methods
- Callbacks
- Iterators and ranges
- Custom operators
- Single and multiple inheritance
- etc.

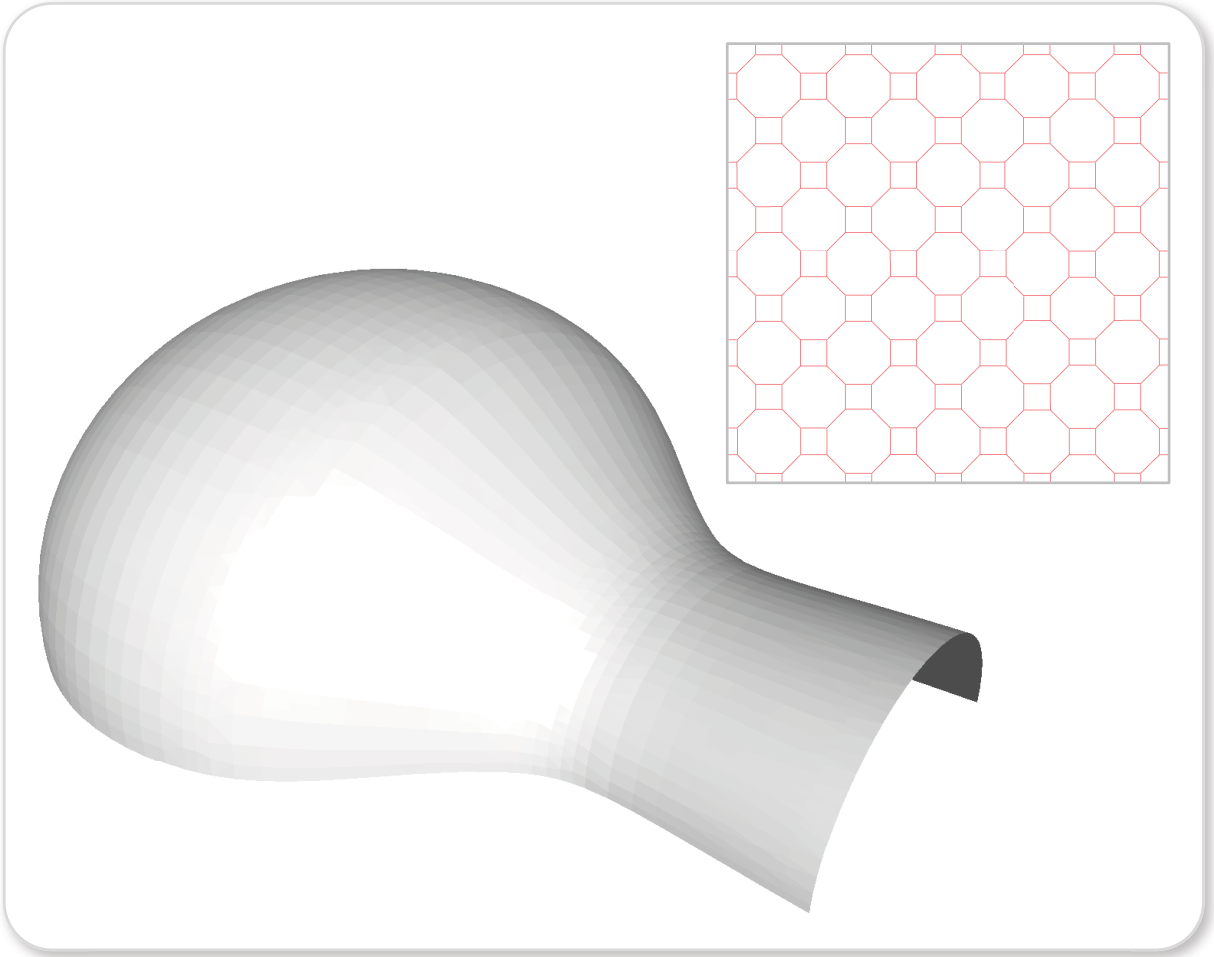
Goal: Interoperability between COMPA and C++

Blender:

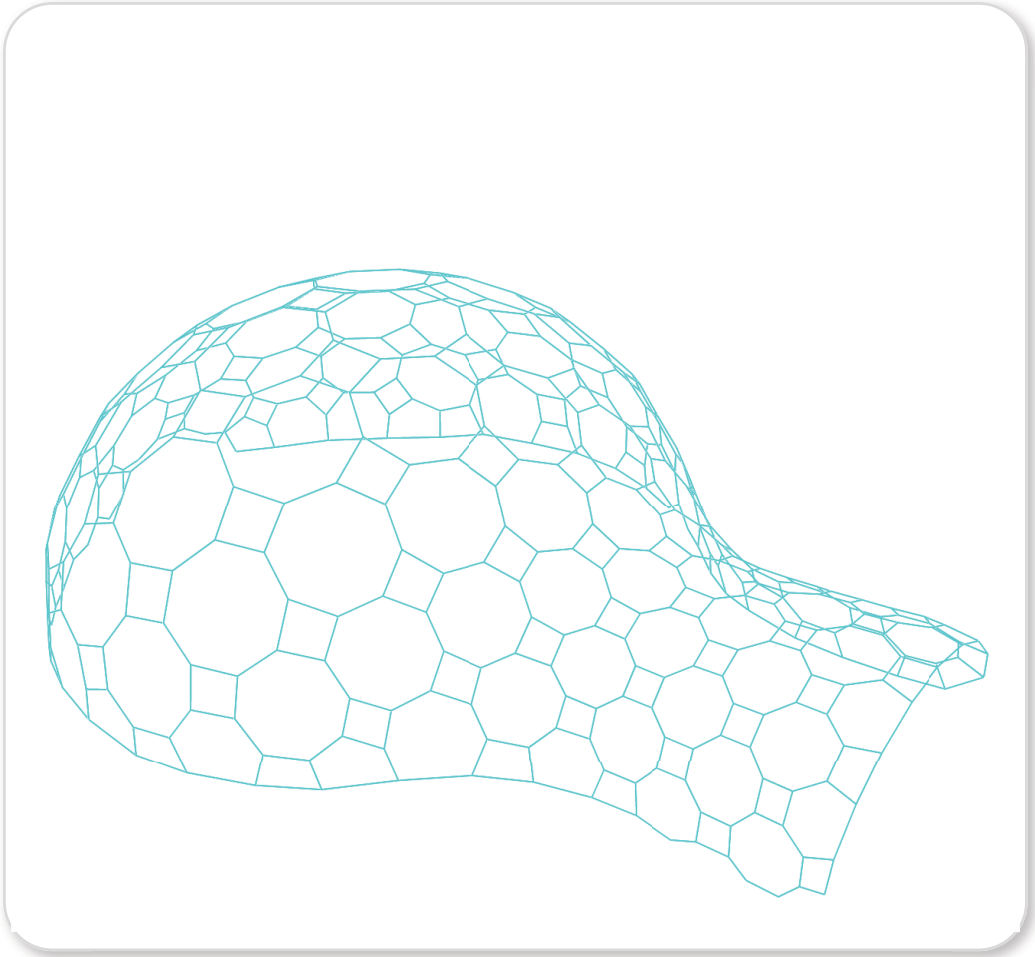
- user interface
- rendering
- Python scripting



Task: Tessellating a 3D Surface



Input: 3D surface + 2D tessellation



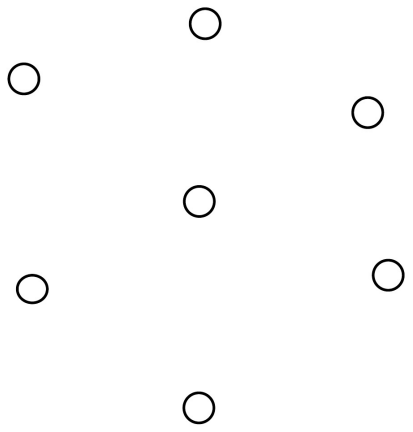
Output: 3D surface tessellation

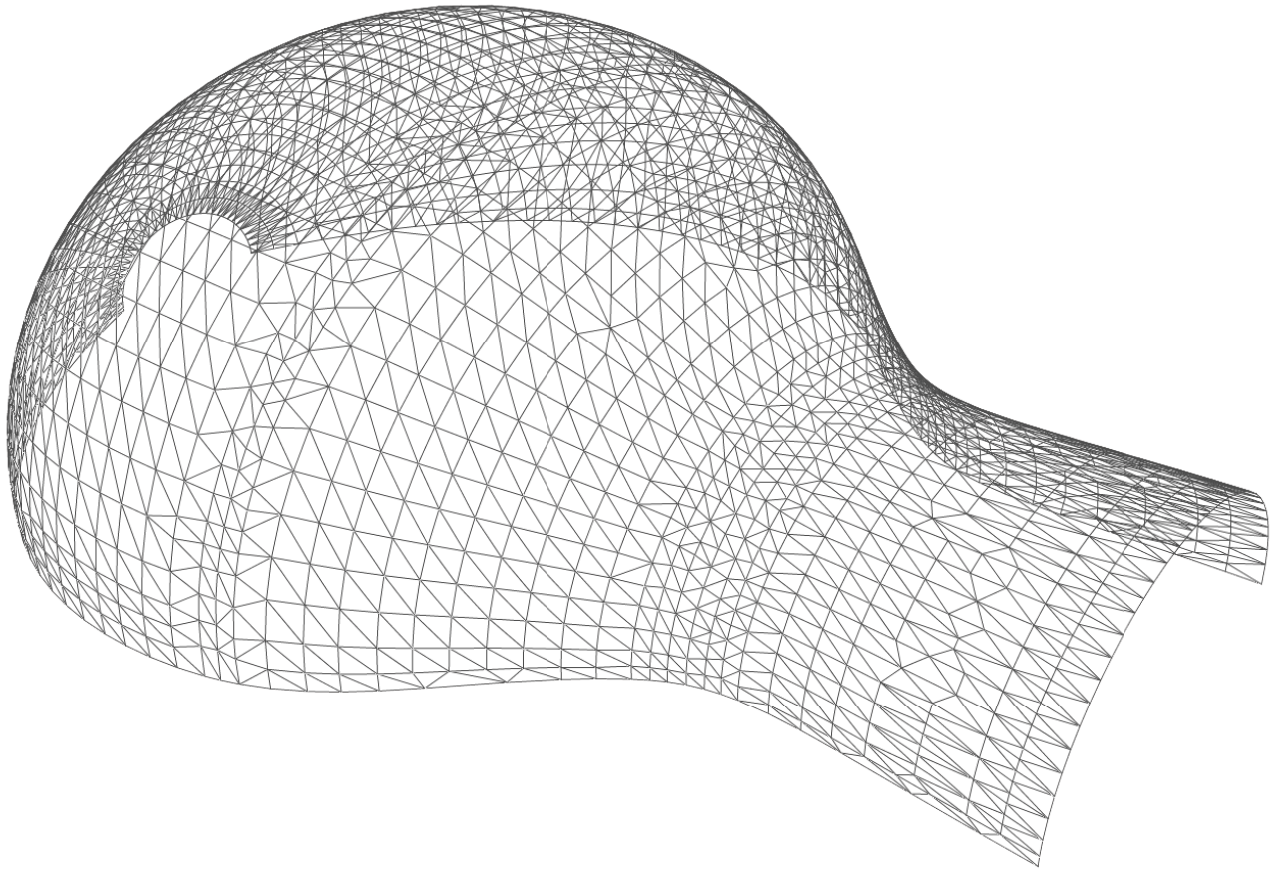
Pipeline: Load 3D Surface

Polygonal Mesh ***M***

$$\mathcal{M} = (\{\mathbf{v}_i\}, \{e_j\}, \{f_k\})$$

$$\textbf{geometry} \;\; \mathbf{v}_i \in \mathbb{R}^3$$





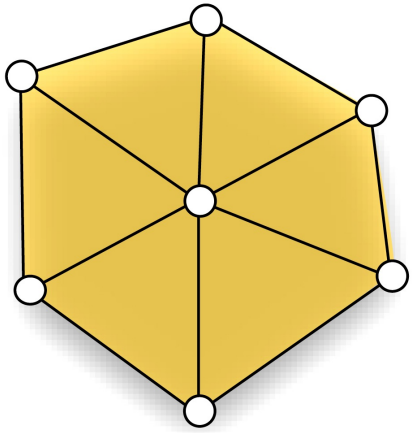
Pipeline: Load 3D Surface

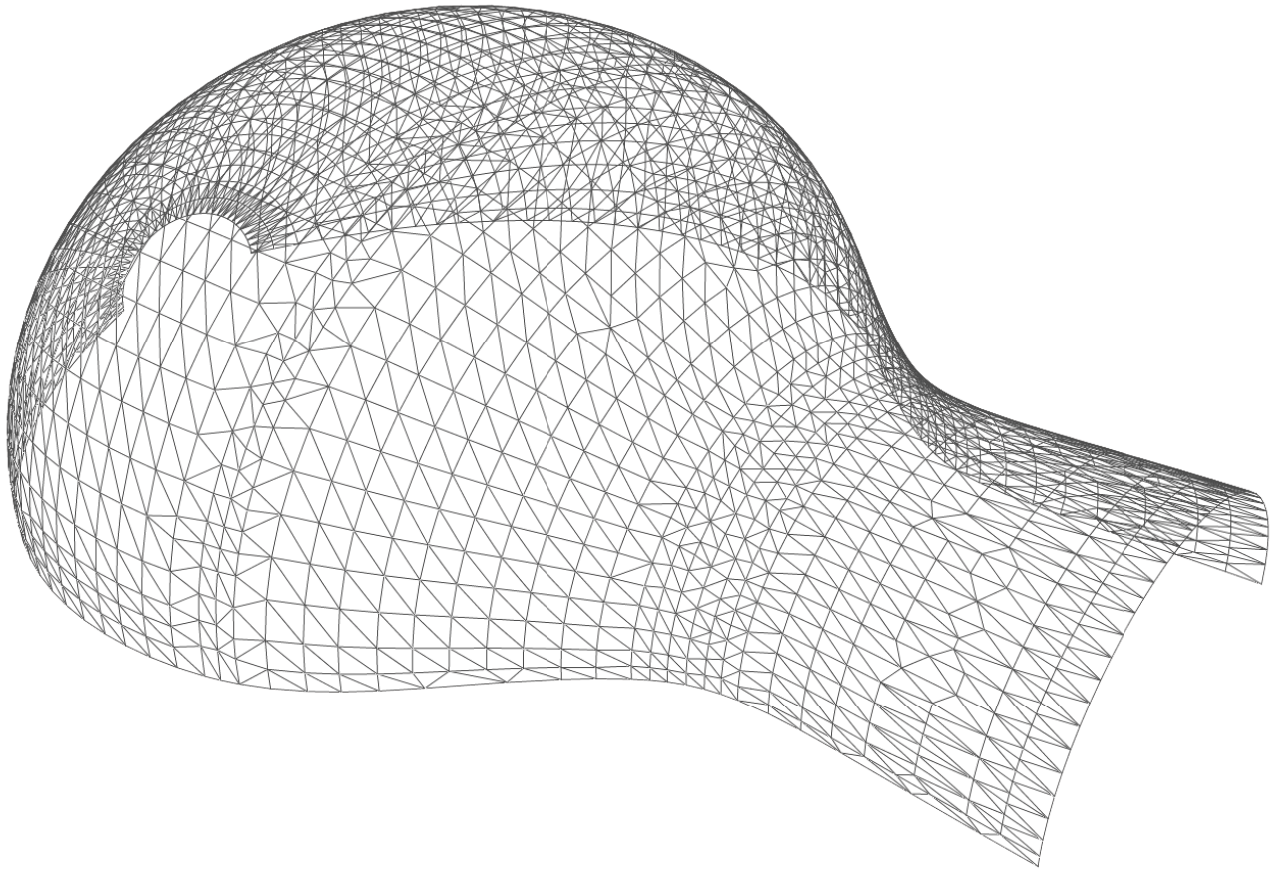
Polygonal Mesh ***M***

$$\mathcal{M} = (\{\mathbf{v}_i\}, \{e_j\}, \{f_k\})$$

geometry $\mathbf{v}_i \in \mathbb{R}^3$

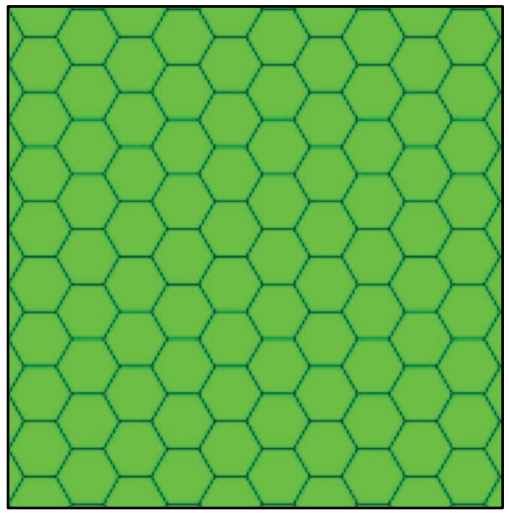
topology $e_i, f_i \subset \mathbb{R}^3$

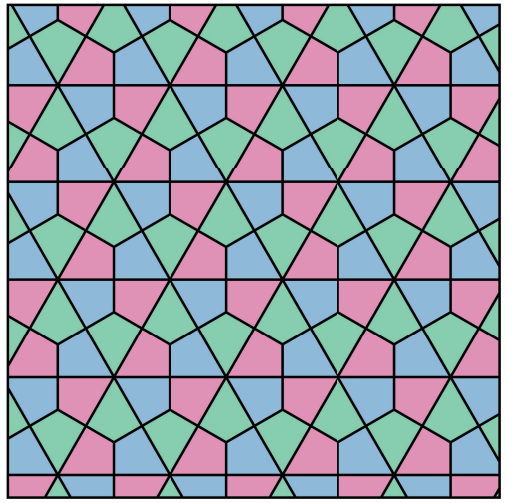


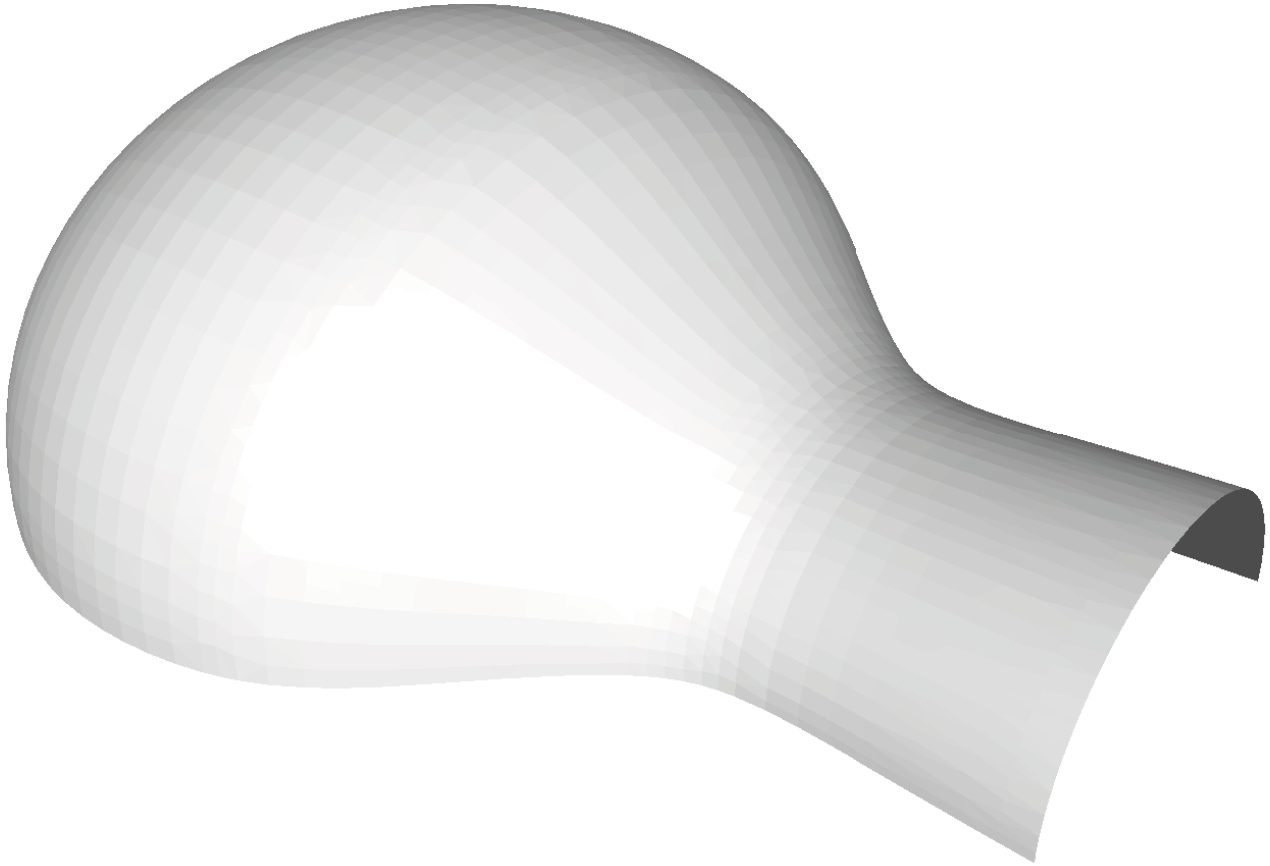


Pipeline: Tessellate 3D Surface

2D tessellation: tiling of a plane using one or more geometric shapes, called tiles, with no overlaps and no gaps

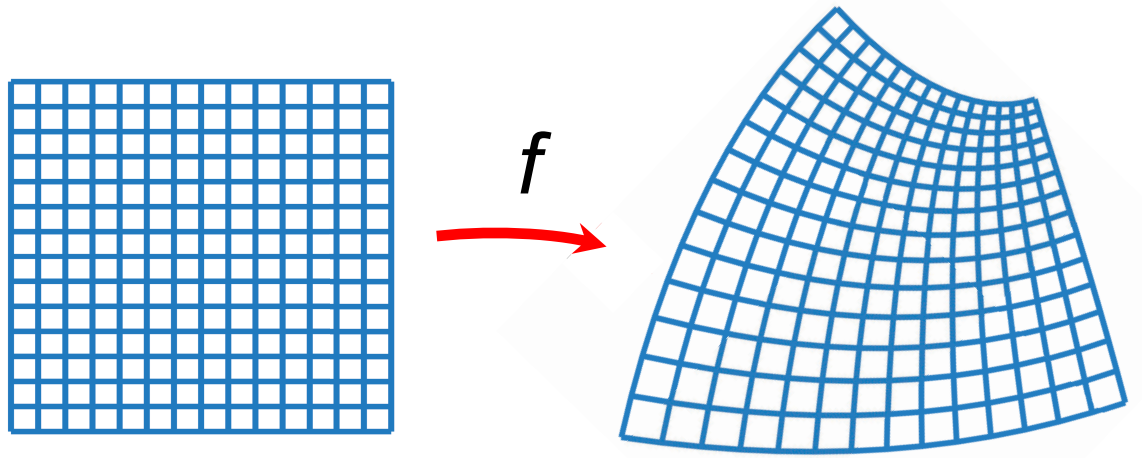


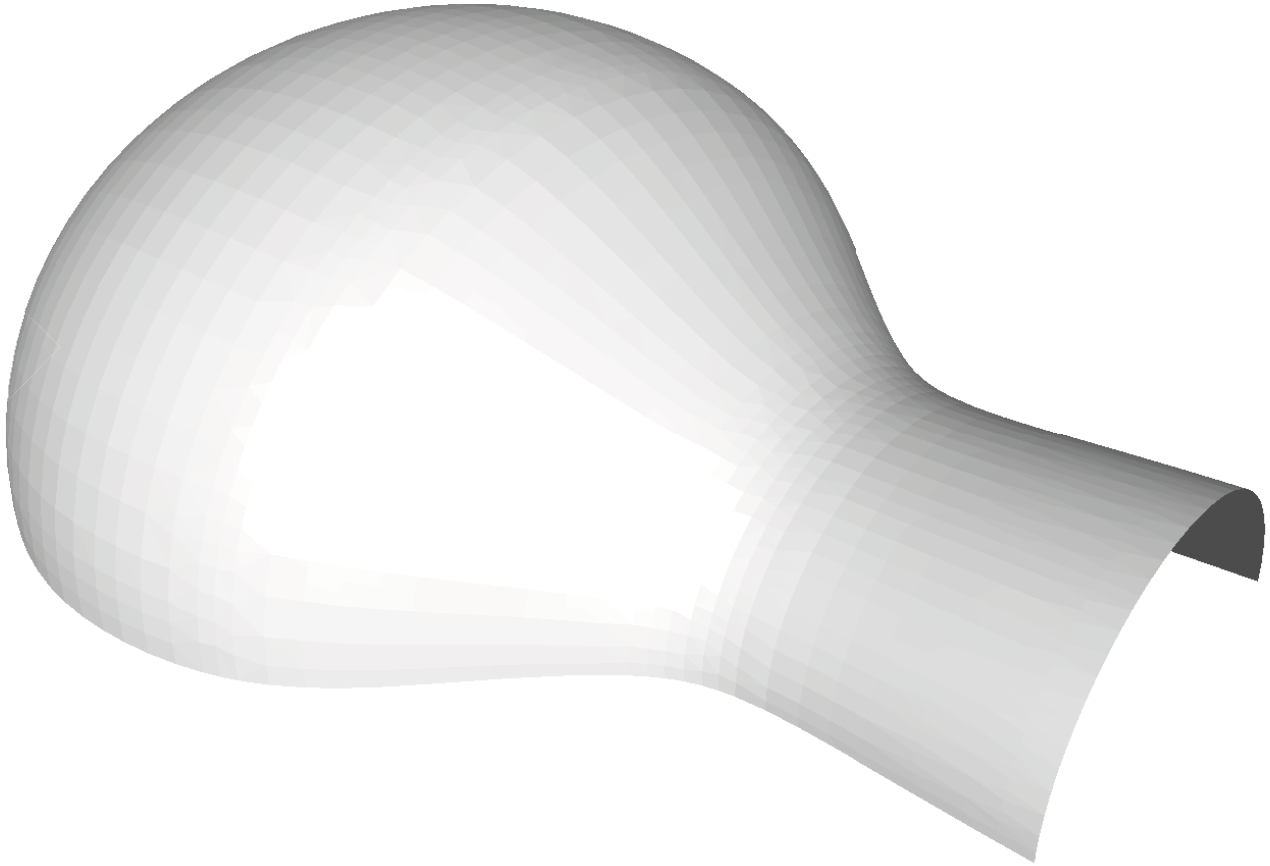




Pipeline: Tessellate 3D Surface

Lift a 2D tessellation onto a 3D surface using *conformal mapping*

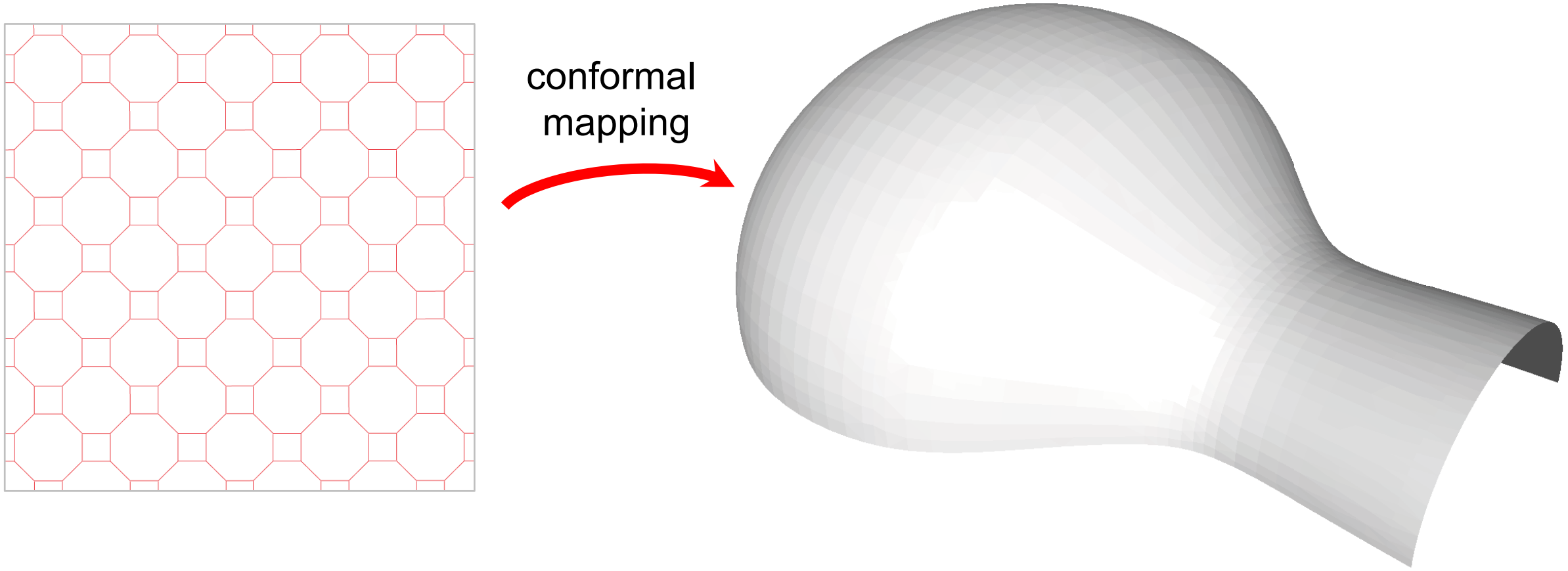




f is a function that preserves *orientation* and *angles* locally, but not necessarily the size.

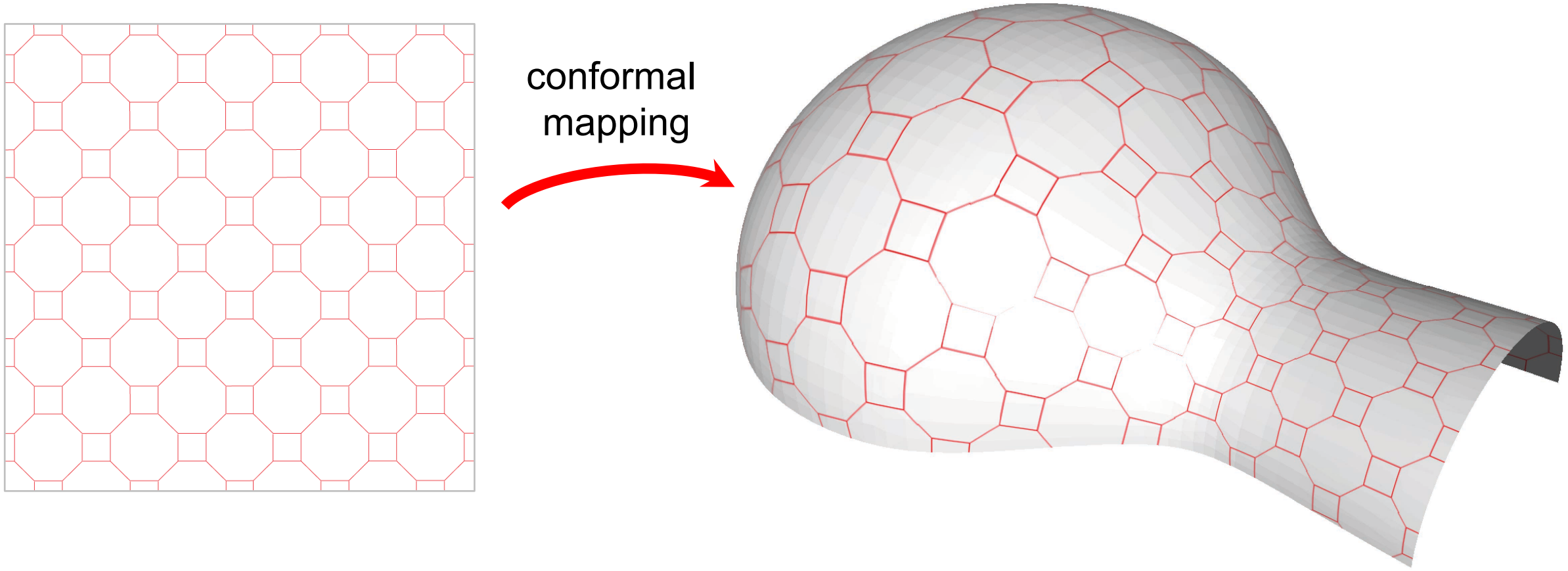
Pipeline: Tessellate 3D Surface

Lift a 2D tessellation onto a 3D surface using *conformal mapping*



Pipeline: Tessellate 3D Surface

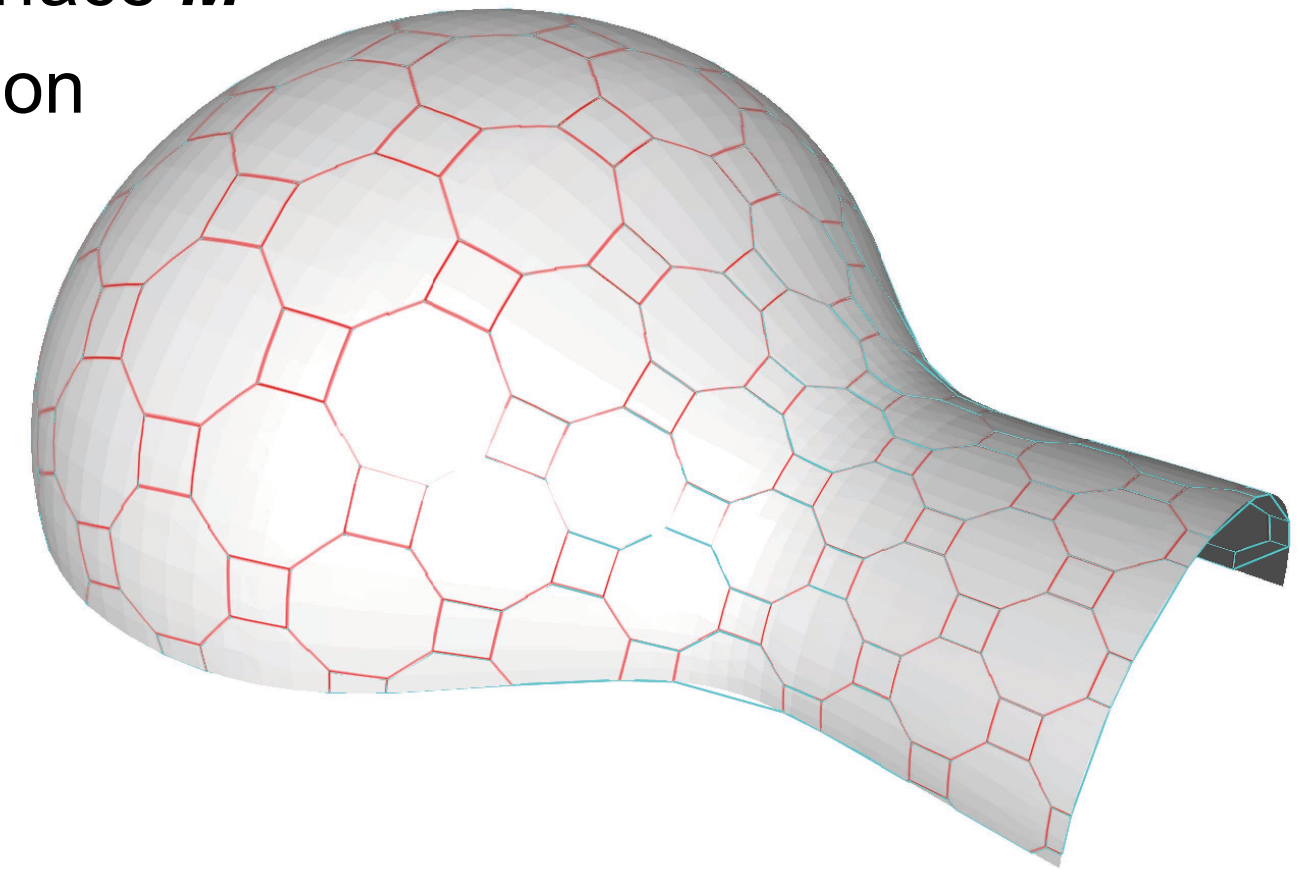
Lift a 2D tessellation onto a 3D surface using *conformal mapping*



Pipeline: Compute 3D Surface Tessellation

3D surface tessellation ***T***

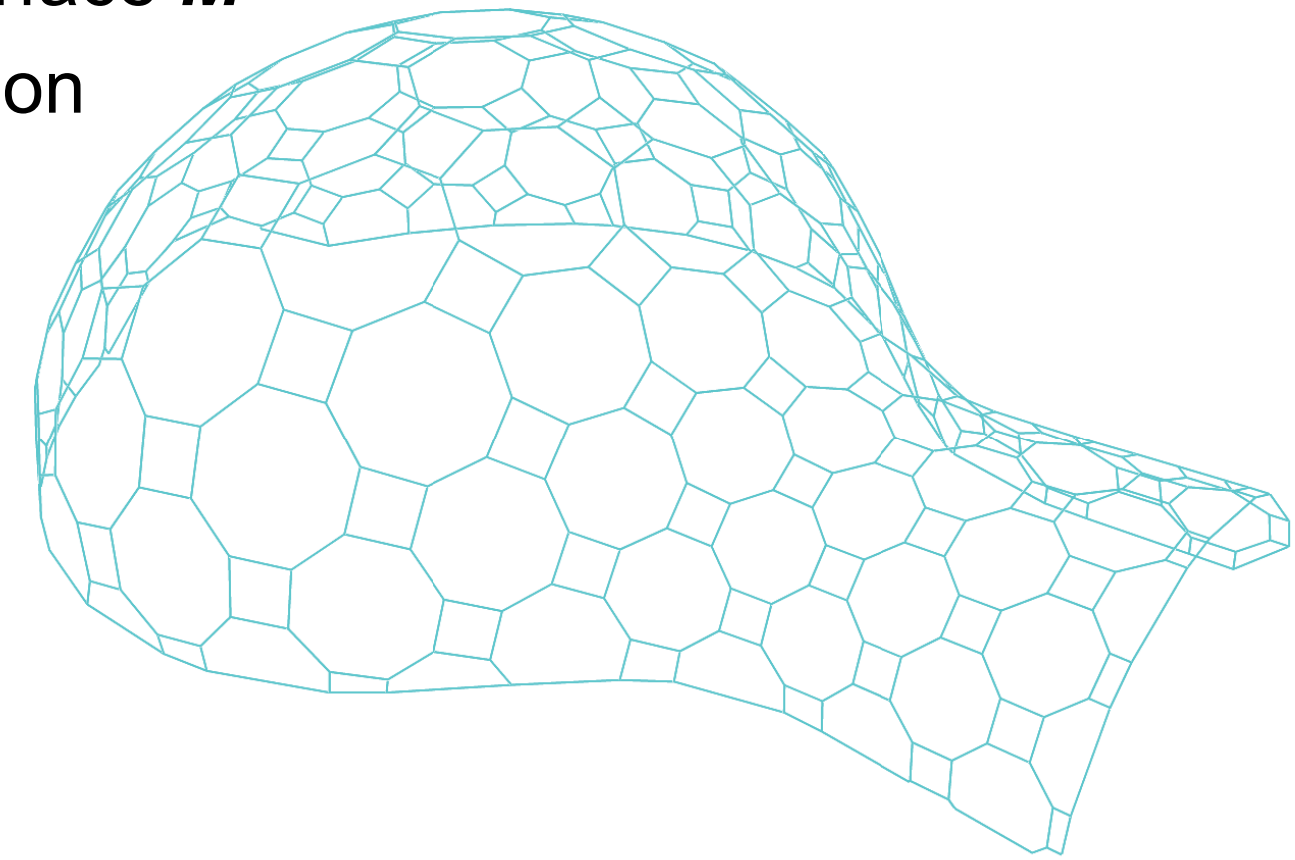
- *Geometry:* compute from 3D surface ***M***
- *Topology:* same as 2D tessellation



Pipeline: Generate 3D Surface Tessellation

3D surface tessellation ***T***

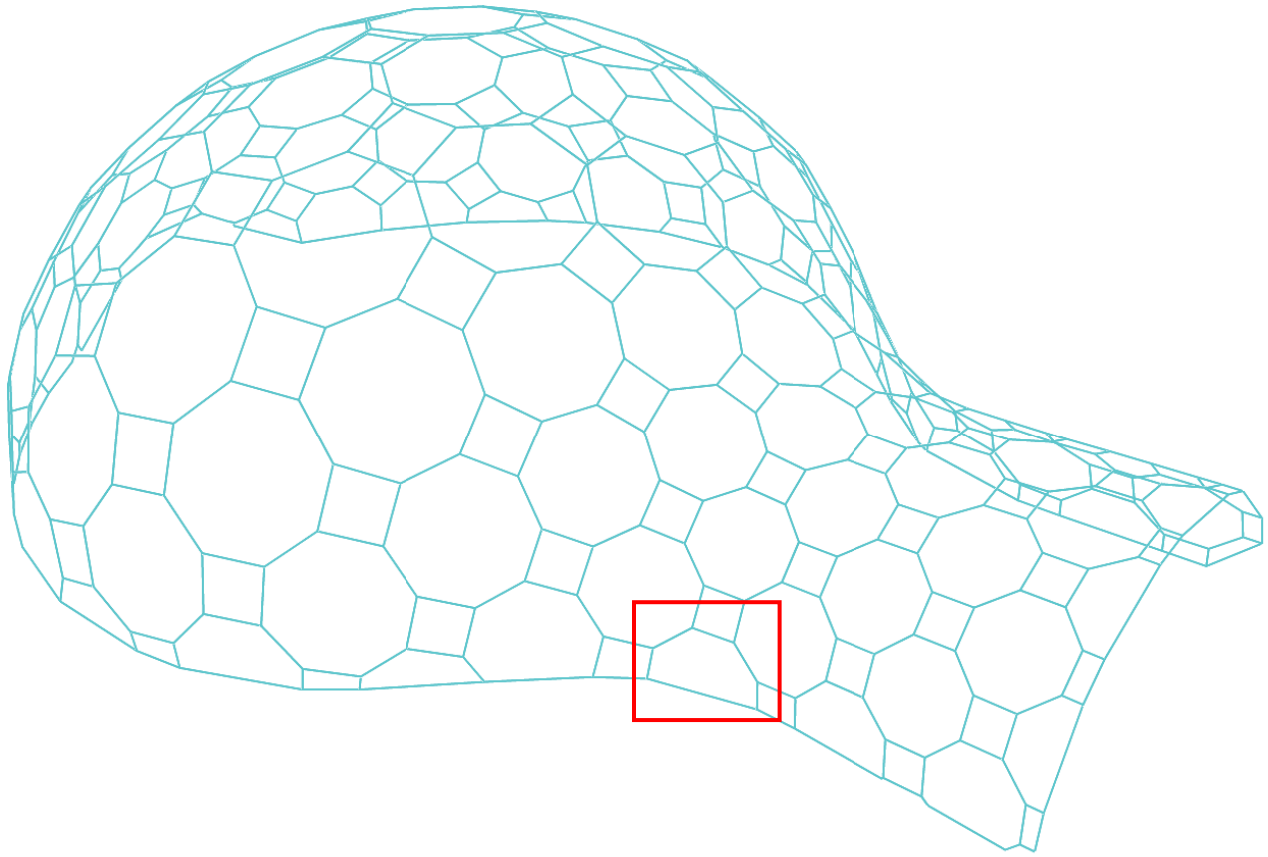
- *Geometry:* compute from 3D surface ***M***
- *Topology:* same as 2D tessellation



Pipeline: Optimize 3D Surface Tessellation

3D surface tessellation ***T***

- non-planar tiles
- non-regular tiles



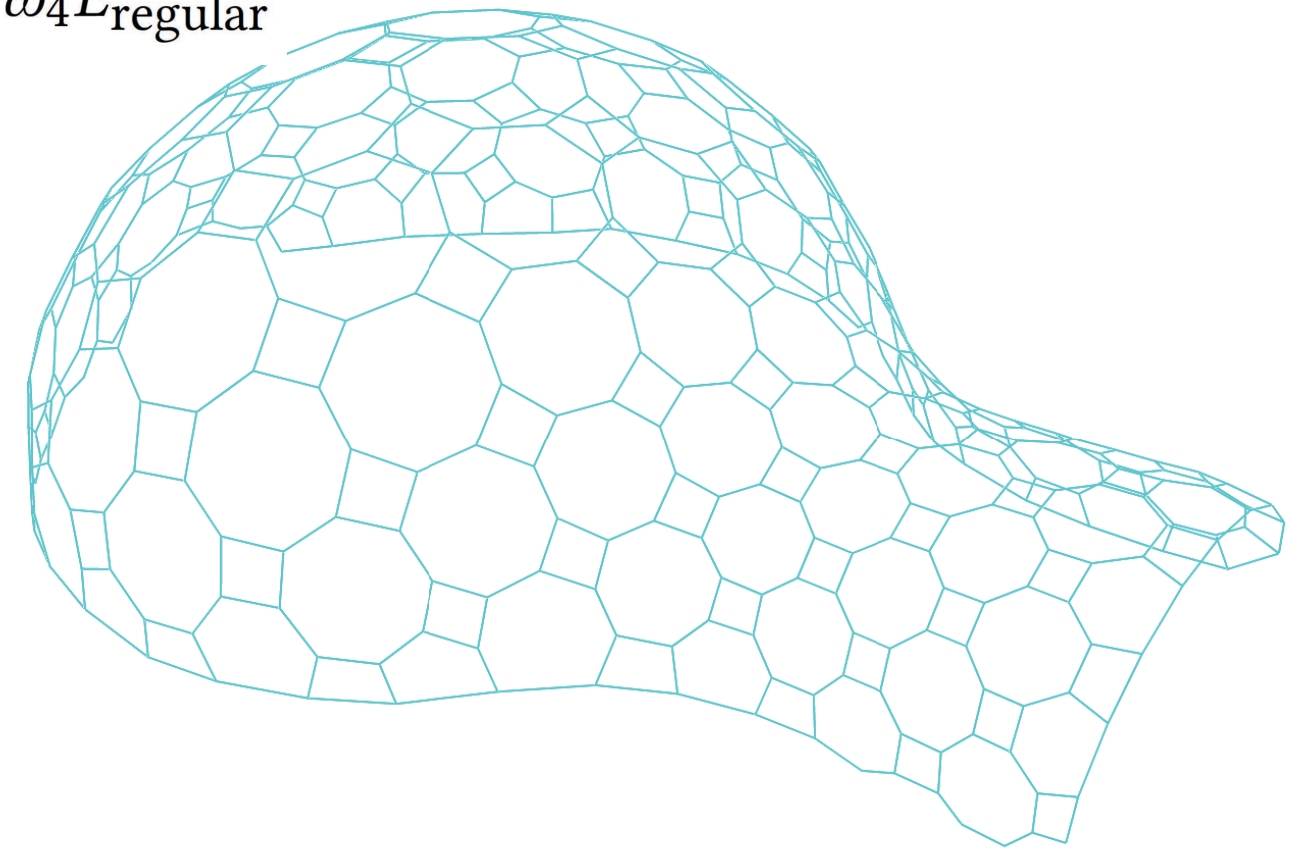
Pipeline: Optimize 3D Surface Tessellation

Optimize ***T*** by minimizing

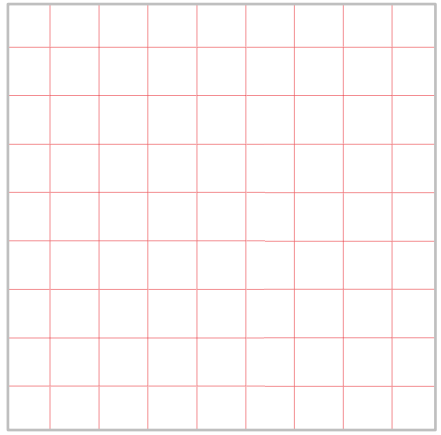
$$E = \omega_1 E_{\text{surf}} + \omega_2 E_{\text{bound}} + \omega_3 E_{\text{planar}} + \omega_4 E_{\text{regular}}$$

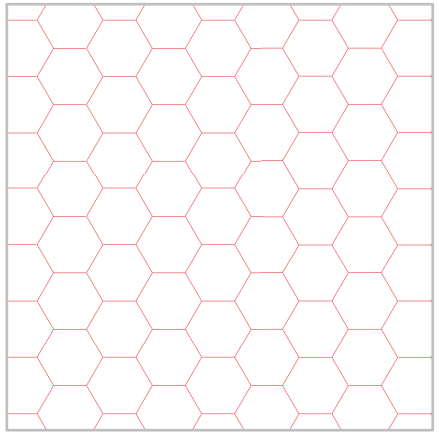
where

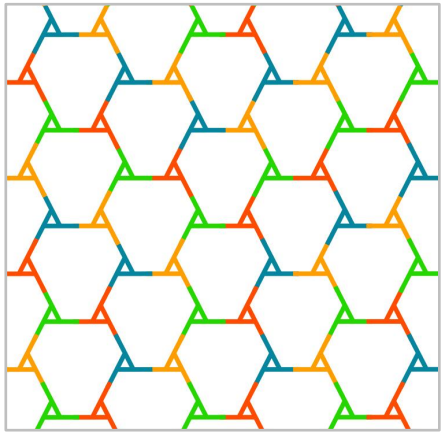
E_{surf}	***T*** resemble the surface of ***M***
E_{bound}	***T*** resemble the boundary of ***M***
E_{planar}	make polygons in ***T*** planar
E_{regular}	make polygons in ***T*** regular

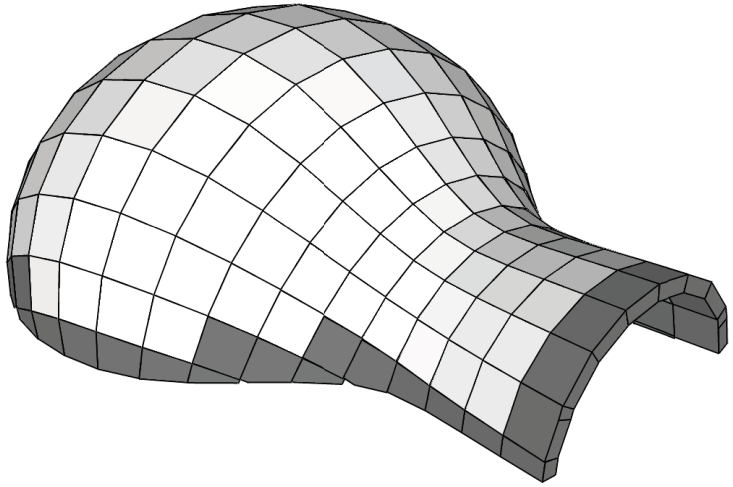


Application: Architecture Design

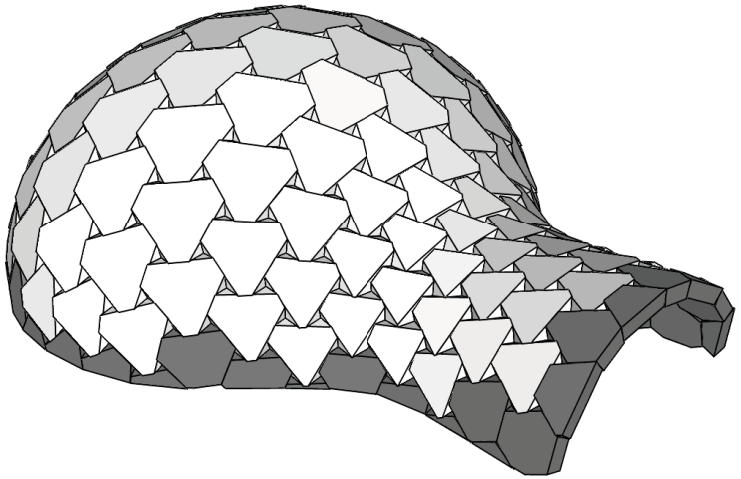




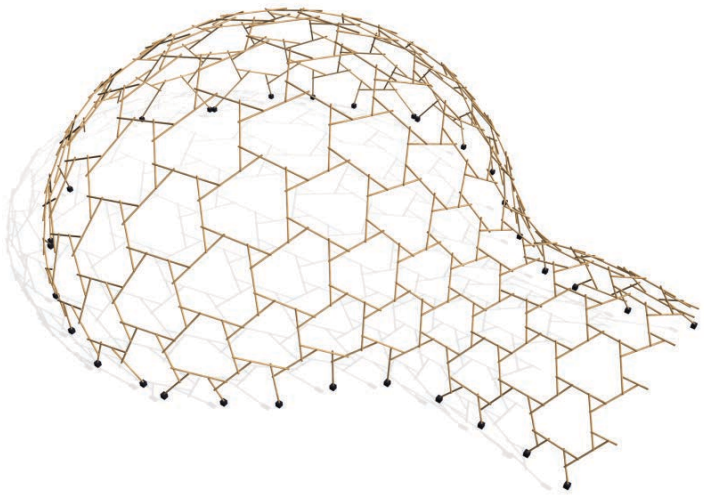




Self-supporting Structure

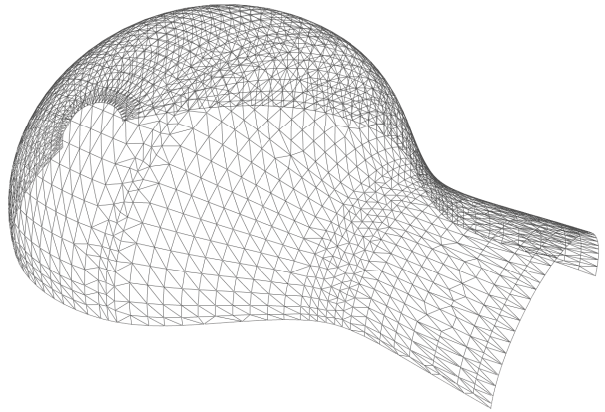


Topological Interlocking Assembly

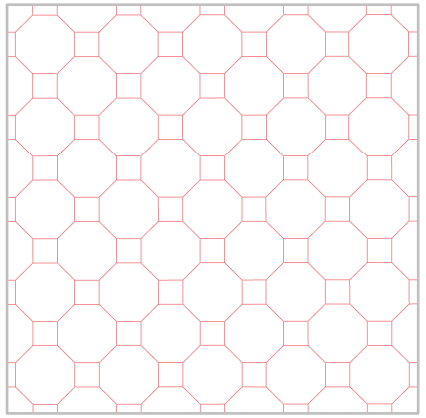


Reciprocal frames

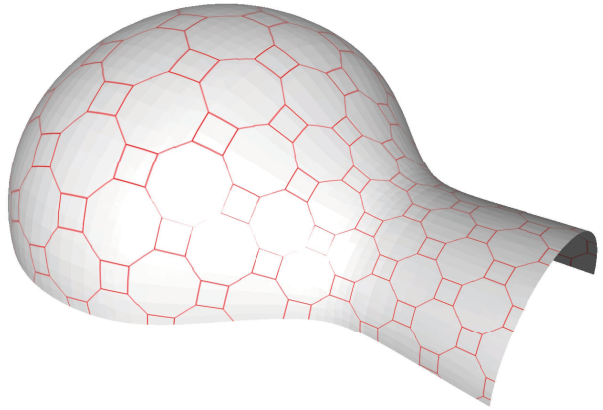
Overview



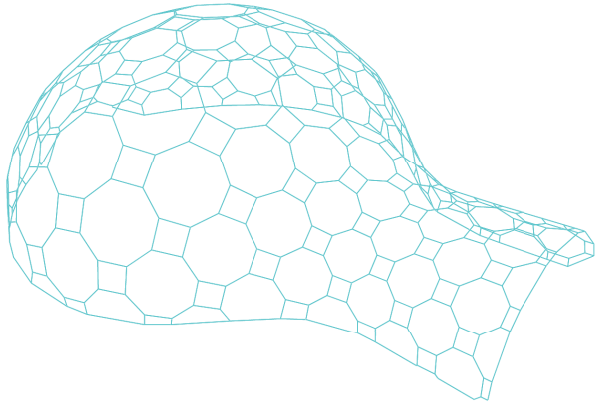
Load 3D mesh



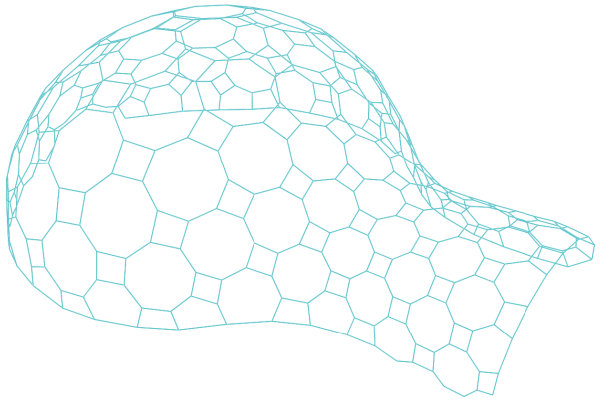
Create 2D tessellation



Conformal mapping

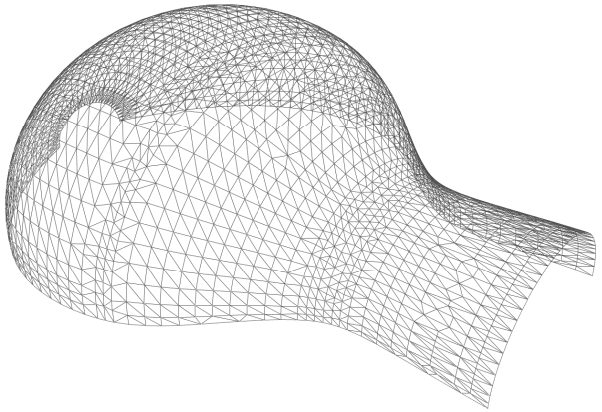


Generate ***T***

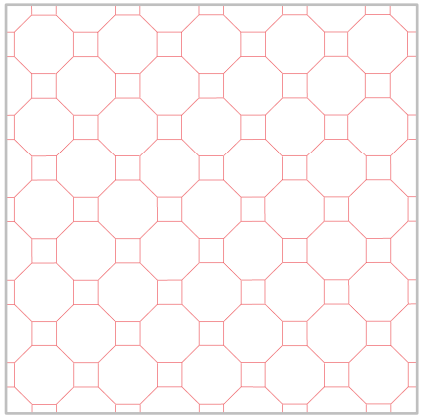


Optimize ***T***

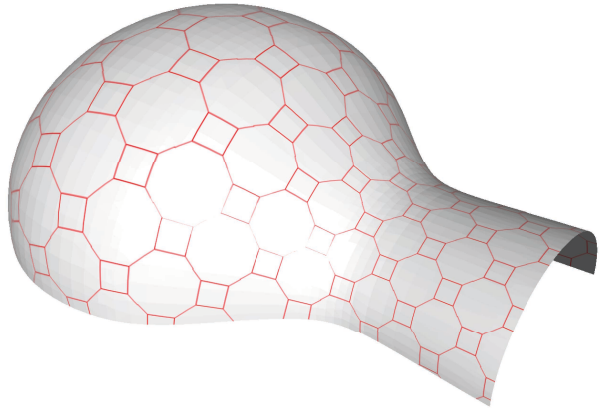
Overview



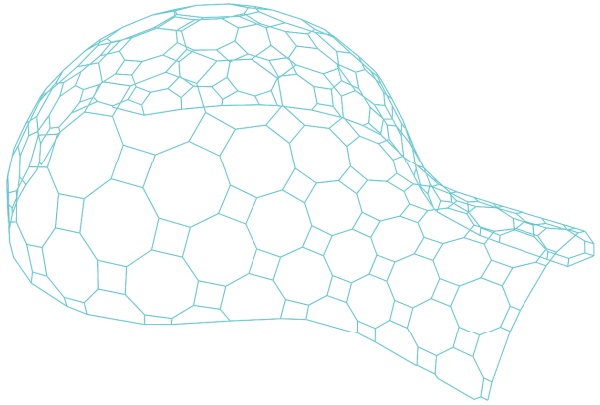
Load 3D mesh
(COMPAS: Mesh)



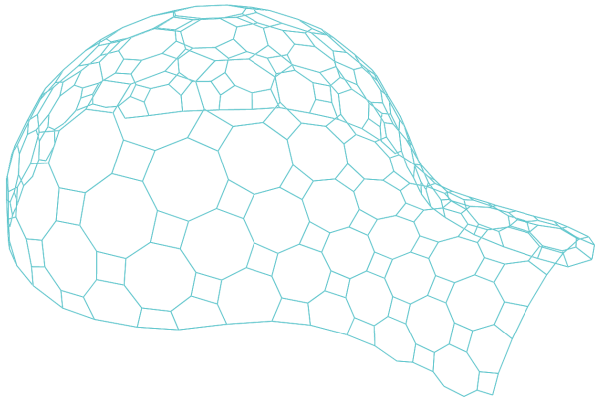
Create 2D tessellation
(Python: tessgon)



Conformal mapping
(C++, libigl: lscm)

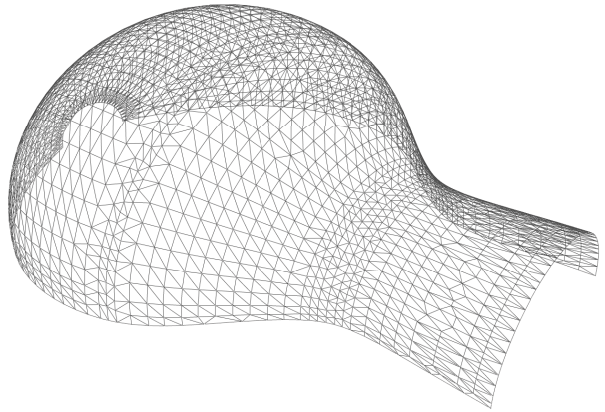


Generate ***T***
(C++, libigl: AABB, Baycentric)

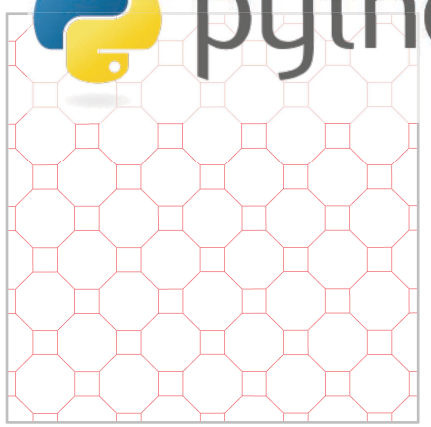


Optimize ***T***
(C++: ShapeOp)

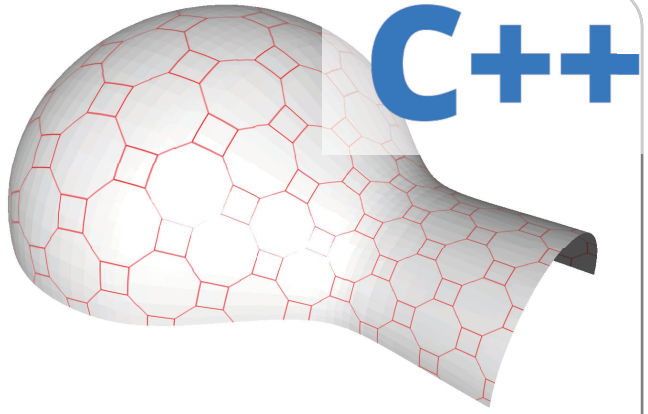
Overview



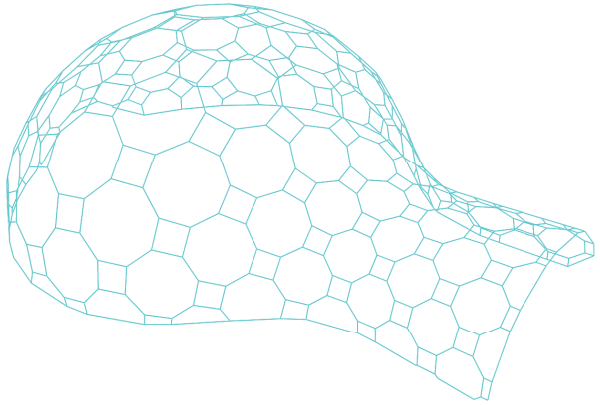
Load 3D mesh
(COMPAS: Mesh)



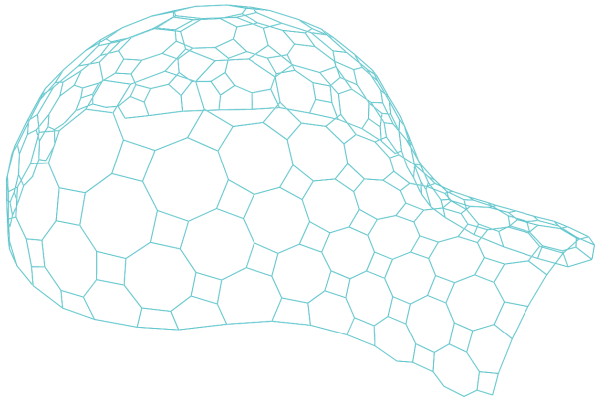
Create 2D tessellation
(Python: tessgon)



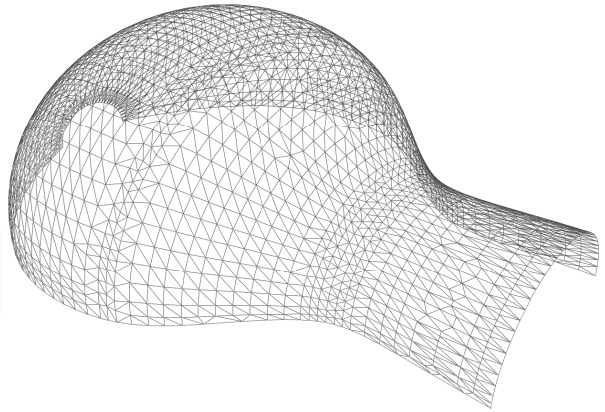
Conformal mapping
(C++, libigl: lscm)



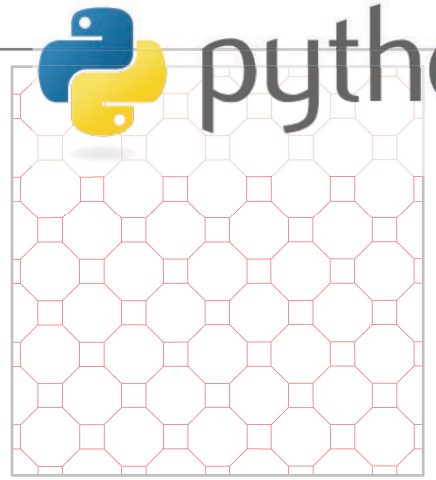
Generate ***T***
(C++, libigl: AABB, Baycentric)



Optimize ***T***
(C++: ShapeOp)

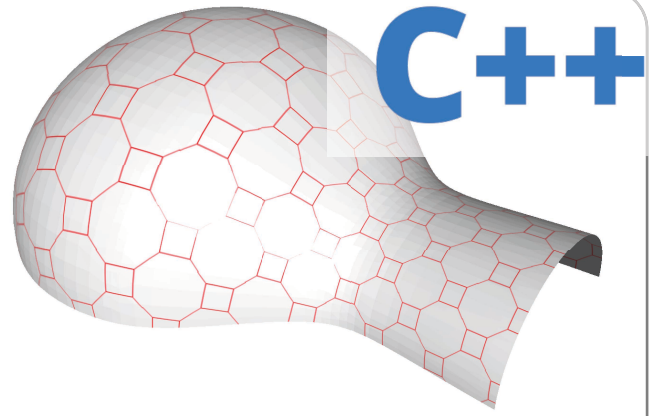


Load 3D mesh
(COMPAS: Mesh)

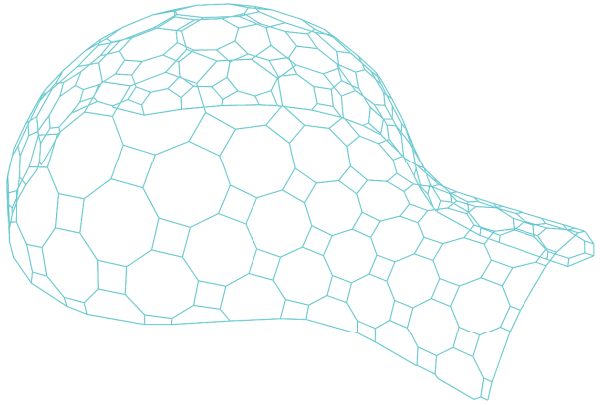


Create 2D tessellation
(Python: tessgon)

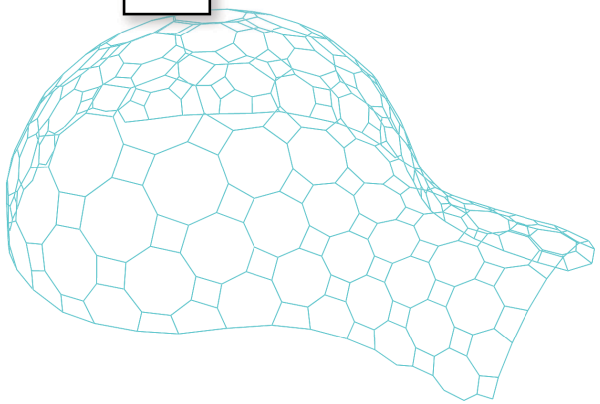
pybind11



Conformal mapping
(C++, libigl: lscm)



Generate ***T***
(C++, libigl: AABB, Baycentric)



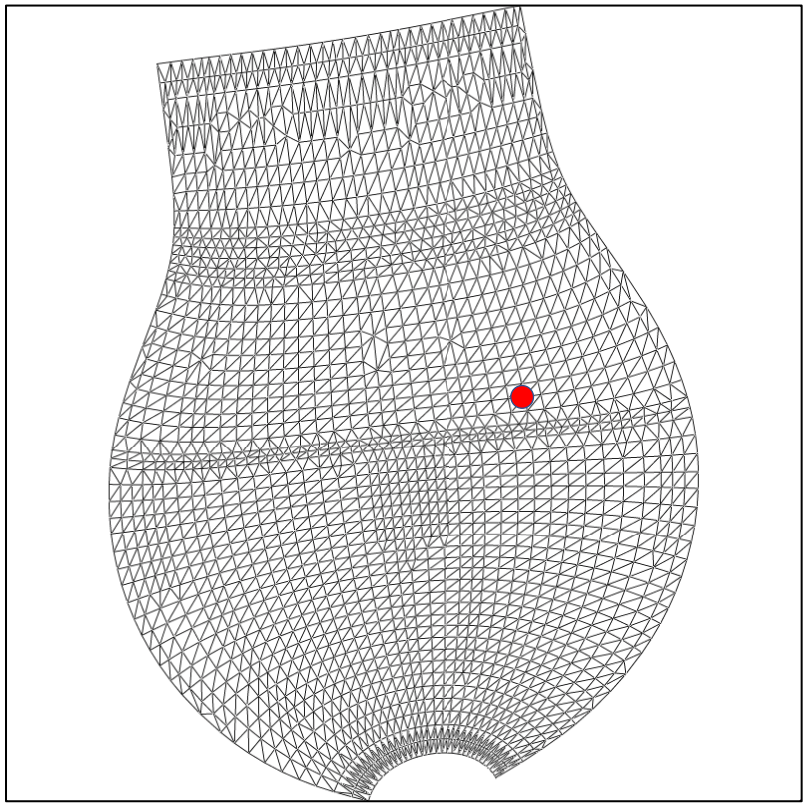
Optimize ***T***
(C++: ShapOp)

Thank You!

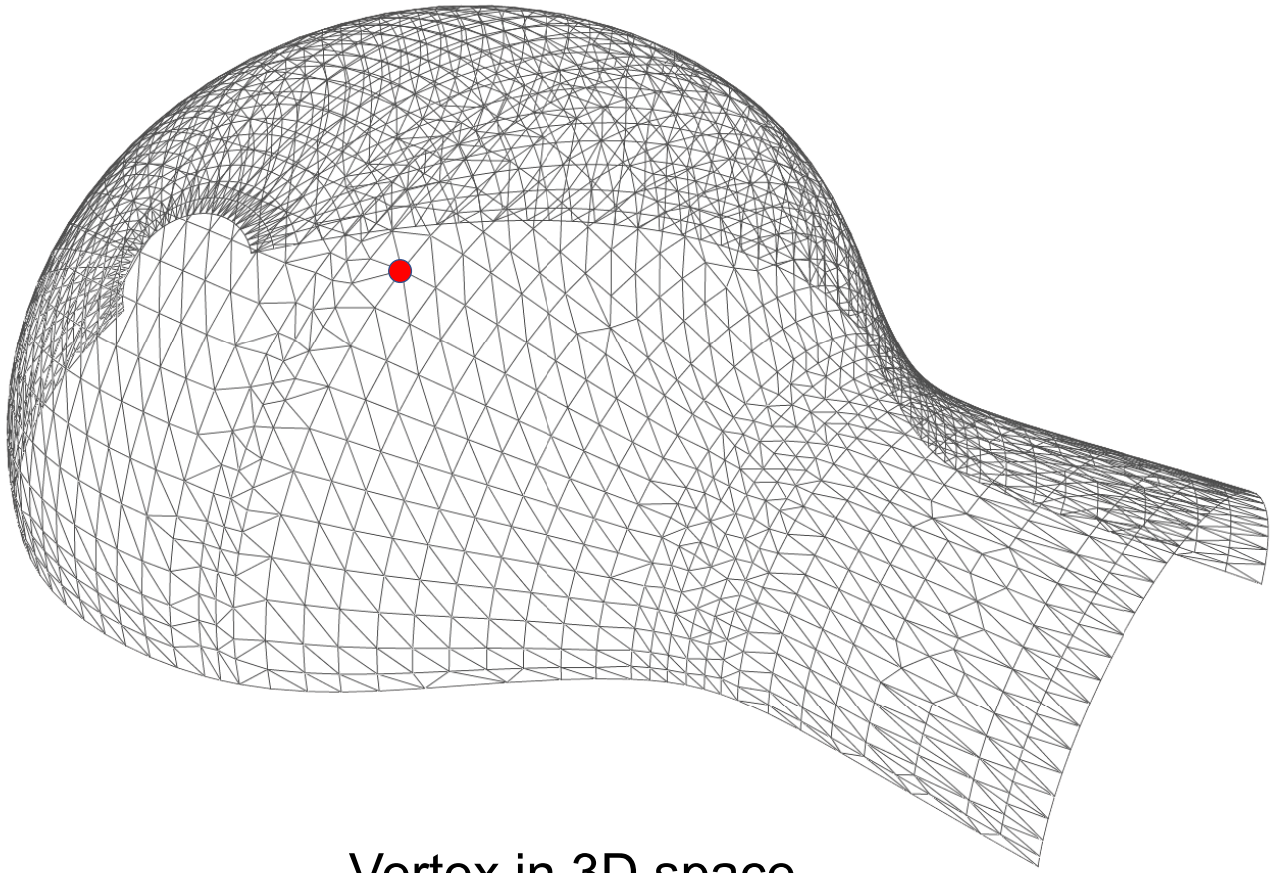
Backup Slides

Pipeline: Load 3D Surface

Conformal map of ***M*** can be saved as texture coordinates associated with each vertex of **M**



Vertex in 2D texture coordinate



Vertex in 3D space