Computer Graphics 1 - Introduction, object representations

Tom Thorne

thomas.thorne@ed.ac.uk www.inf.ed.ac.uk/teaching/courses/cg/

What is in the course?

Graphics algorithms, data structures and rendering

- Object representations
- Lighting and shading
- 3D transformations
- OpenGL
- Rasterisation

Photorealistic rendering

- Ray tracing
- Monte Carlo methods
- Photon mapping

Curves and curved surfaces

Graphics pipeline

Geometry

- Transformation
- Perspective projection
- Hidden surface removal

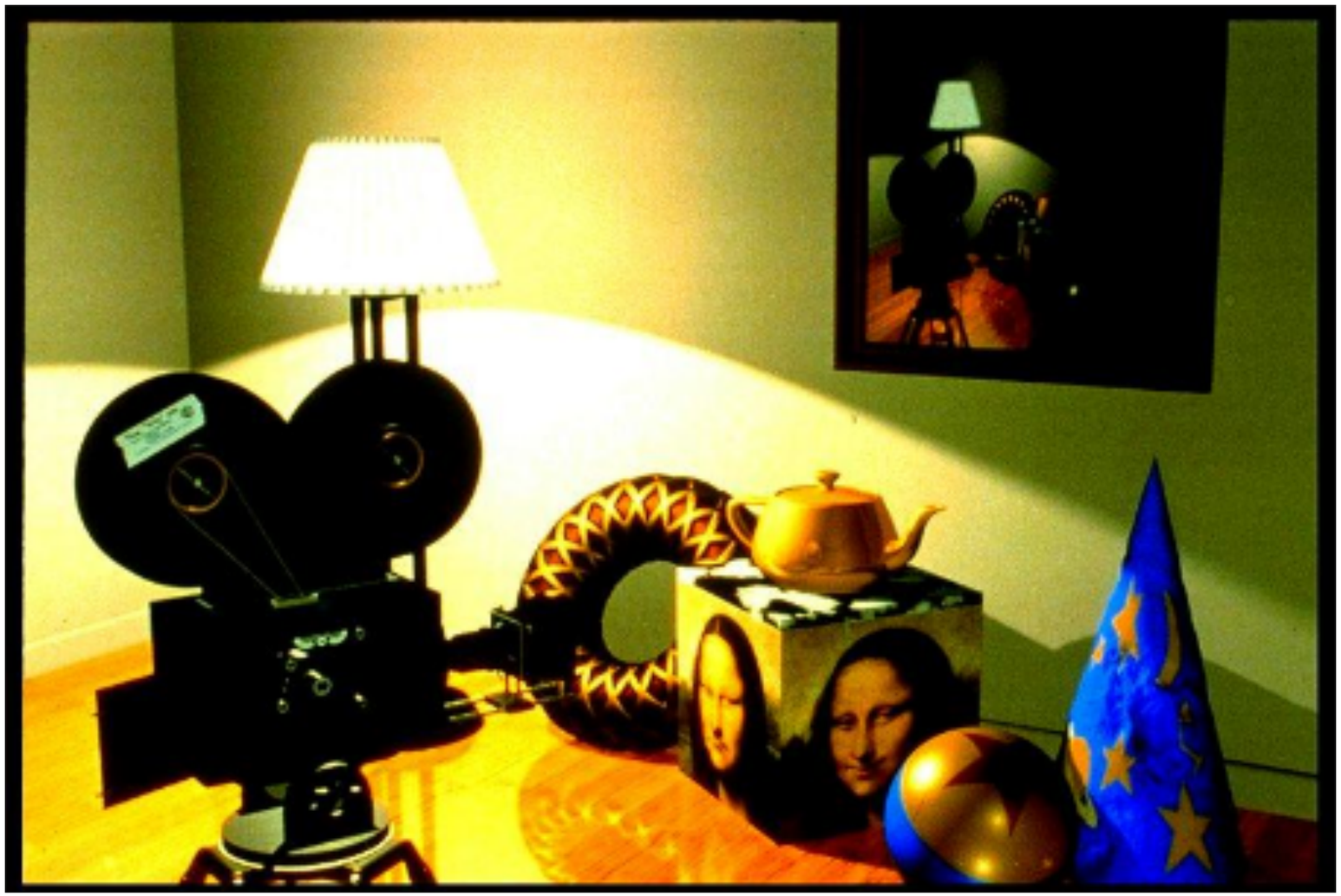
Shading and lighting

- Reflections
- Shadows

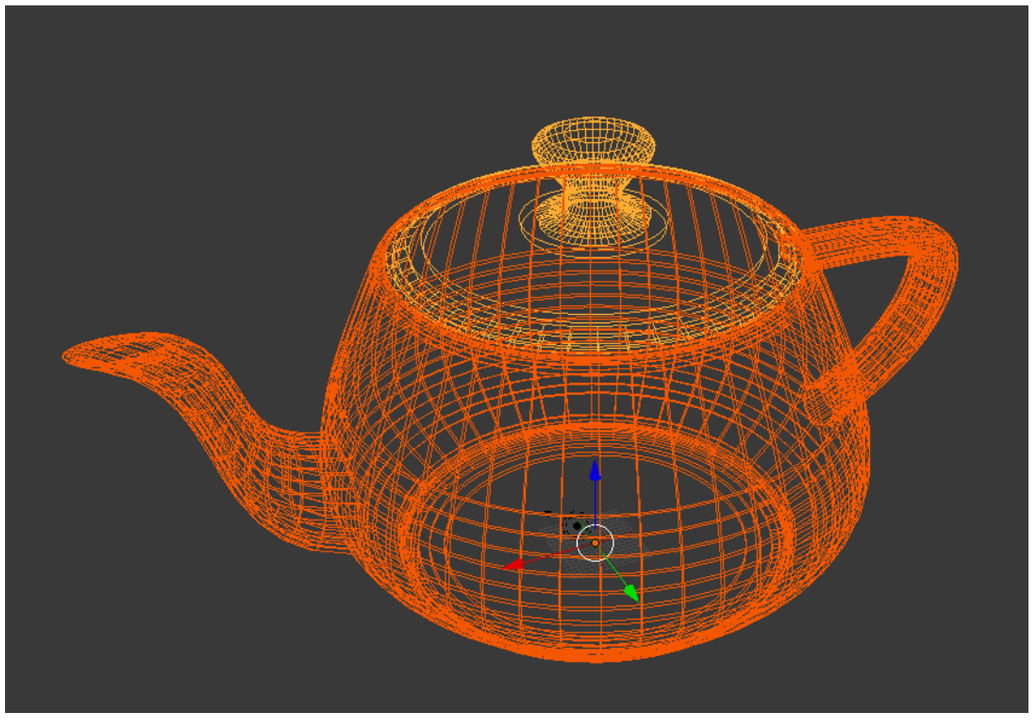
Rasterisation

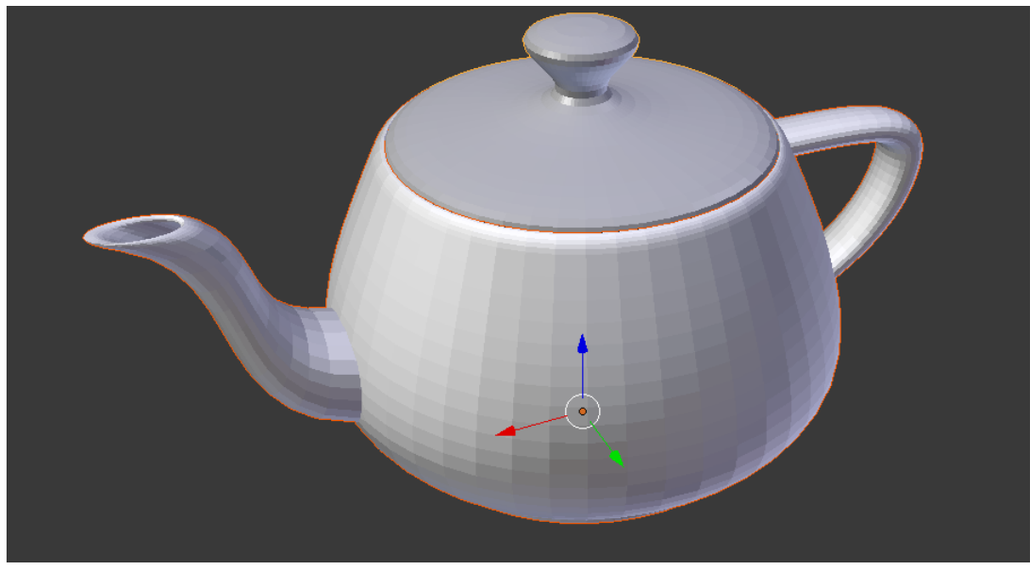
- Anti aliasing
- Texture mapping
- Bump mapping
- Ambient occlusion

Example scene



Object models





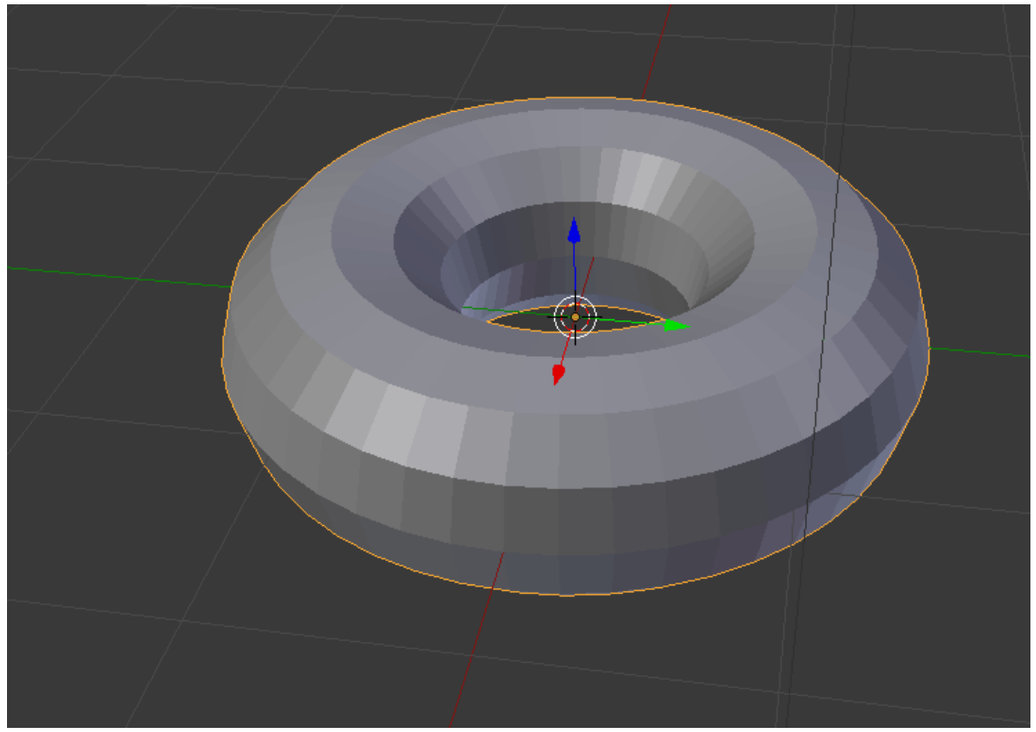
Geometry stage

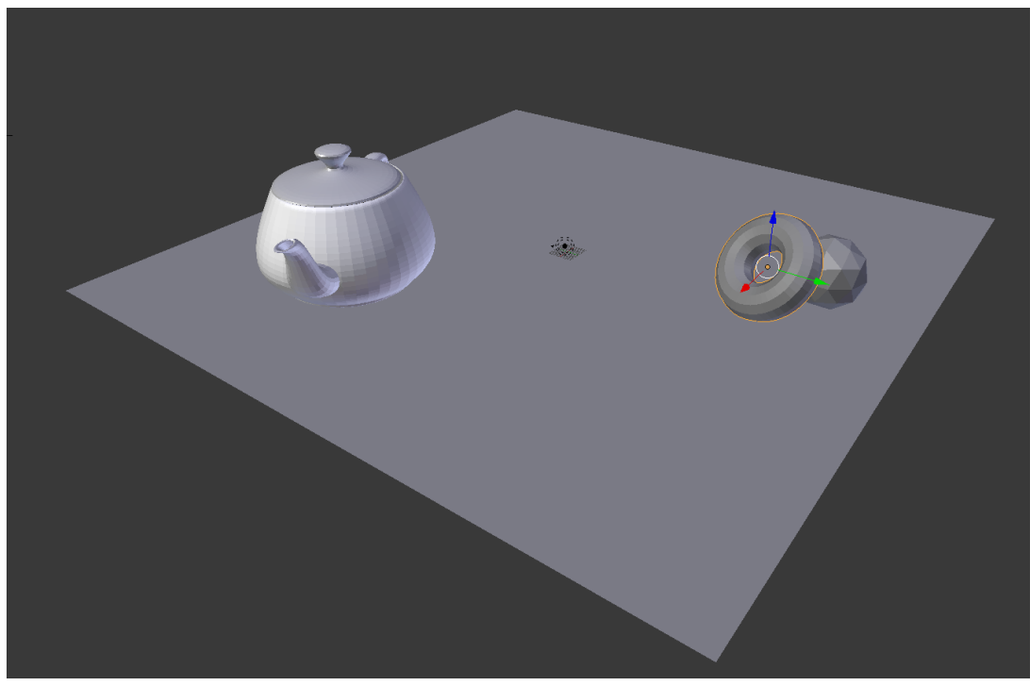
- Transforming objects
- View transformation
- Illumination and shading (for vertices)

Object transformation (lecture 3)

Placing objects in the scene

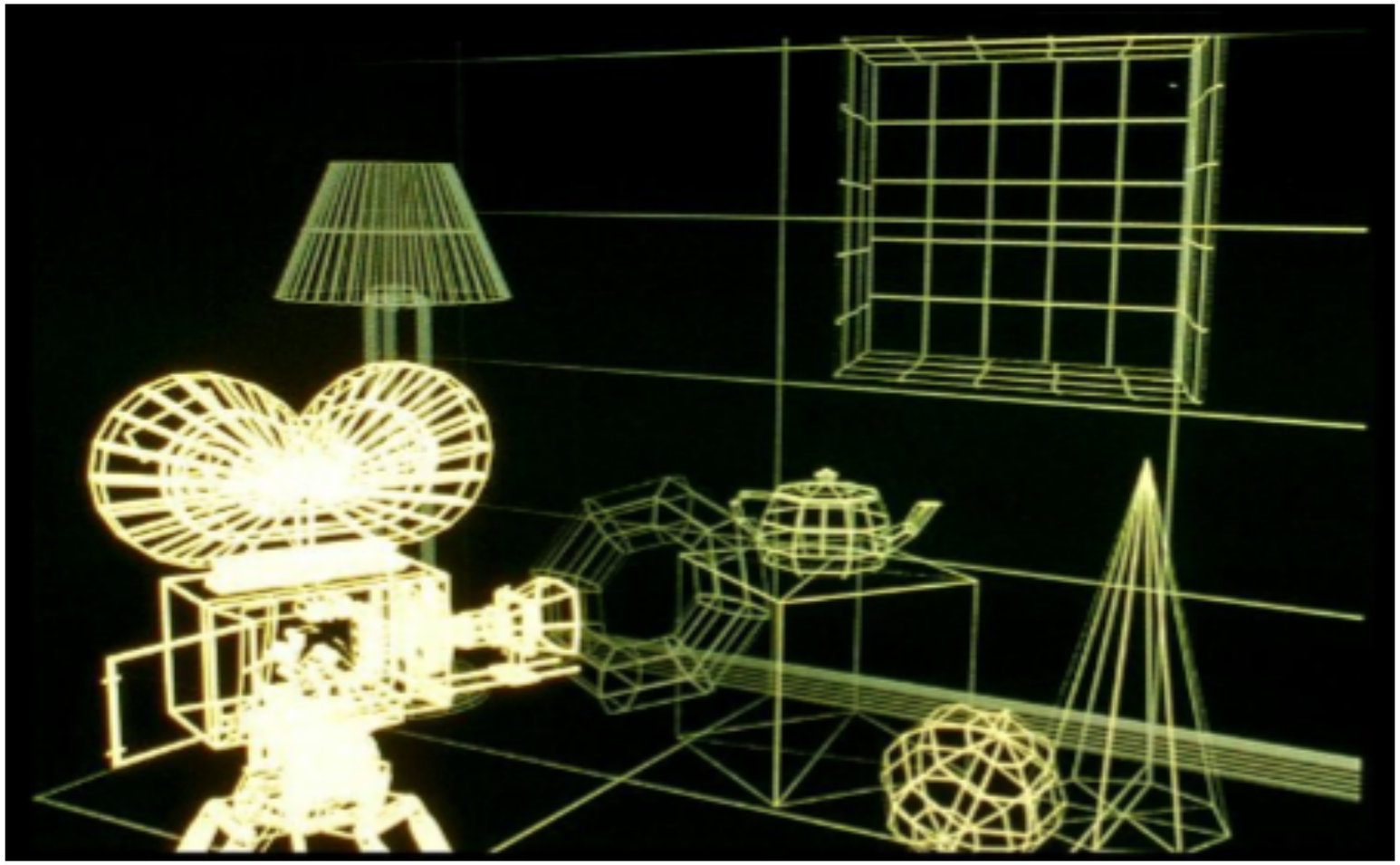
- Rotation
- Scaling
- Translation



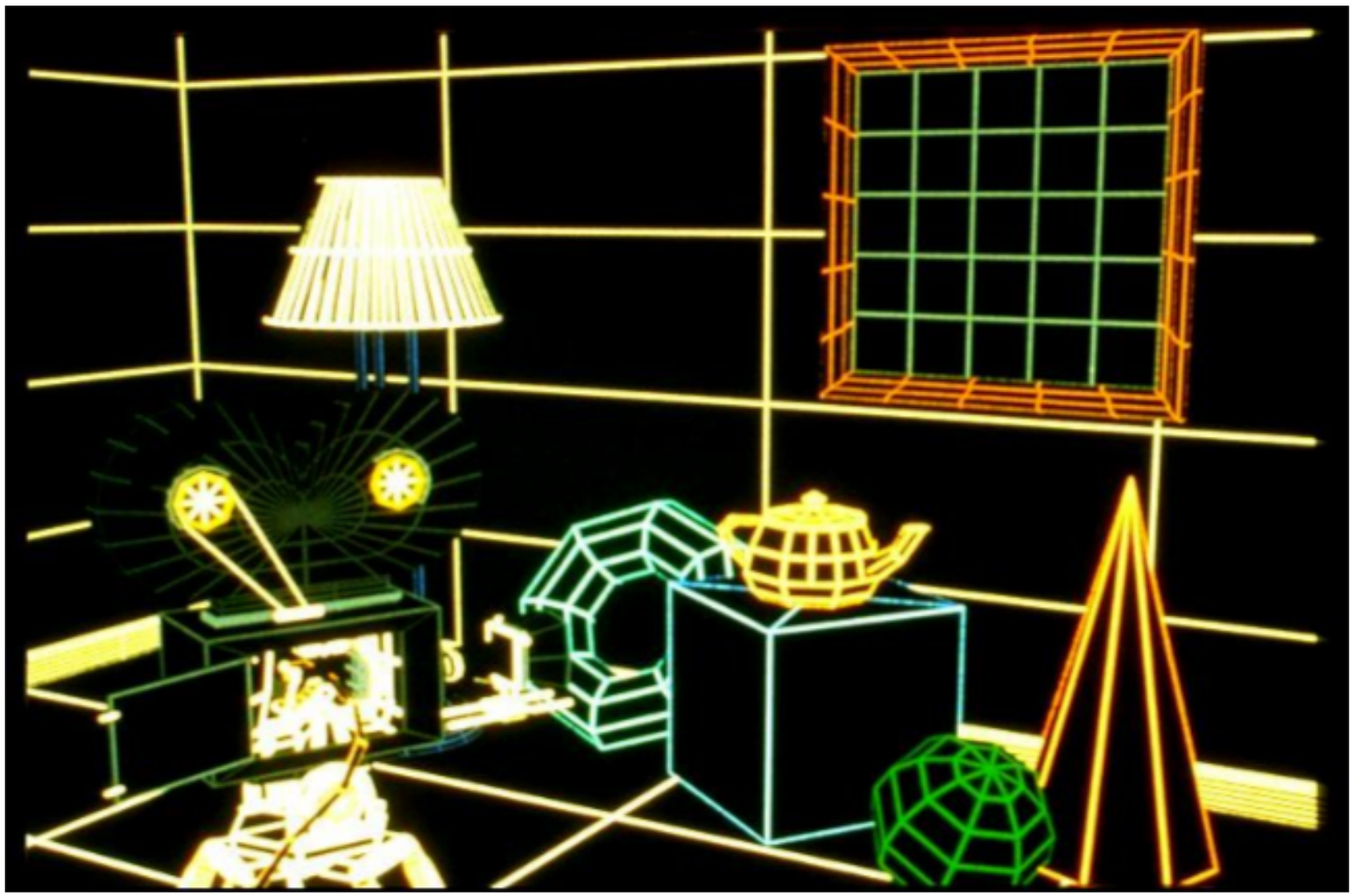


Perspective projection

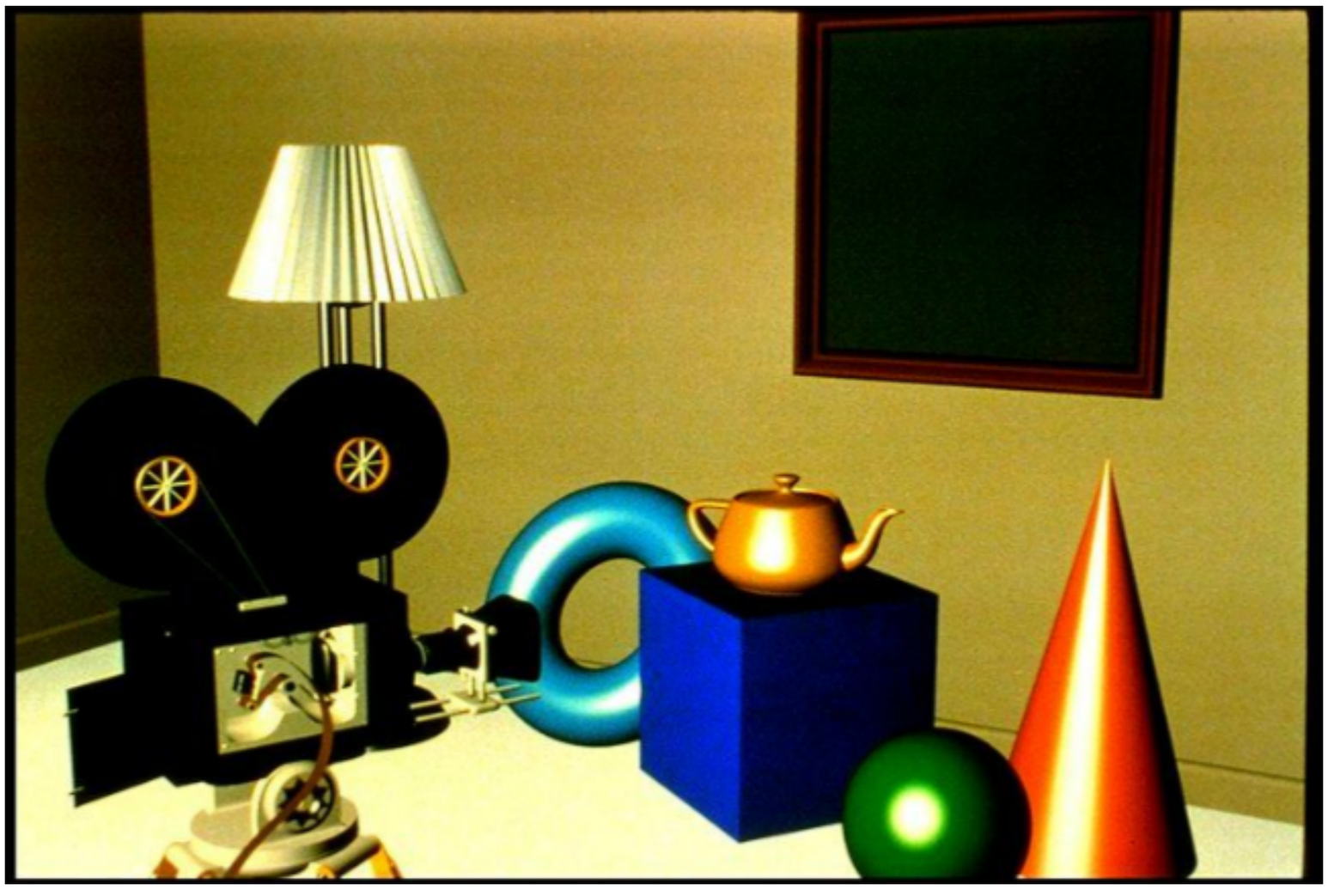
Viewing the scene from a specific camera position - perspective transformation of 3D coordinates to 2D screen coordinates



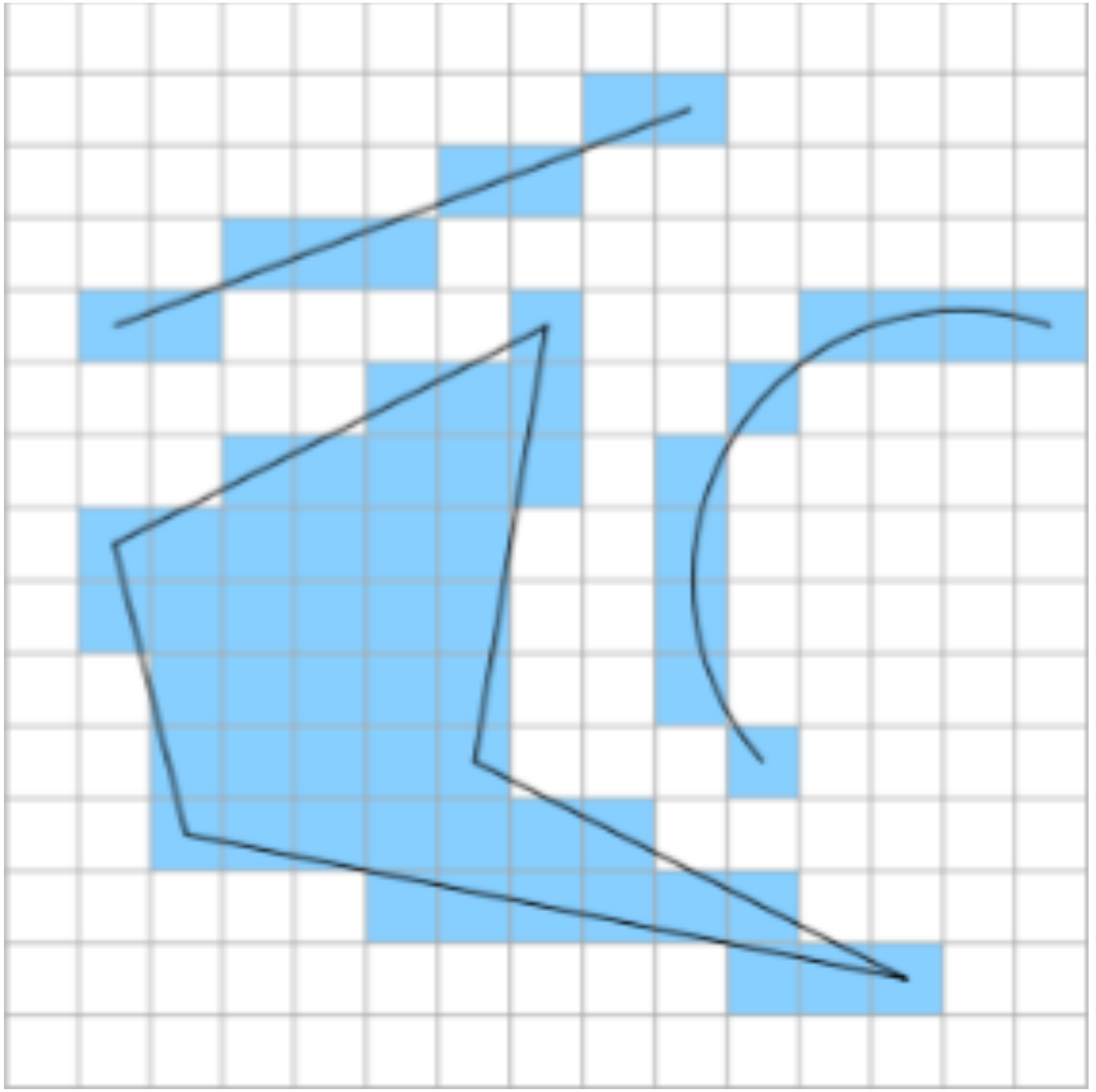
Hidden surface removal



Lighting



Rasterisation



Texture mapping

Other effects

- Bump mapping
- Reflections
- Shadows

Overview

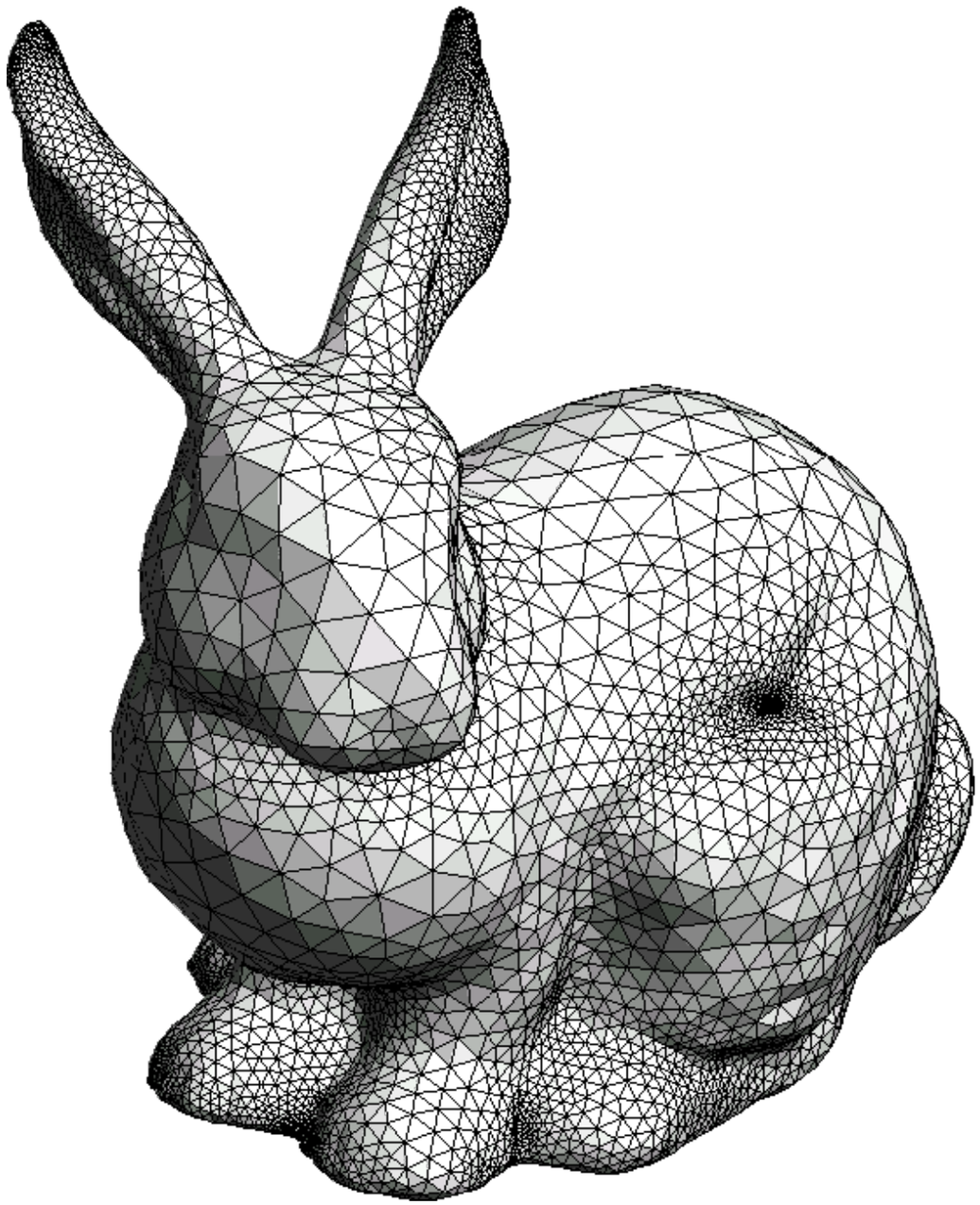
Polygon meshes

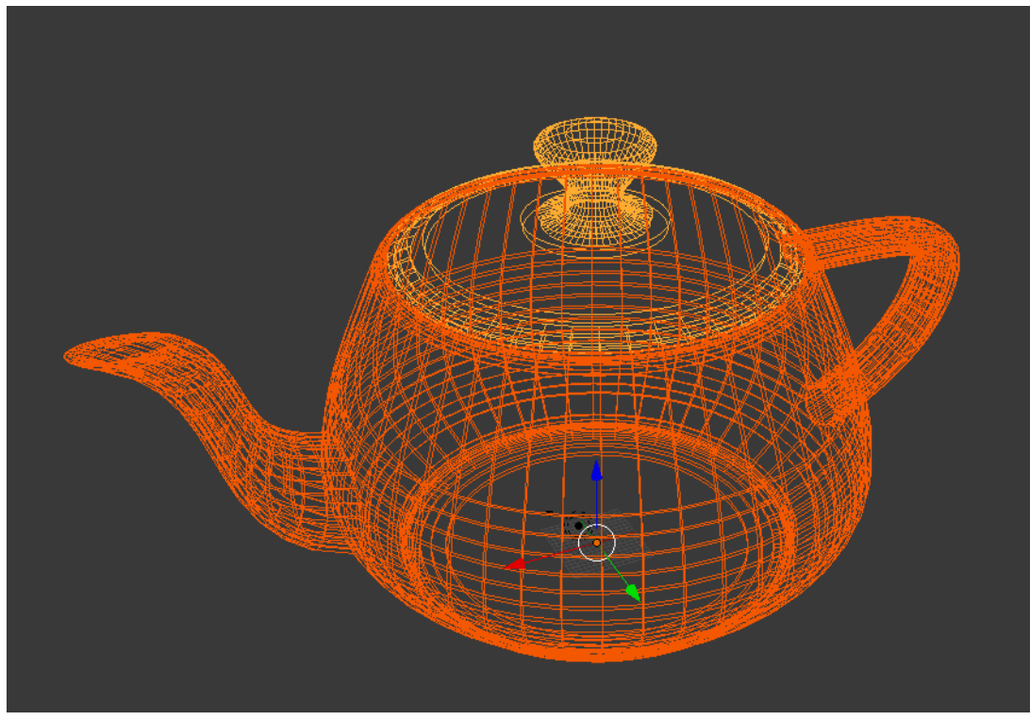
- Storage
- Representation
- Decomposition
- Quad meshes
- Generation
- Implicit surfaces

Level of detail

- Scaling methods

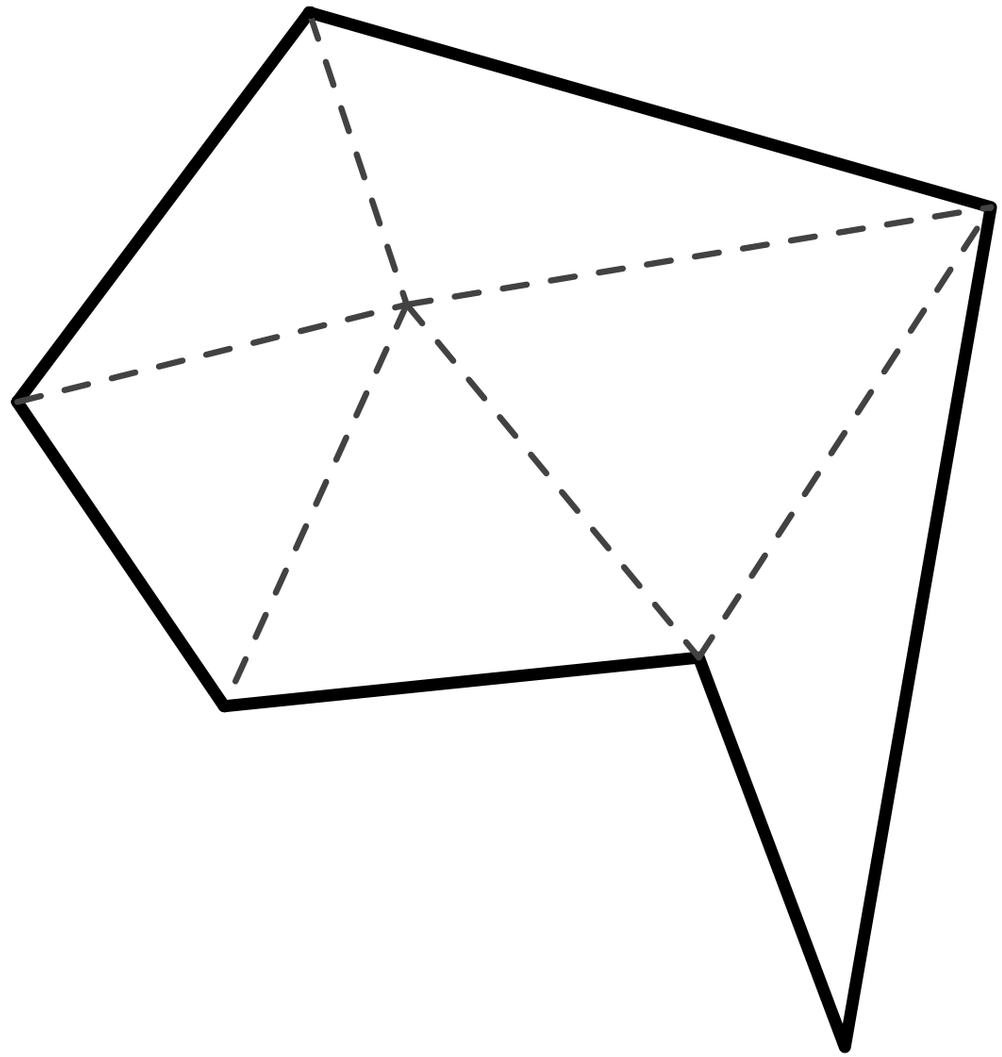
Mesh structures





Objects represented as a set of polygons

Triangle representation

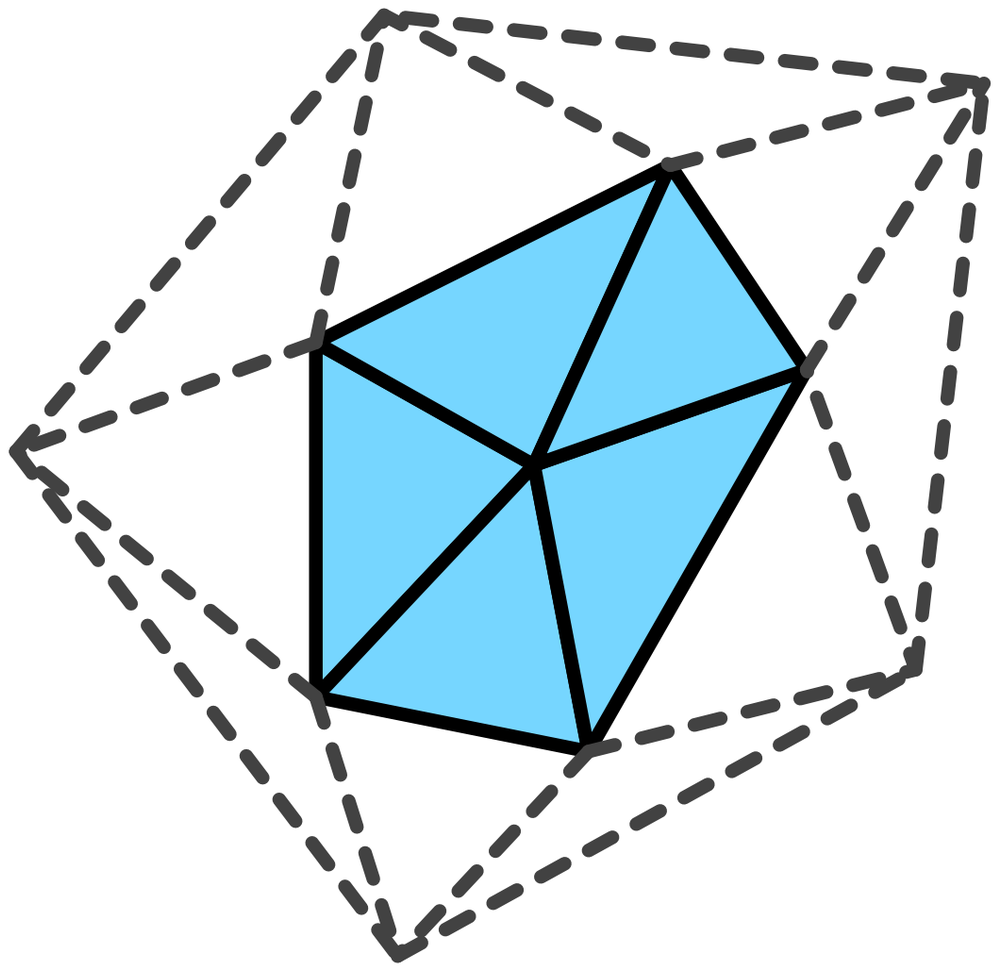


- Polygons can be broken down into triangles

Mesh topology

Manifolds:

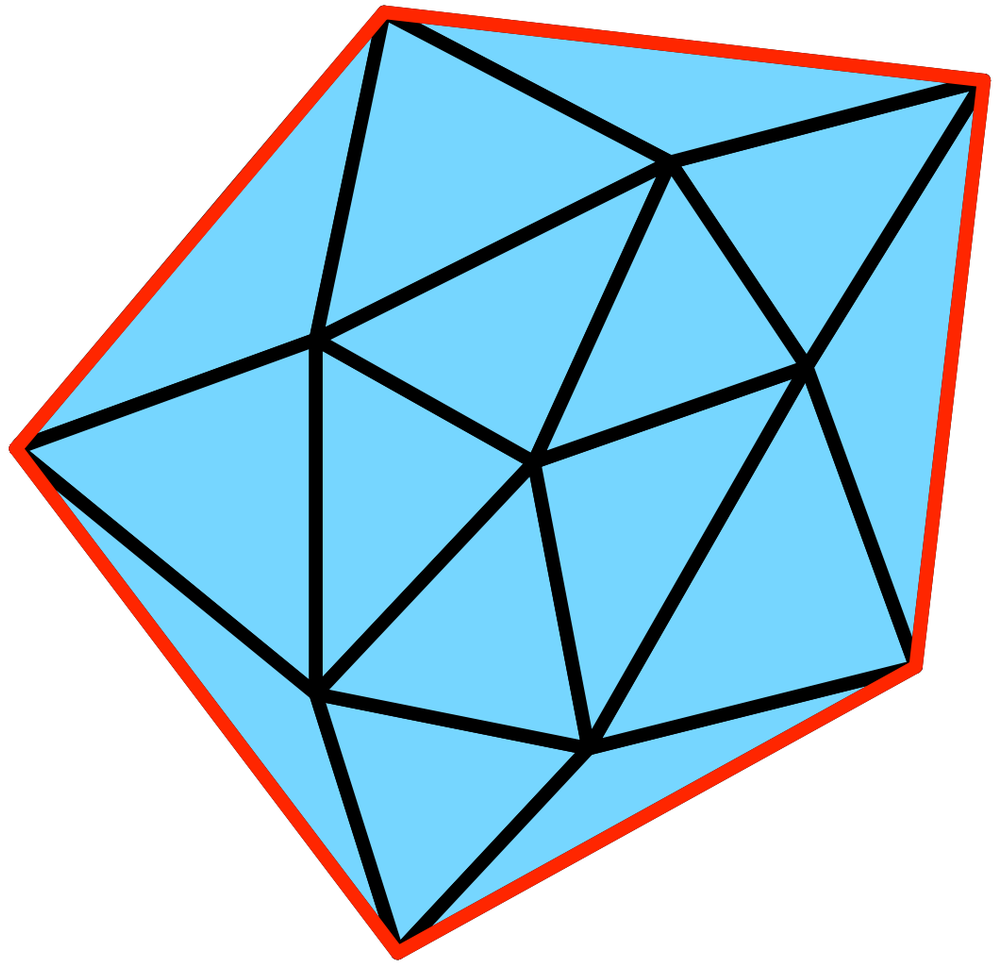
- All edges belong to two triangles
- All vertices have a single continuous set of triangles around them



Mesh topology

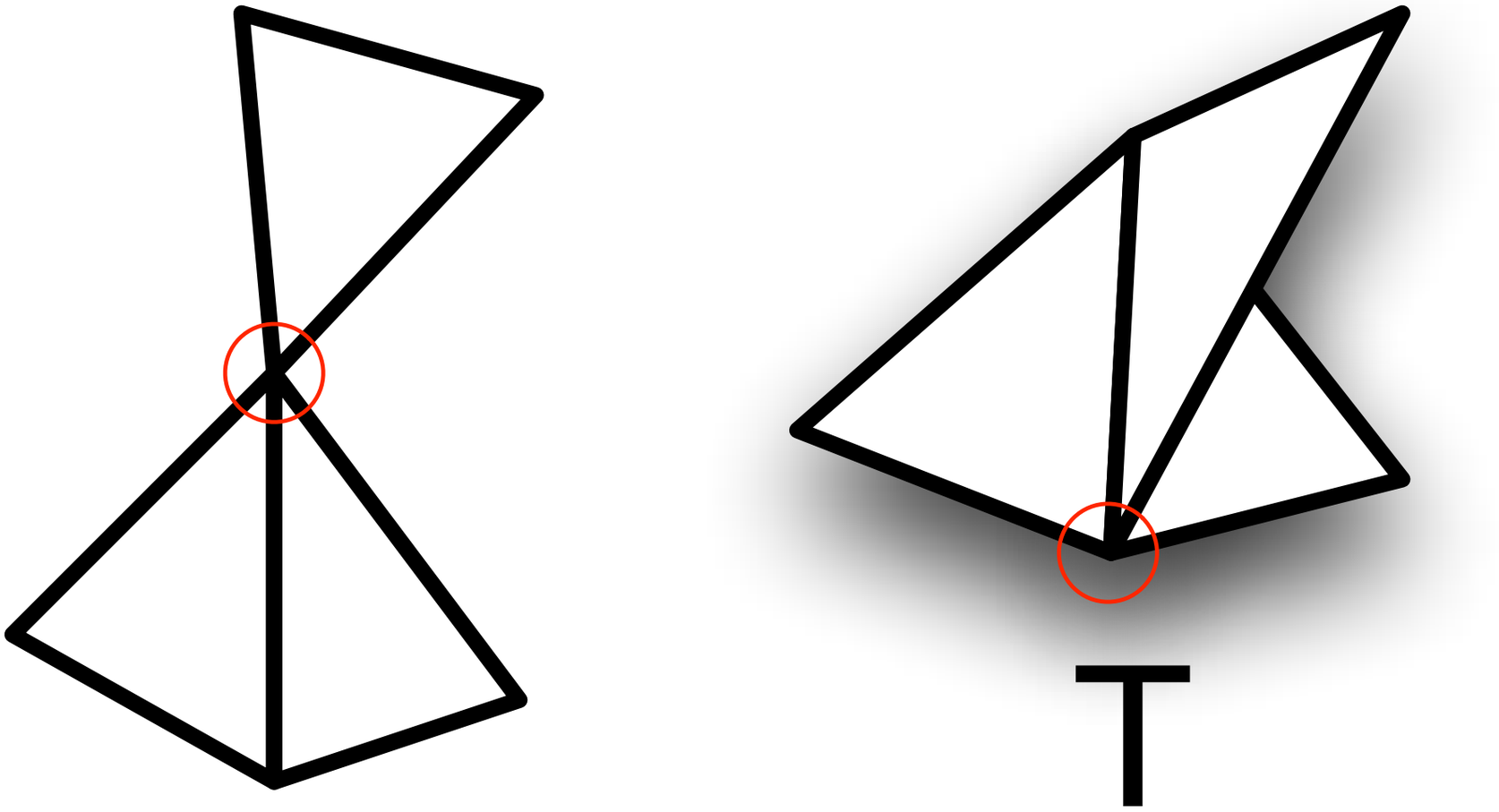
Manifolds with boundaries:

- All edges belong to one or two triangles
- All vertices have a single continuous set of triangles around them



Mesh topology

Examples of non-manifold meshes:



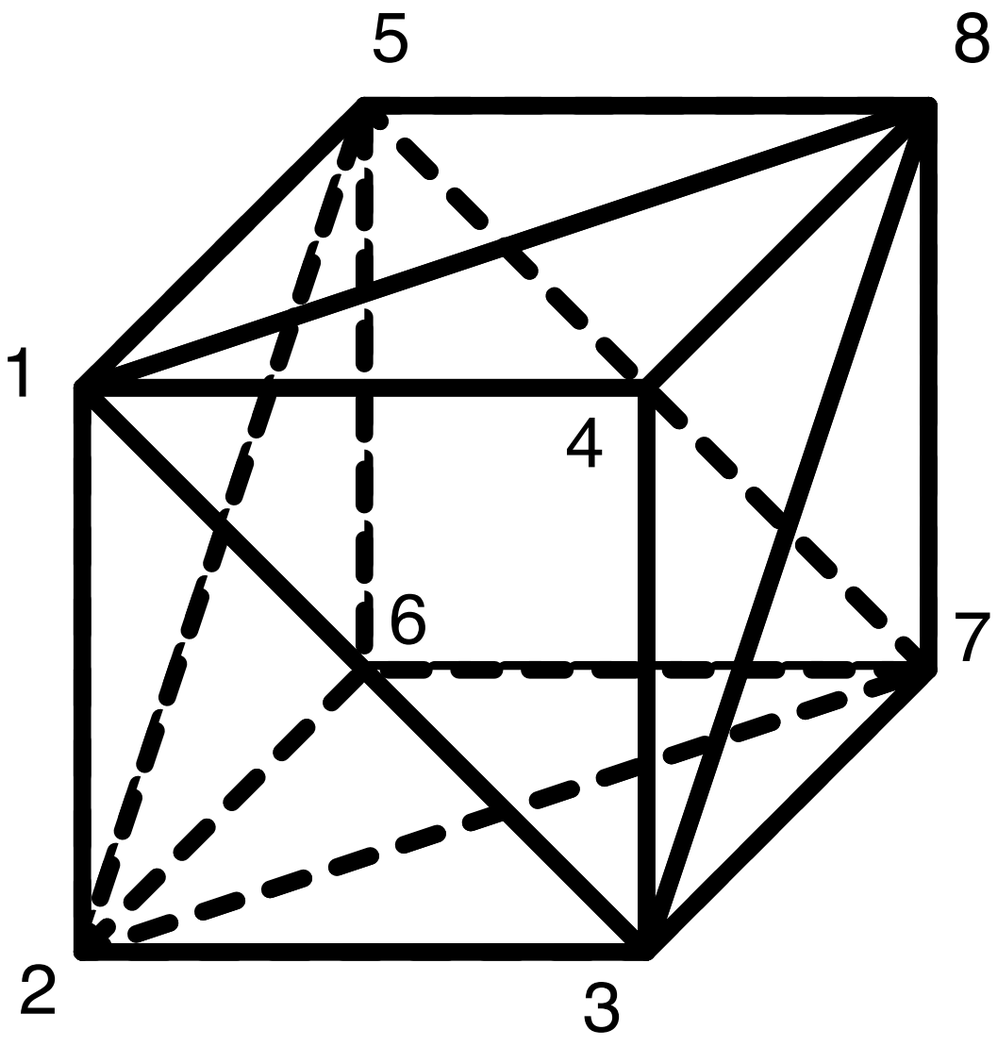
Storage format

Vertices:

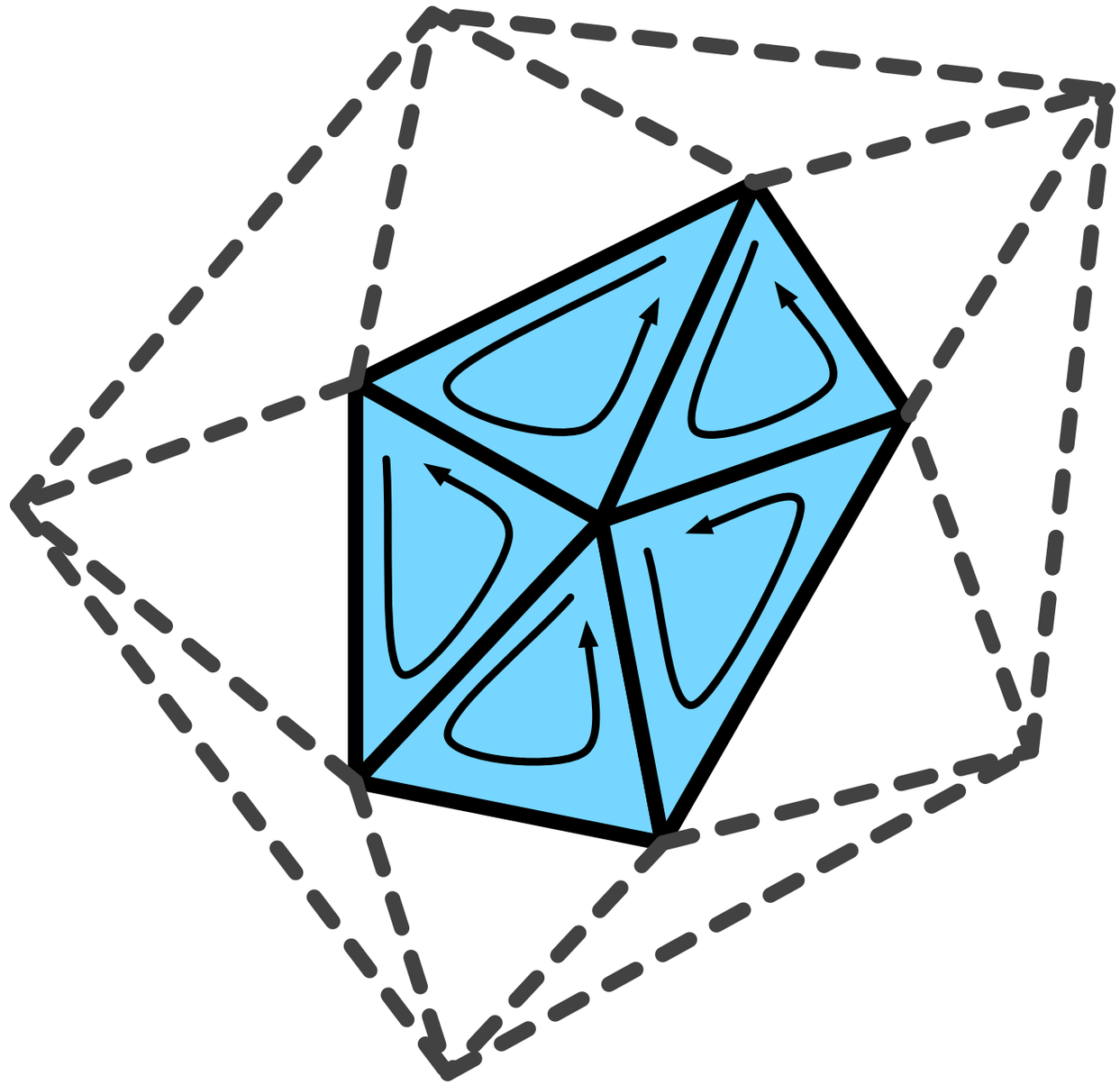
-1	1	1
-1	-1	1
1	-1	1
1	1	1
-1	1	-1
-1	-1	-1
1	-1	-1
1	1	-1

Polygons:

1	2	3
1	3	4
1	4	8
1	8	5
4	3	8
3	7	8
5	7	6
5	8	7
1	5	2
5	6	2
2	6	7
2	7	3

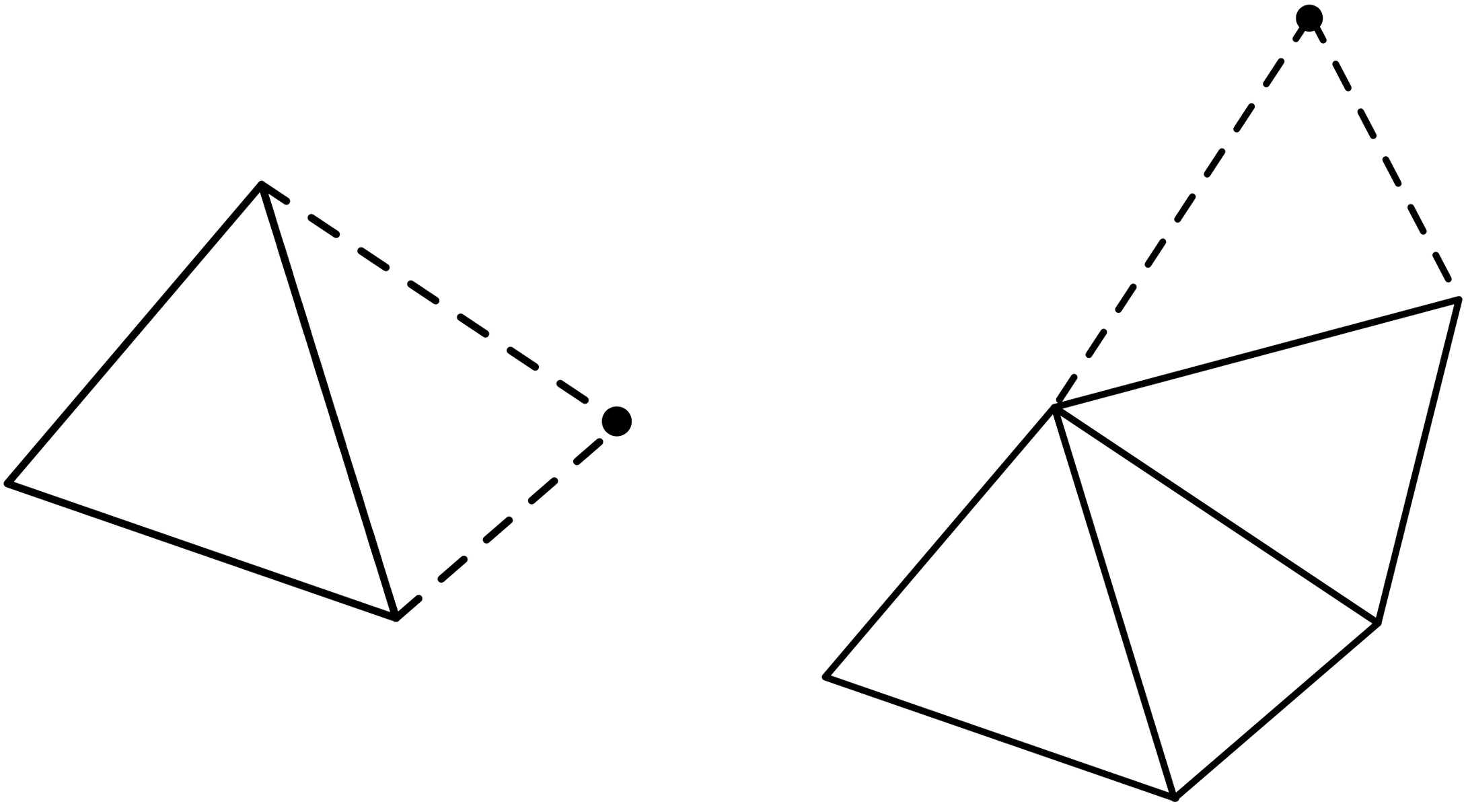


Orientation



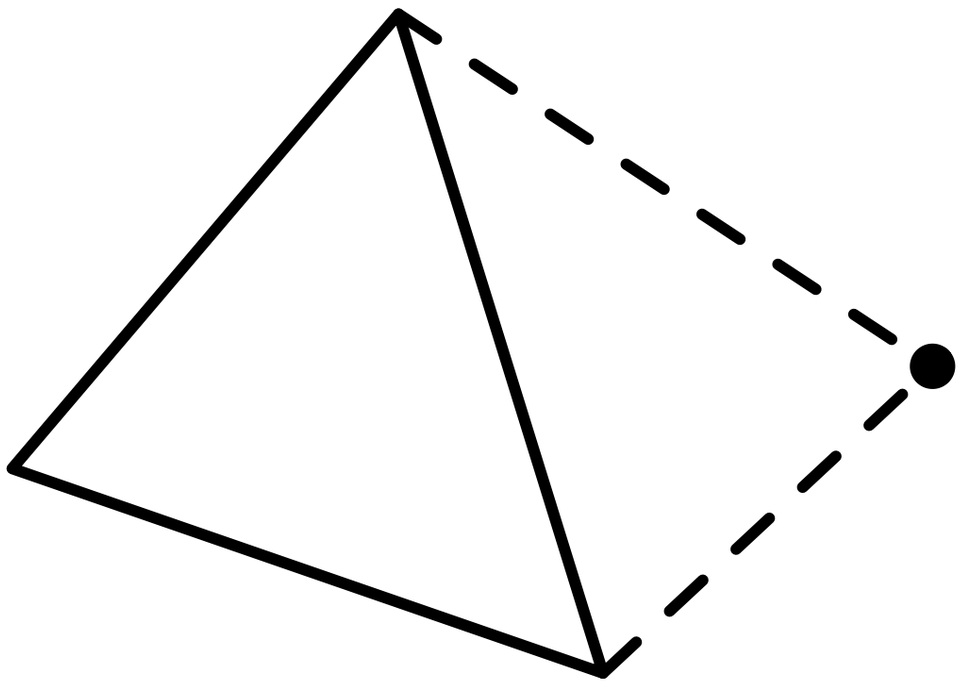
Vertexes in triangle list stored in counter clockwise order

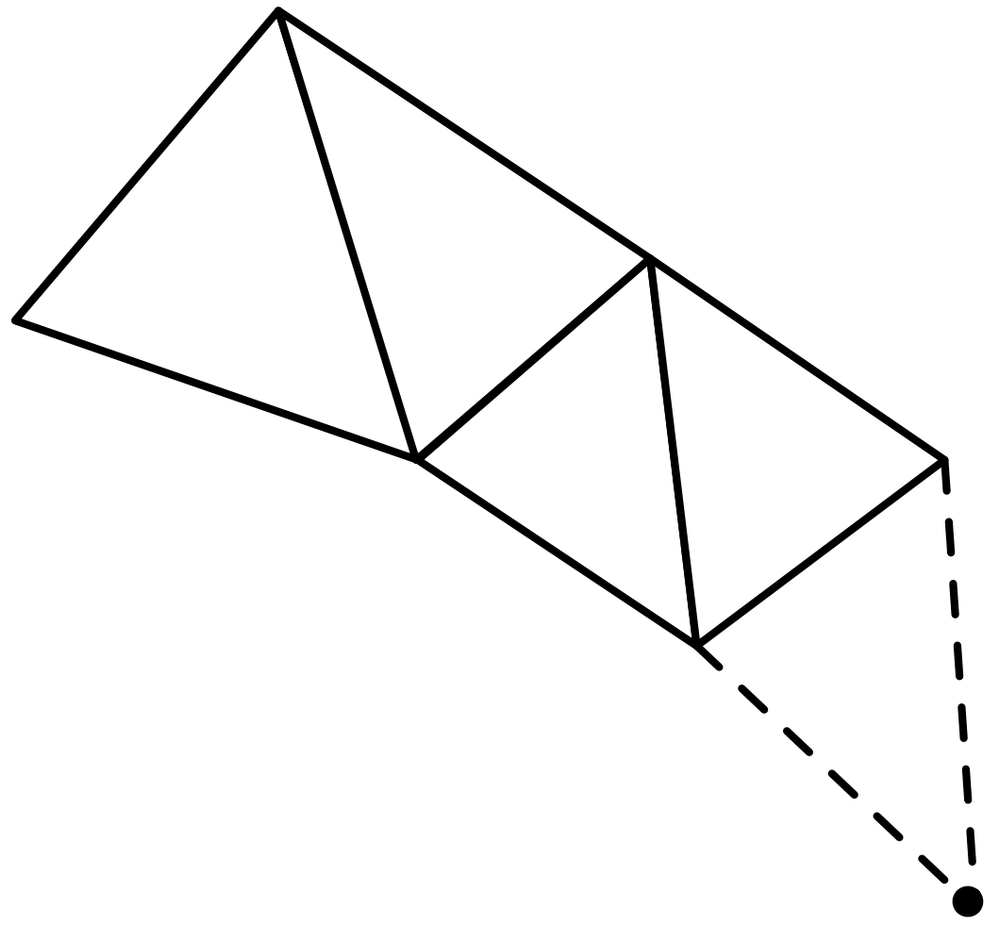
Triangle fans



Instead of storing $3T$ vertices, store $T + 2$

Triangle strips



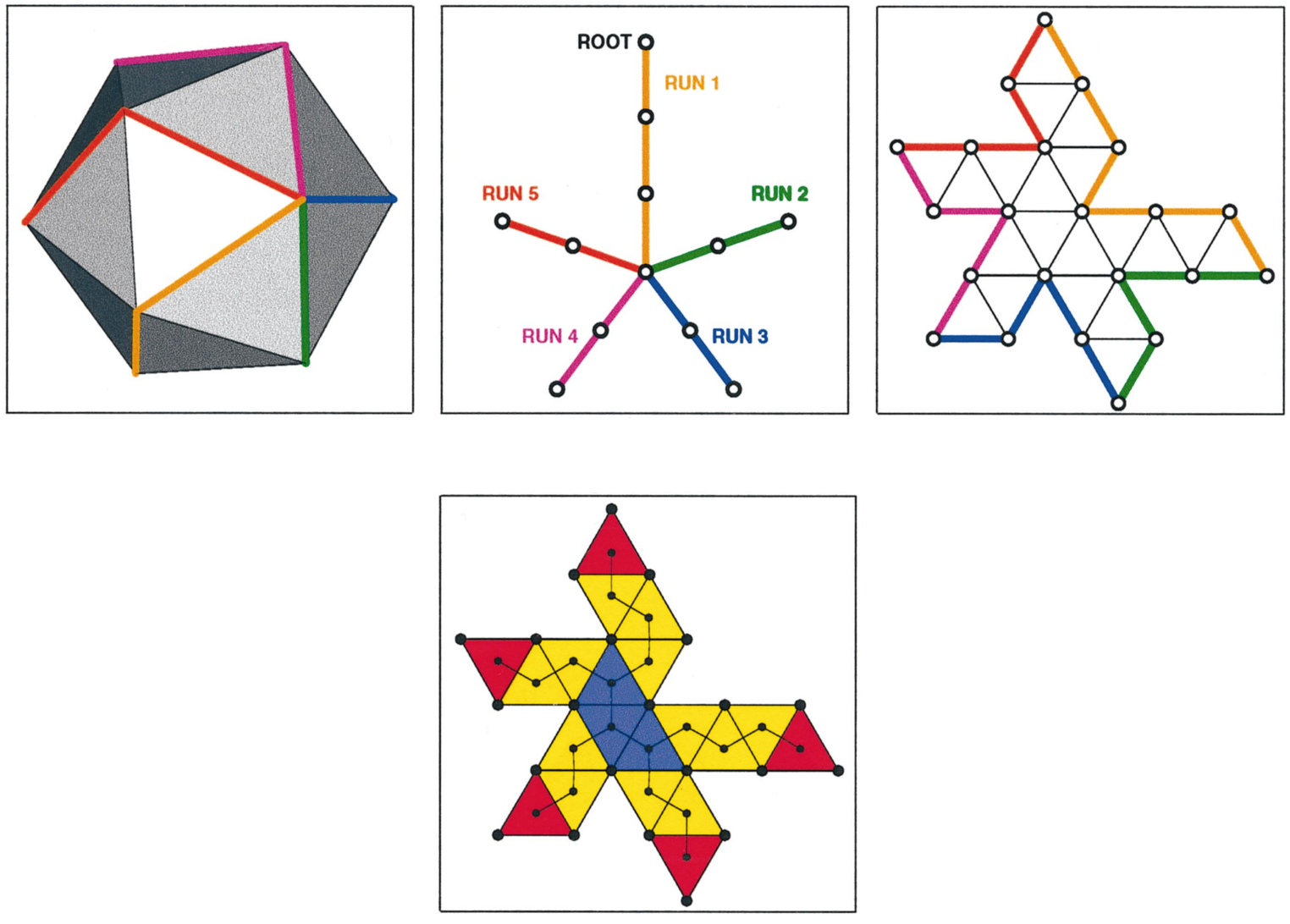


Instead of storing $3T$ vertices, store $T + 2$

Minimal spanning tree decomposition

From a mesh, produce a graph where:

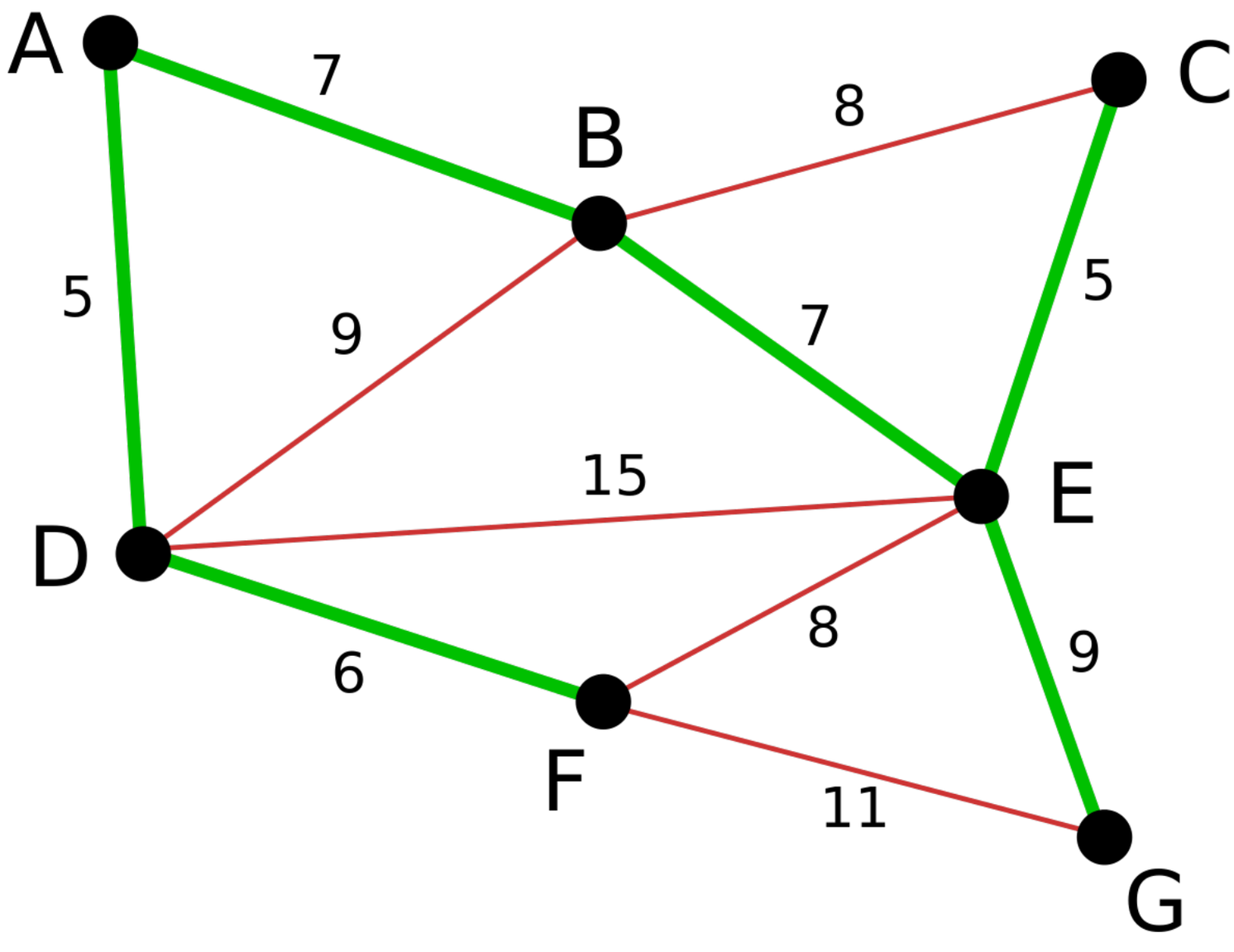
- nodes correspond to triangles of the mesh
- edges shared by a pair of triangles become edges in the graph
- cost is set to the Euclidean distance between the triangle and the root triangle



Minimal spanning tree decomposition

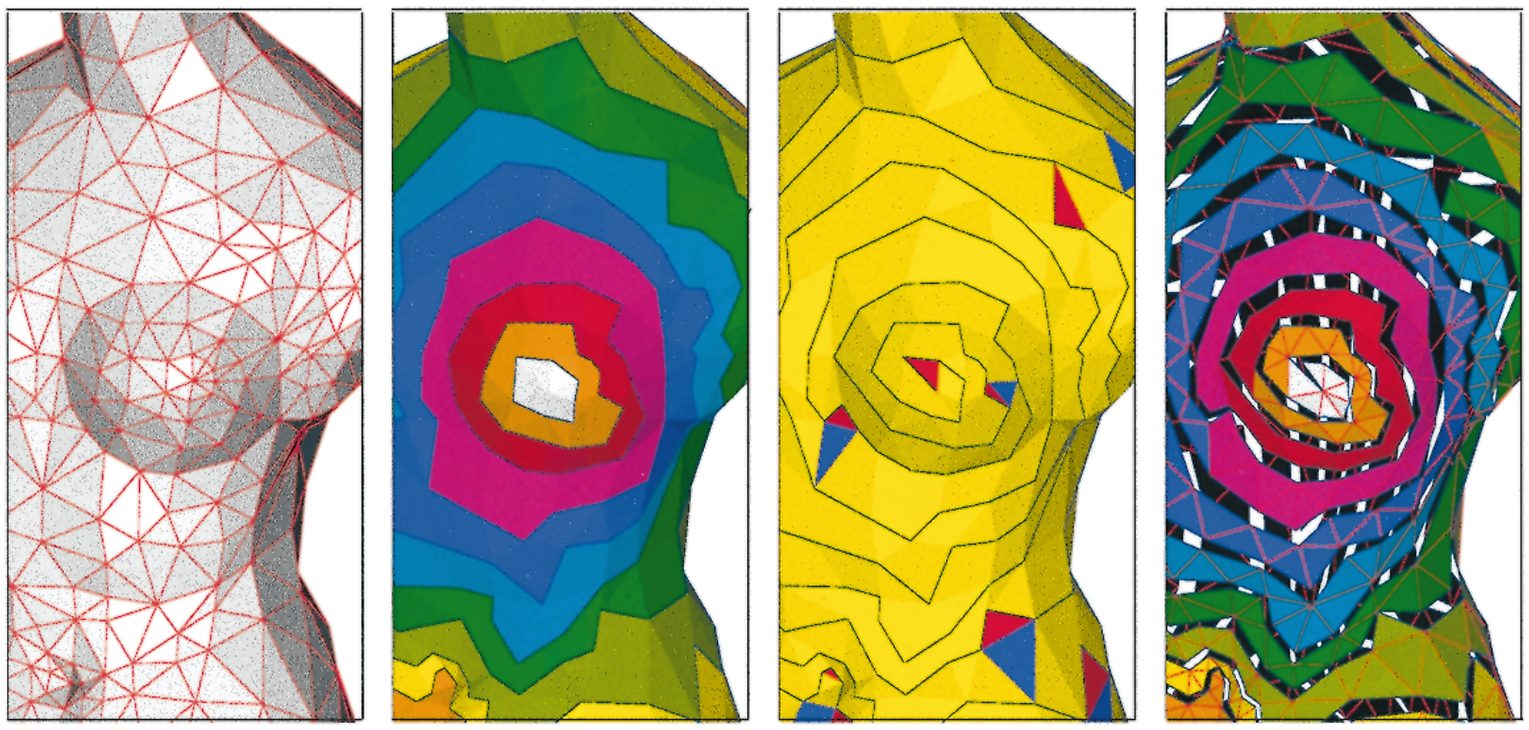
A minimal spanning tree produces a tree over a graph which visits every node whilst minimising costs

- edges are given costs
- a tree with minimal total cost is found



Sprial decomposition

- Generate layers of triangles
- Connect layers into a single spiral of triangles



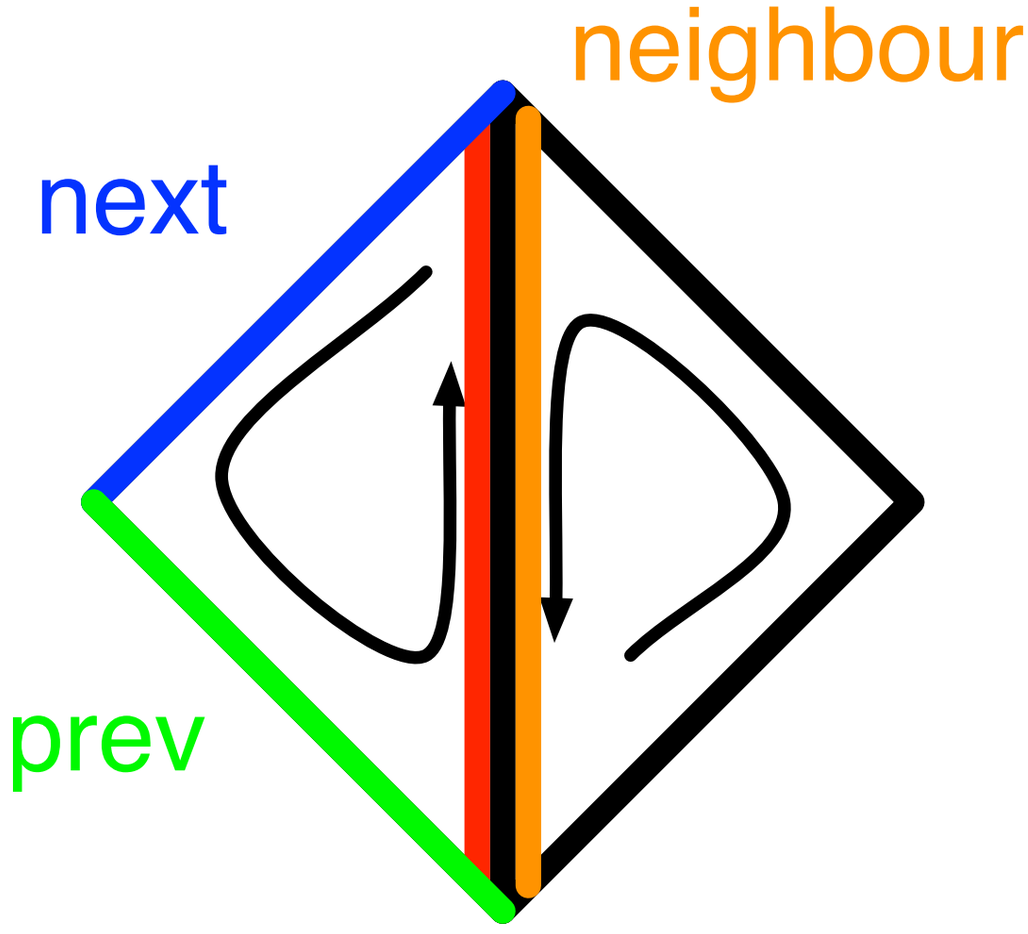
Directed edge data structure

Vertices

x_1	y_1	z_1	e_r
x_2	y_2	z_2	e_s
⋮	⋮	⋮	⋮

Edges

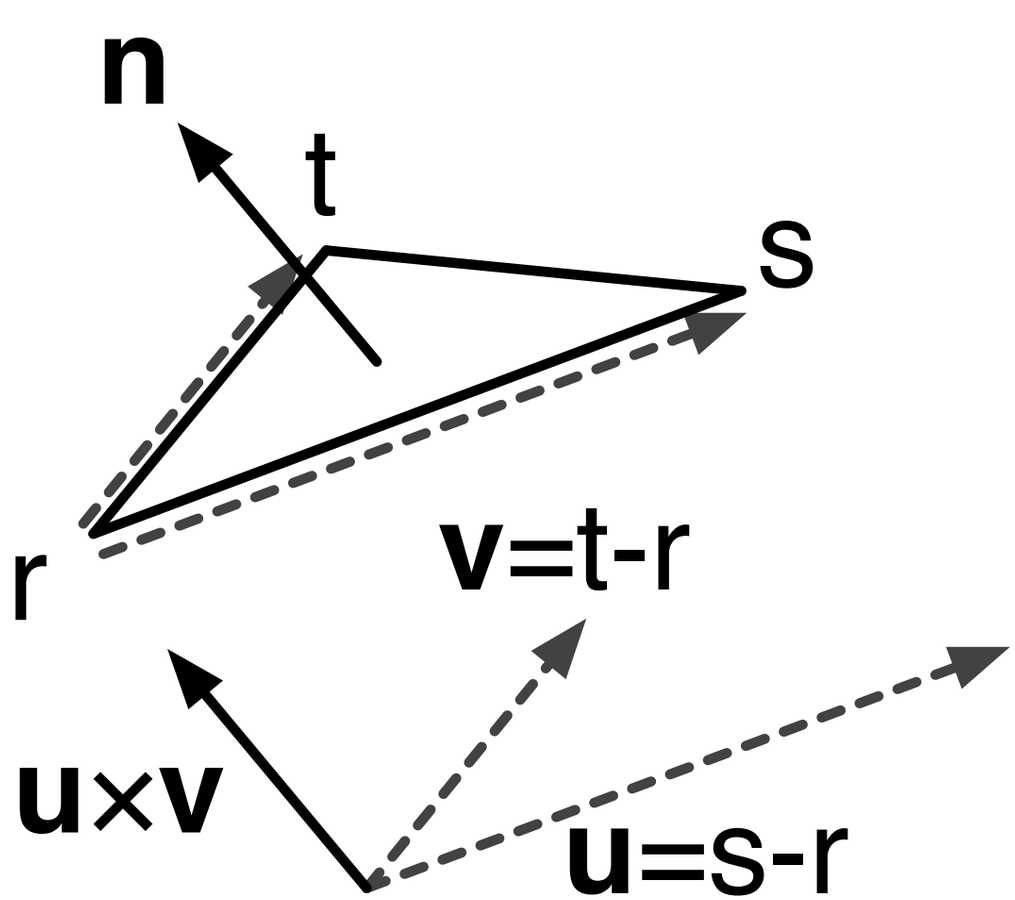
v_a	v_b	$e_{neighbour}$	e_{next}	e_{prev}
⋮	⋮	⋮	⋮	⋮

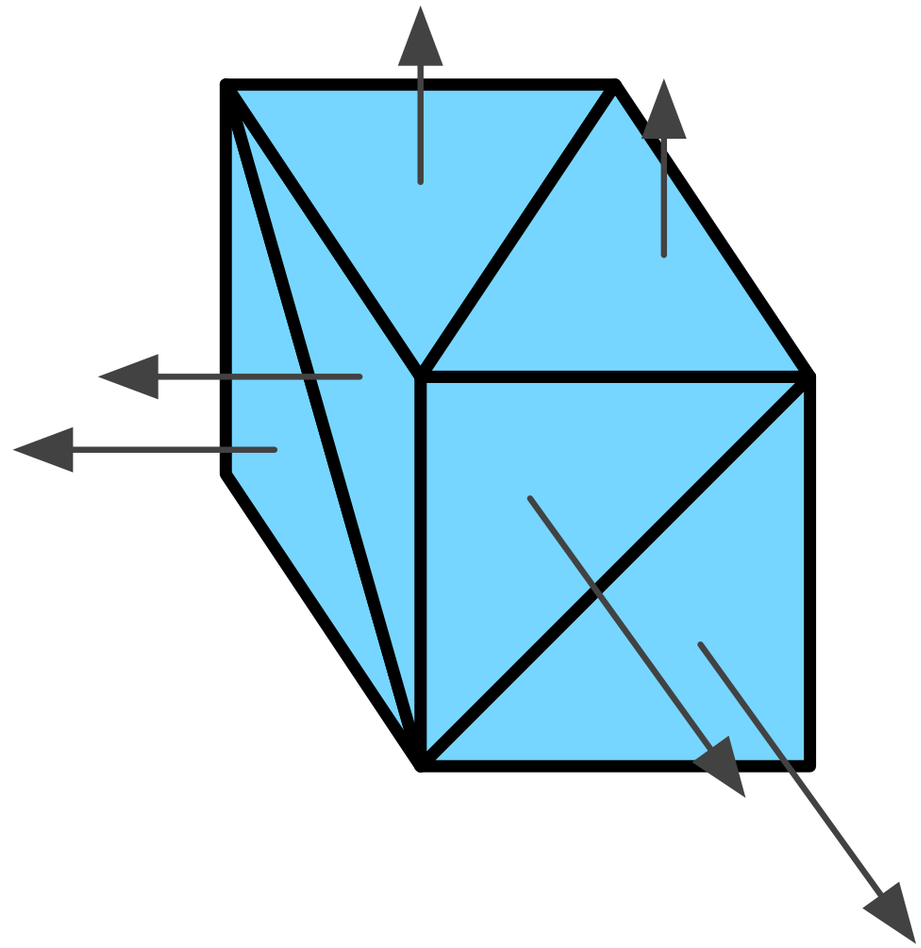


Surface normals

Calculation:

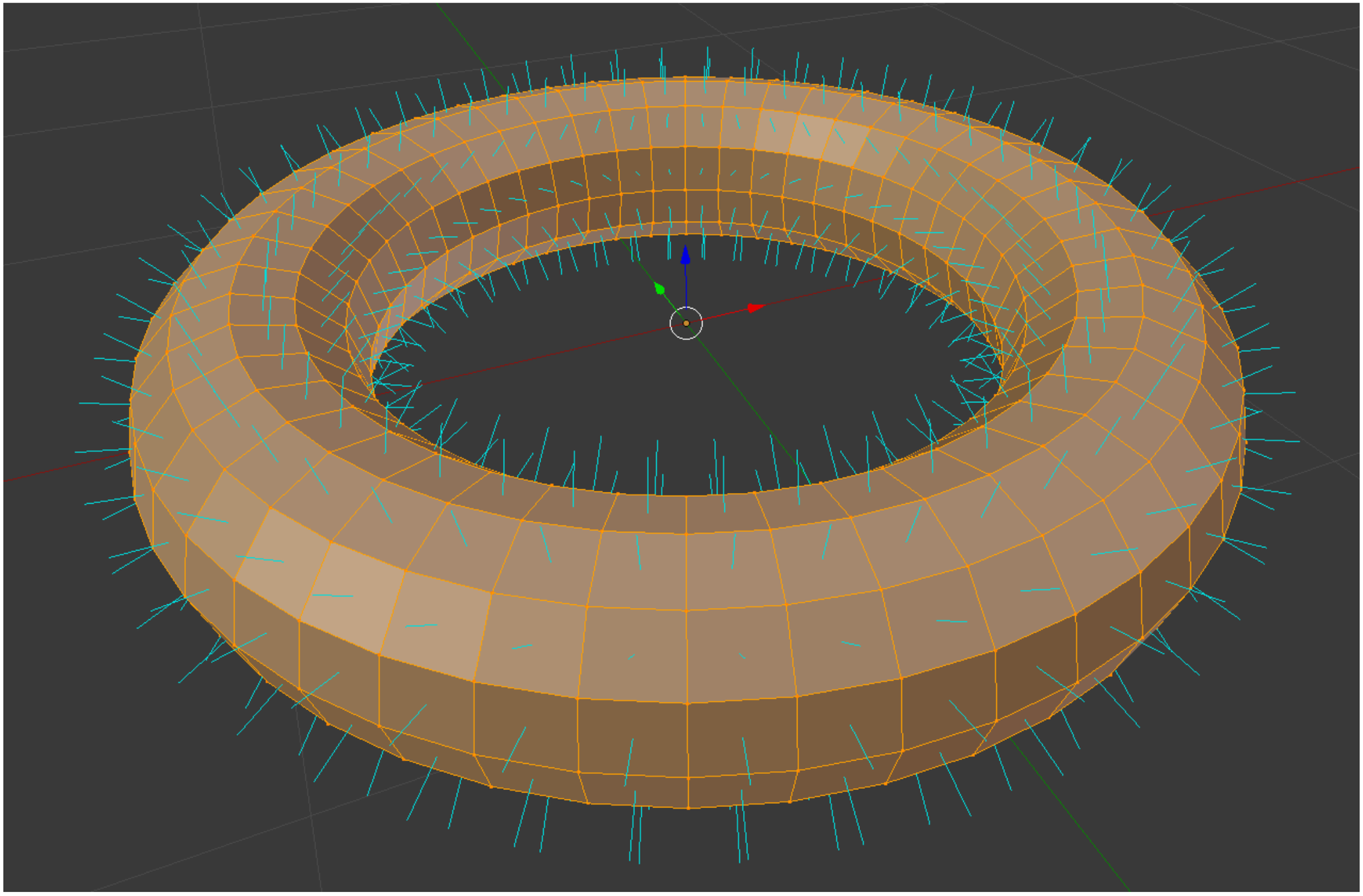
$$\boldsymbol{n} \quad = \quad (s - r) \times (t - r)$$





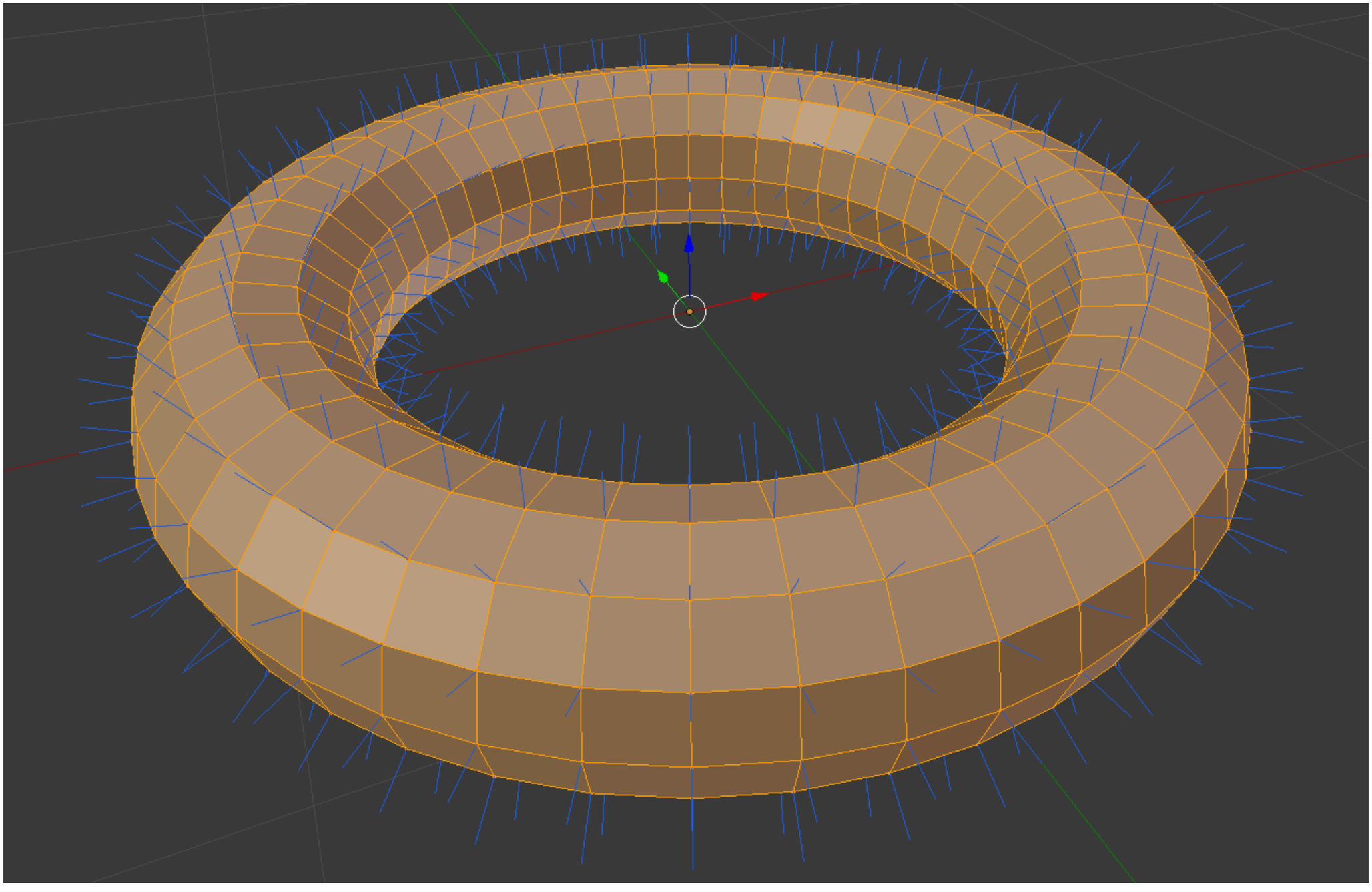
Surface normals

Face normals



Surface normals

Vertex normals



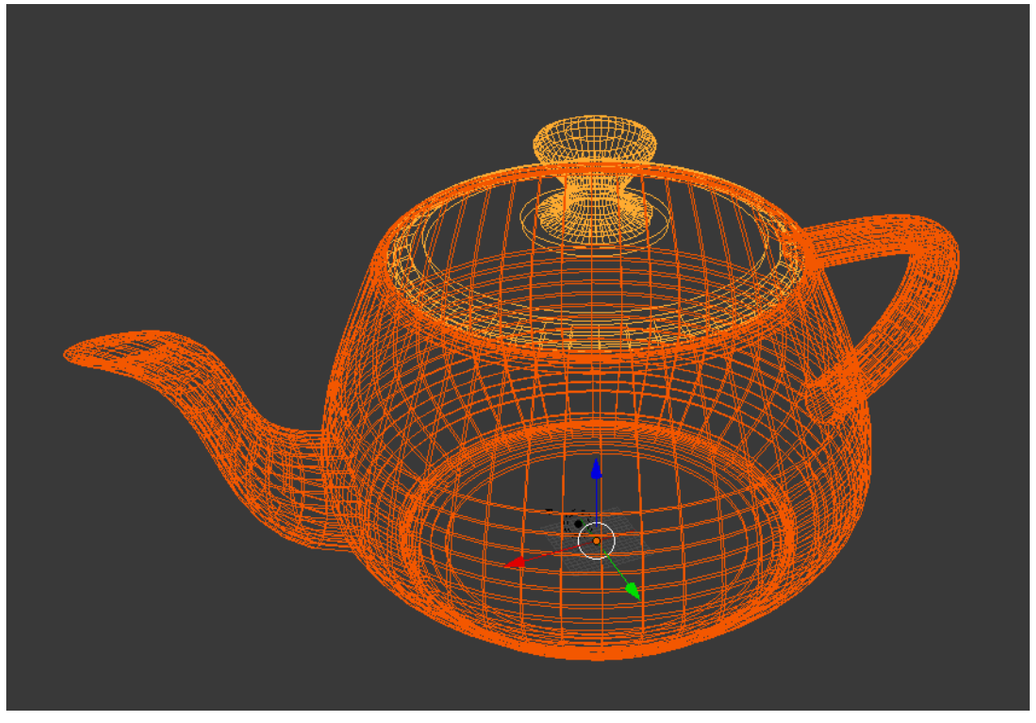
Quadrilateral meshes

Relatively easy to extend to quadrilateral meshes (quadmeshes):

- Same storage format

Problems:

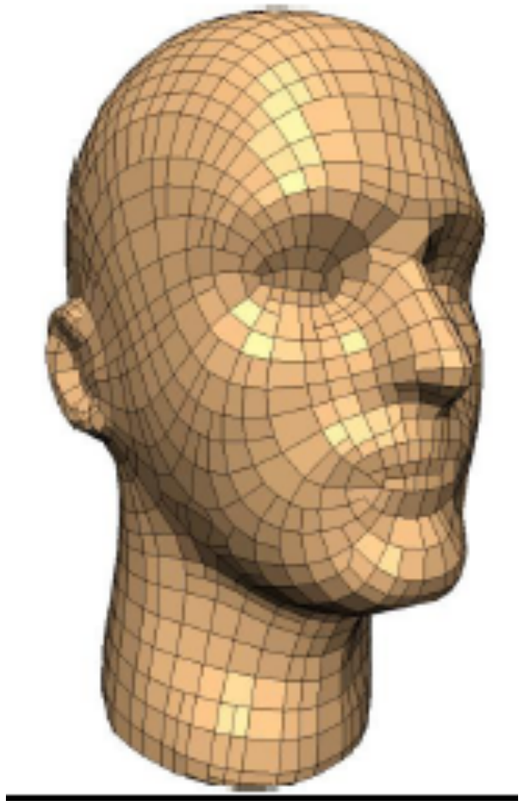
- 4 points don't always lie on a plane

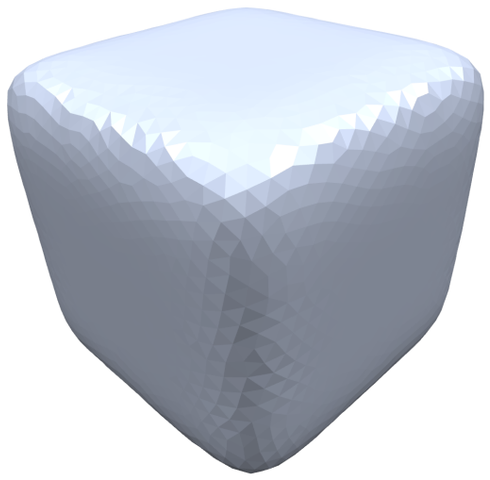


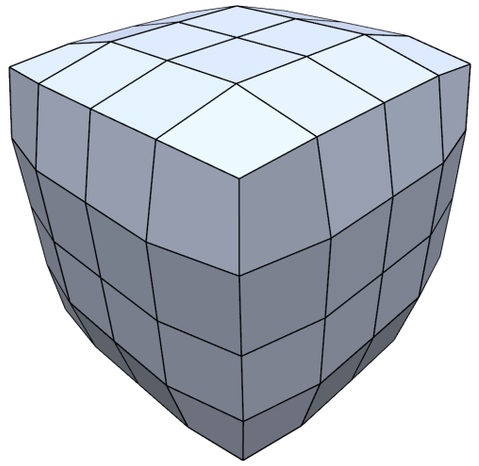
Quadrilateral meshes

Benefits of quadmeshes:

- Easier to align edges to curvature or feature lines defining an object
- Easier to apply texture maps
- Simpler to fit using parametric surfaces (e.g. curved surfaces)



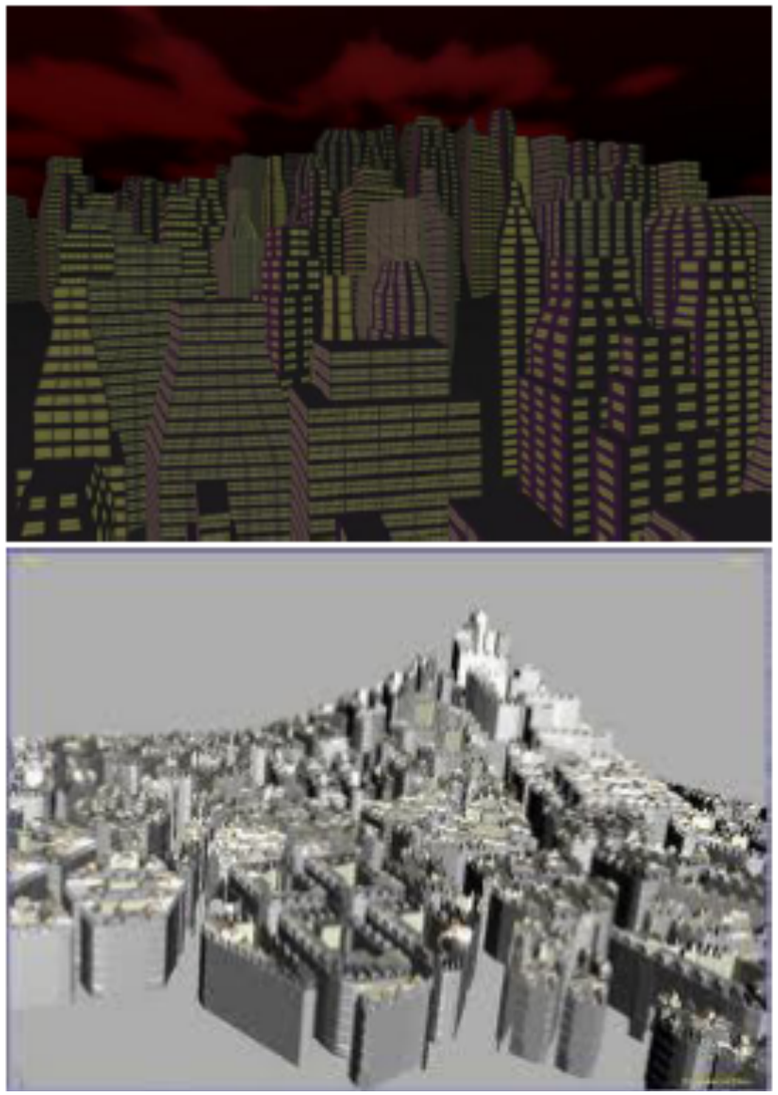


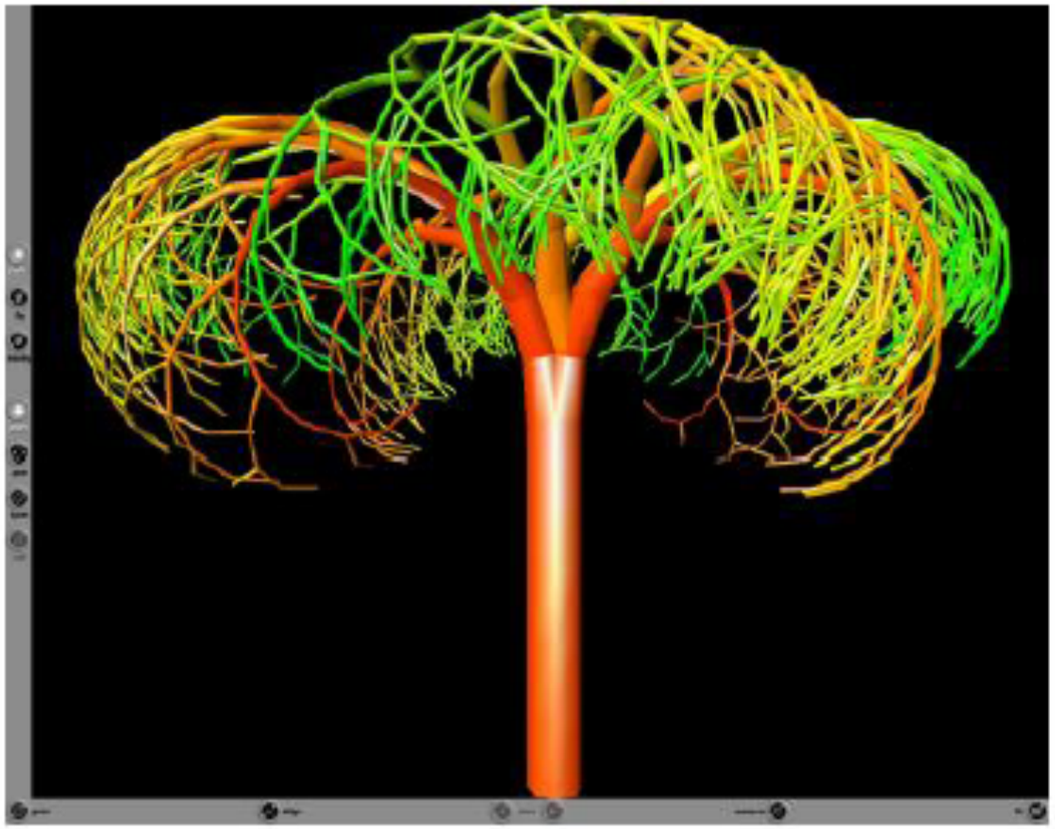


Generating polygon mesh data

Where does the mesh come from?

- Modelling using software (e.g. Blender)
- Scanning (laser range scanning, stereo vision, KINECT)
- Procedural methods



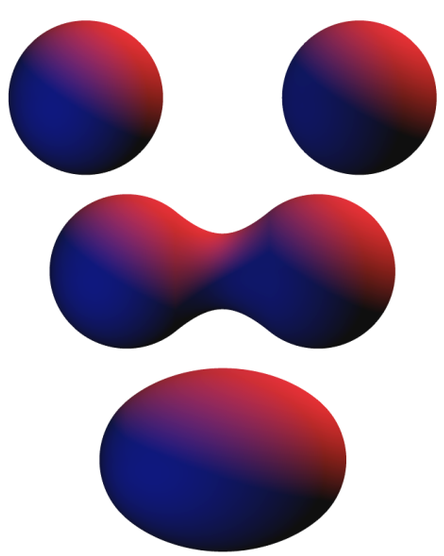


Implicit surfaces

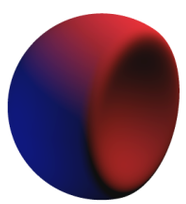
Given some potential function $f(x)$ we can evaluate at every point in space, define an isosurface of all points where $f(x) = c$

- Metaballs (Blinn, Ohmura)
- $\sum_i \frac{a_i}{r}$

1



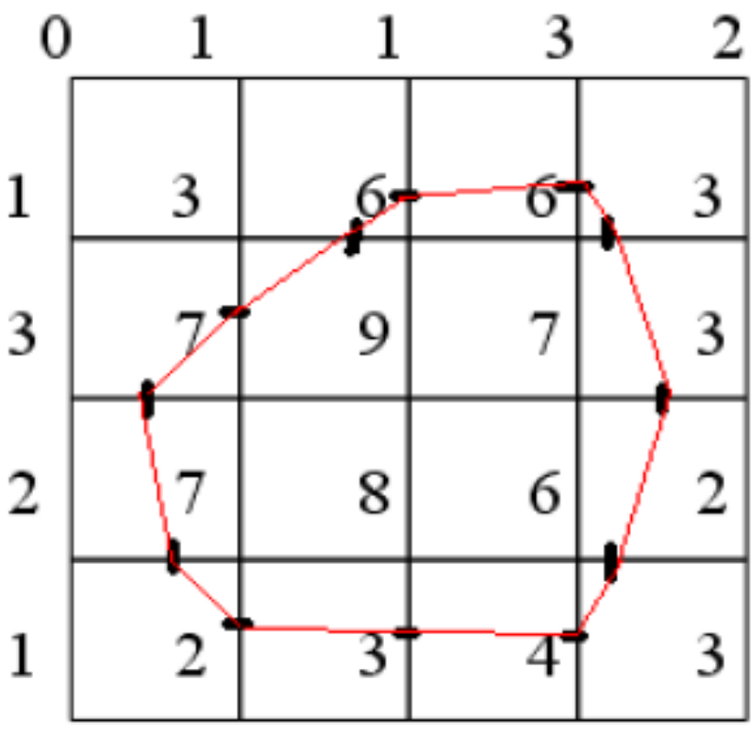
2

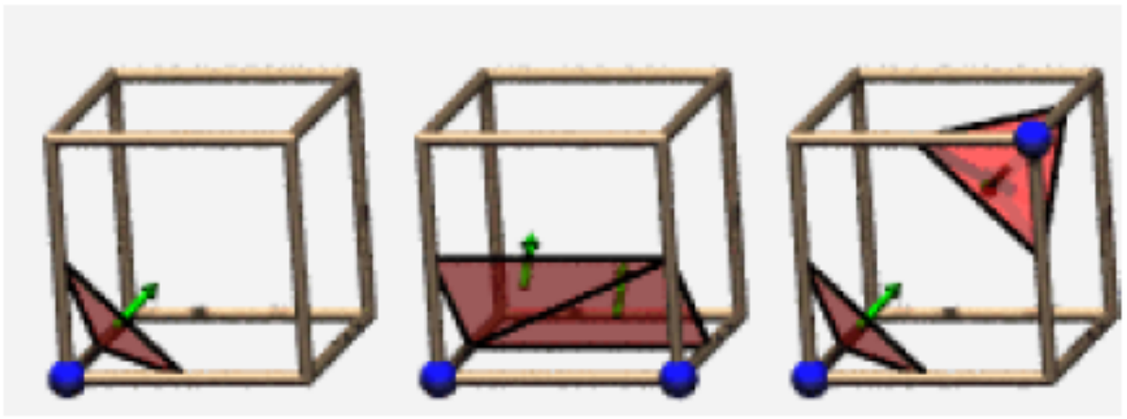


Marching cubes

Potential is computed at each point on a grid

- where edges of the grid cross the threshold, the surface is produced
- set of rules for producing the surface based on which edges cross the threshold





Level of detail scaling

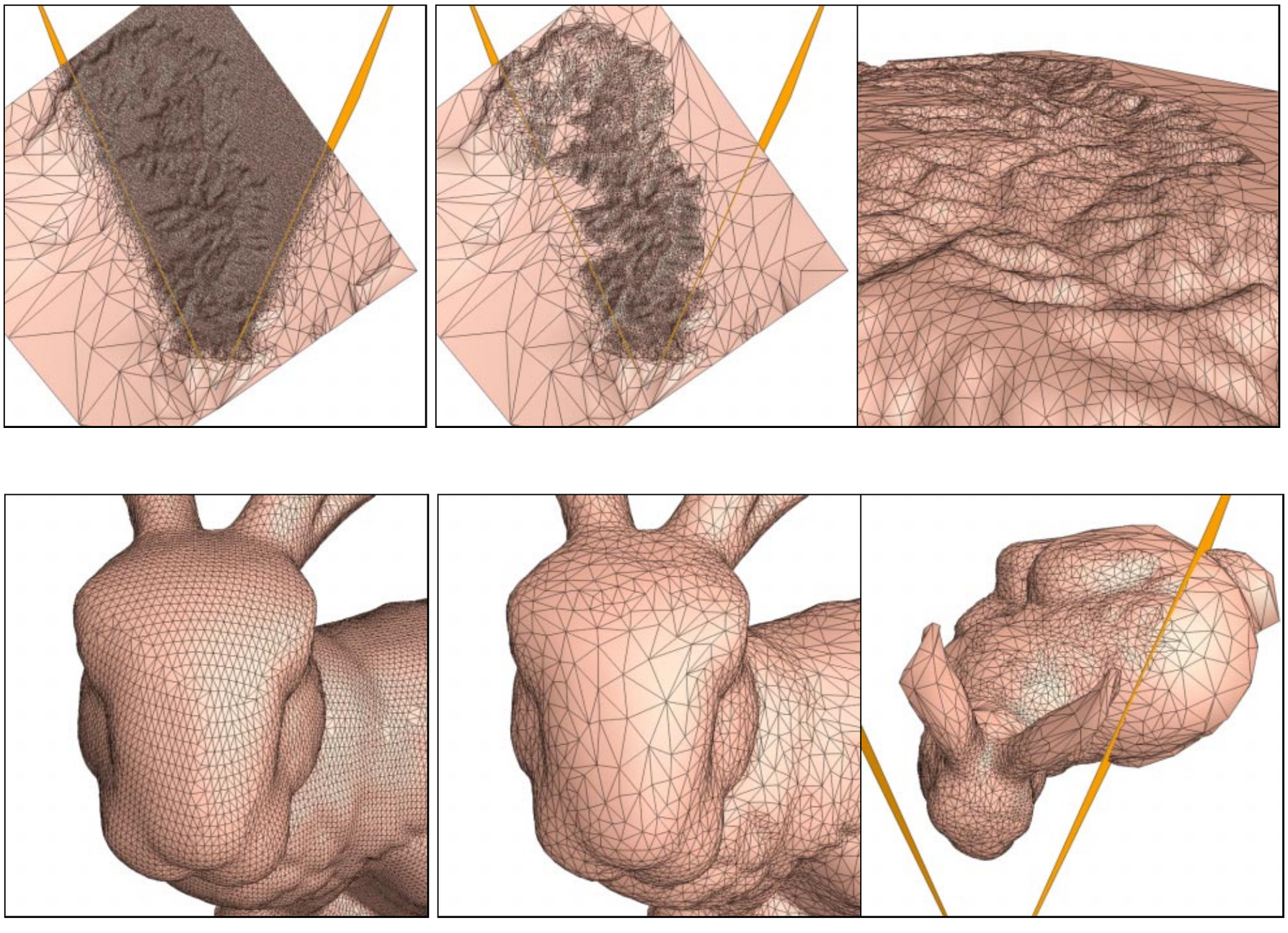
When an object is close to the camera we usually want a very detailed model. But not when:

- further away (taking up a small amount of space on screen)
- facing away from the camera
- outside the view volume

Level of detail scaling

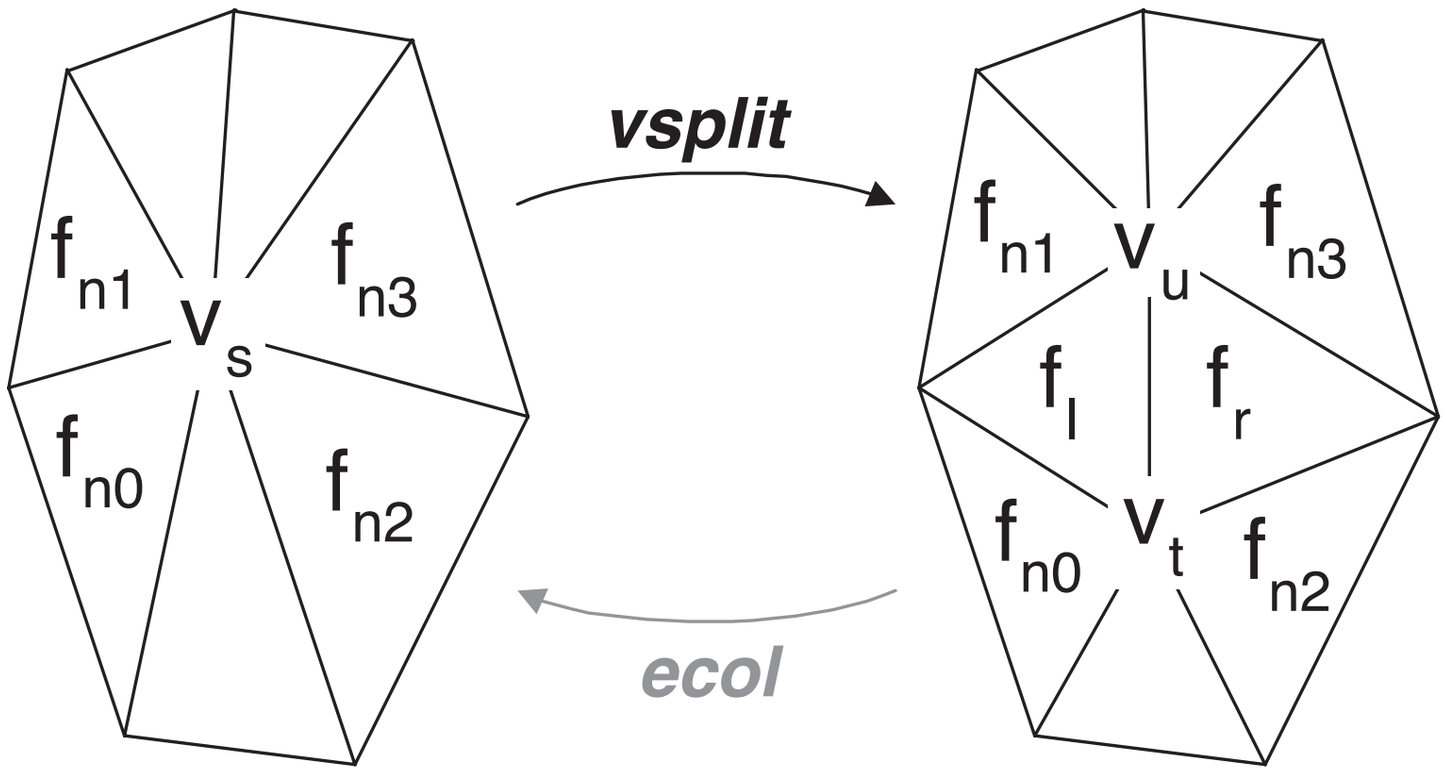
Progressive meshes

- some polygons may be much closer to camera than others, within a single mesh



Level of detail scaling

Progressive meshes



Summary

- Computer graphics pipeline
- Object representations
- Object decomposition
- Level of detail

Coursework

Coursework 1:

- OpenGL vertex and fragment shaders
- Deadline 4pm on 23/10/15

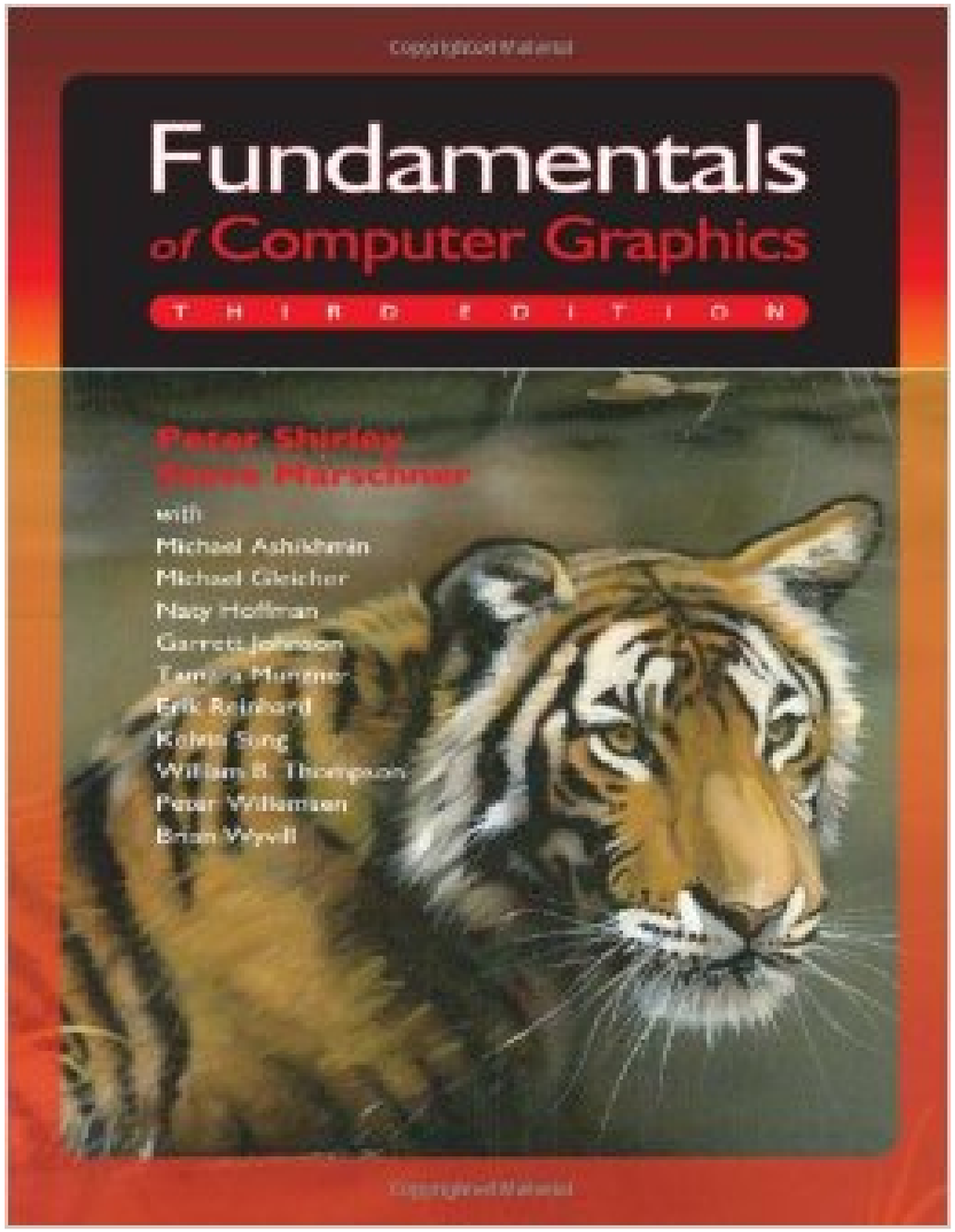
Coursework 2:

- Ray tracing
- Deadline 4pm on 20/11/15

For both:

- Written in C++
- **Must** compile and run on DICE (Scientific Linux 7)
- See course page for submission details

Books

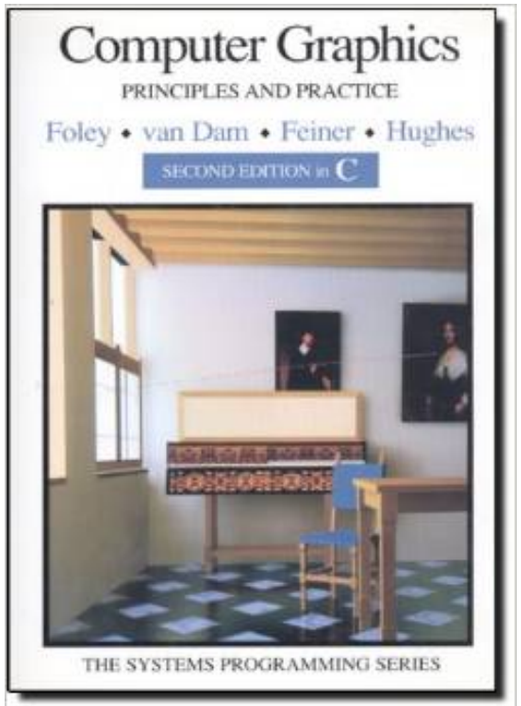


Available online via the library

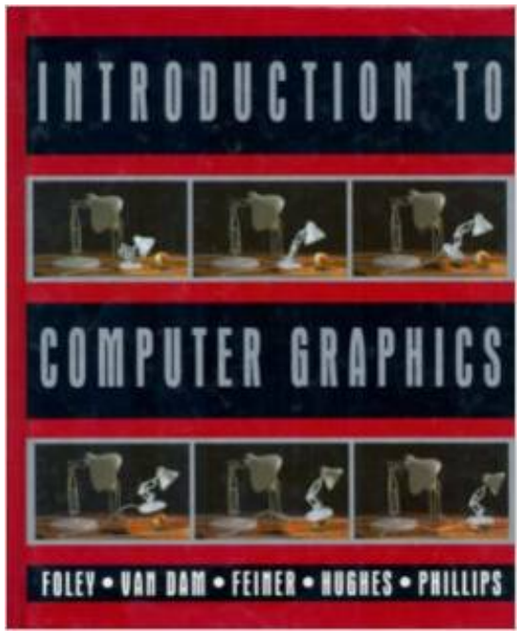
Fundamentals of Computer Graphics

Shirley and Marschner, CRC Press, 2010.

Books



Computer Graphics Principles and Practice
Foley, van Dam, Feiner and Hughes, Addison Wesley, 1997.



Introduction to Computer Graphics
Foley, van Dam, Feiner, Hughes and Phillips, Addison Wesley, 1995.

Reading

- Reference materials – strongly recommended
- Reference materials – not compulsory but may be useful to help in understanding of materials
- Papers etc – extra materials for those interested

References

Shirley (recommended):

- Chapter 2.4 (Miscellaneous Math(s) – Vectors)
- Chapter 12.1 (Data Structures for Graphics – Triangle Meshes)

Foley (reference):

- Appendix A.1-A.5 (Maths for Computer Graphics)

Papers (optional extra):

- Taubin, G., Rossignac, J. (1998). Geometric compression through topological surgery. ACM Transactions on Graphics, 17(2), 84–115.
- D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini and D. Zorin, Eurographics STARS, 2012
- Hoppe, H. (1997). View-dependent refinement of progressive meshes (pp. 189–198). SIGGRAPH '97 }