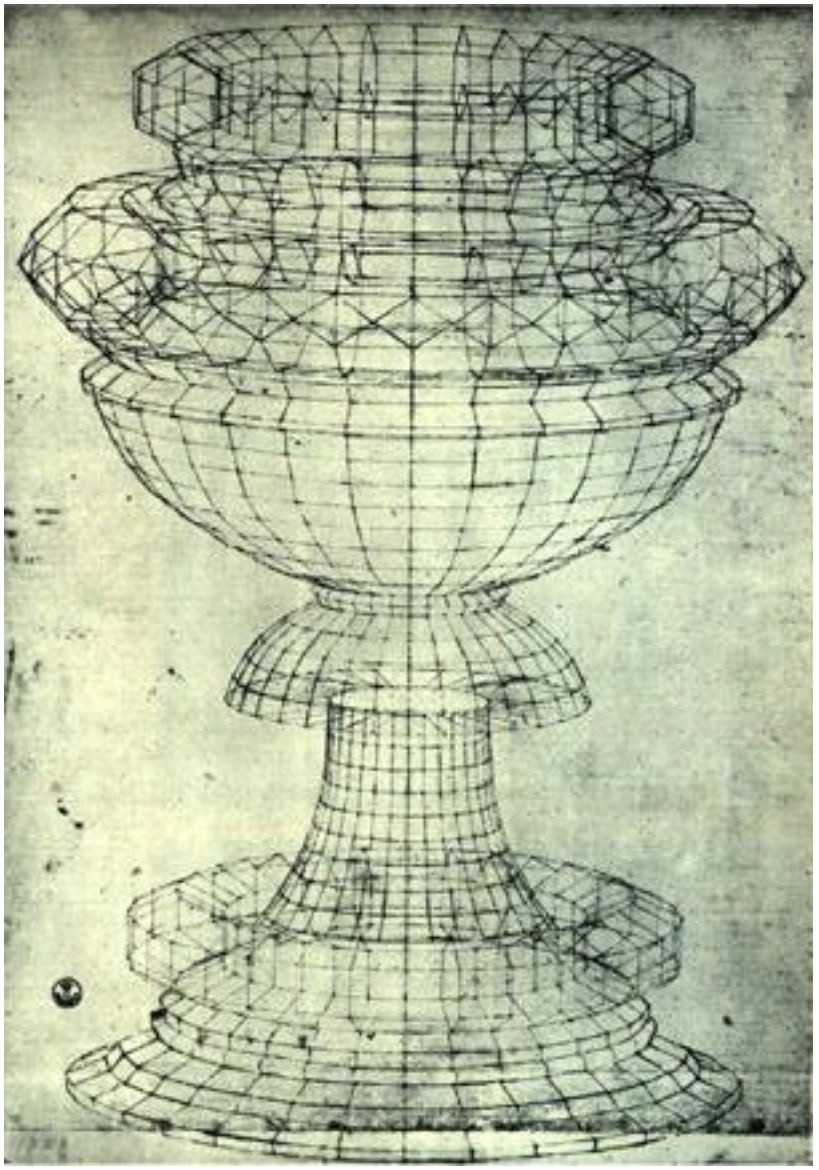
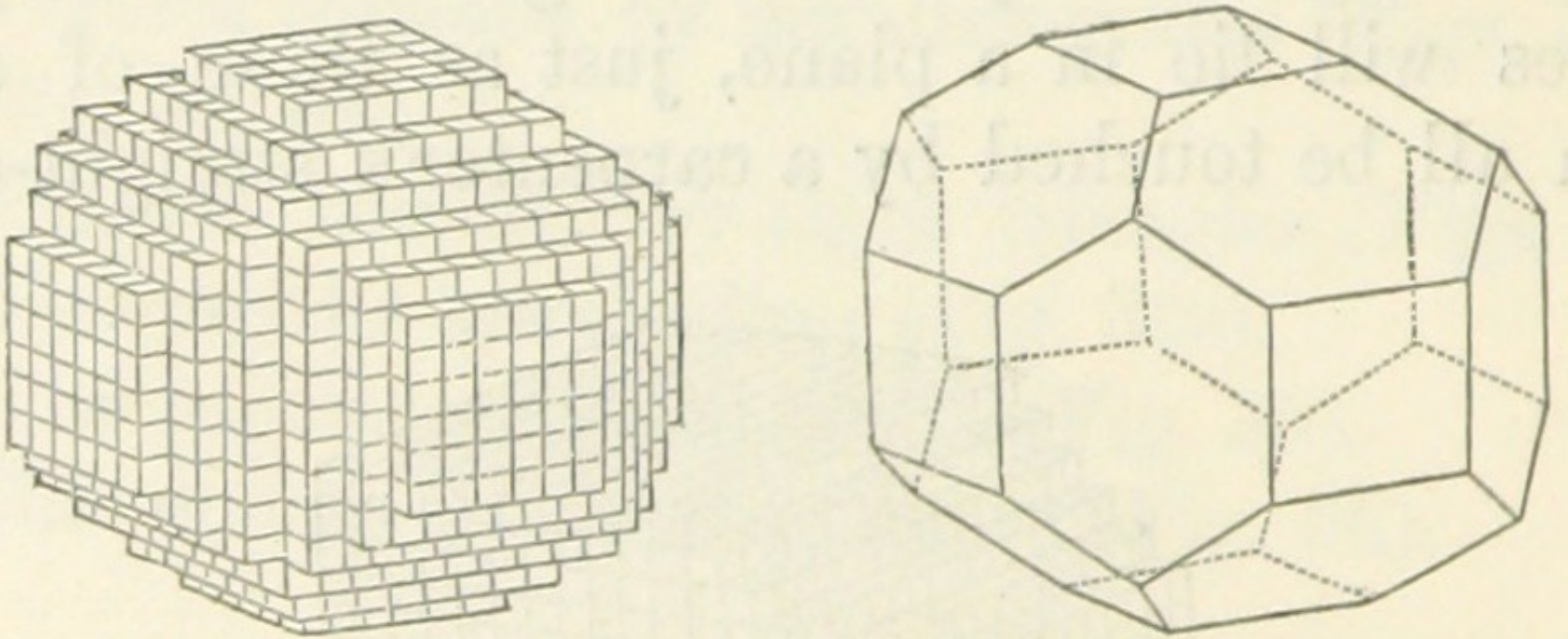
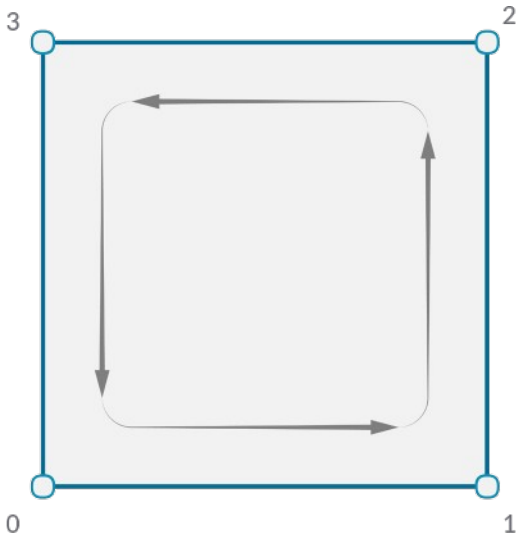


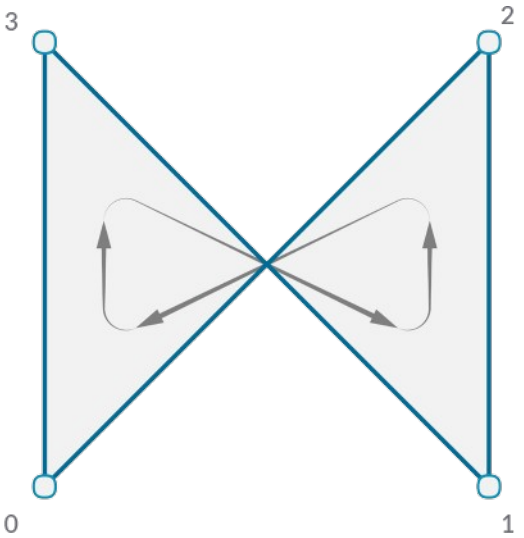


a *b*

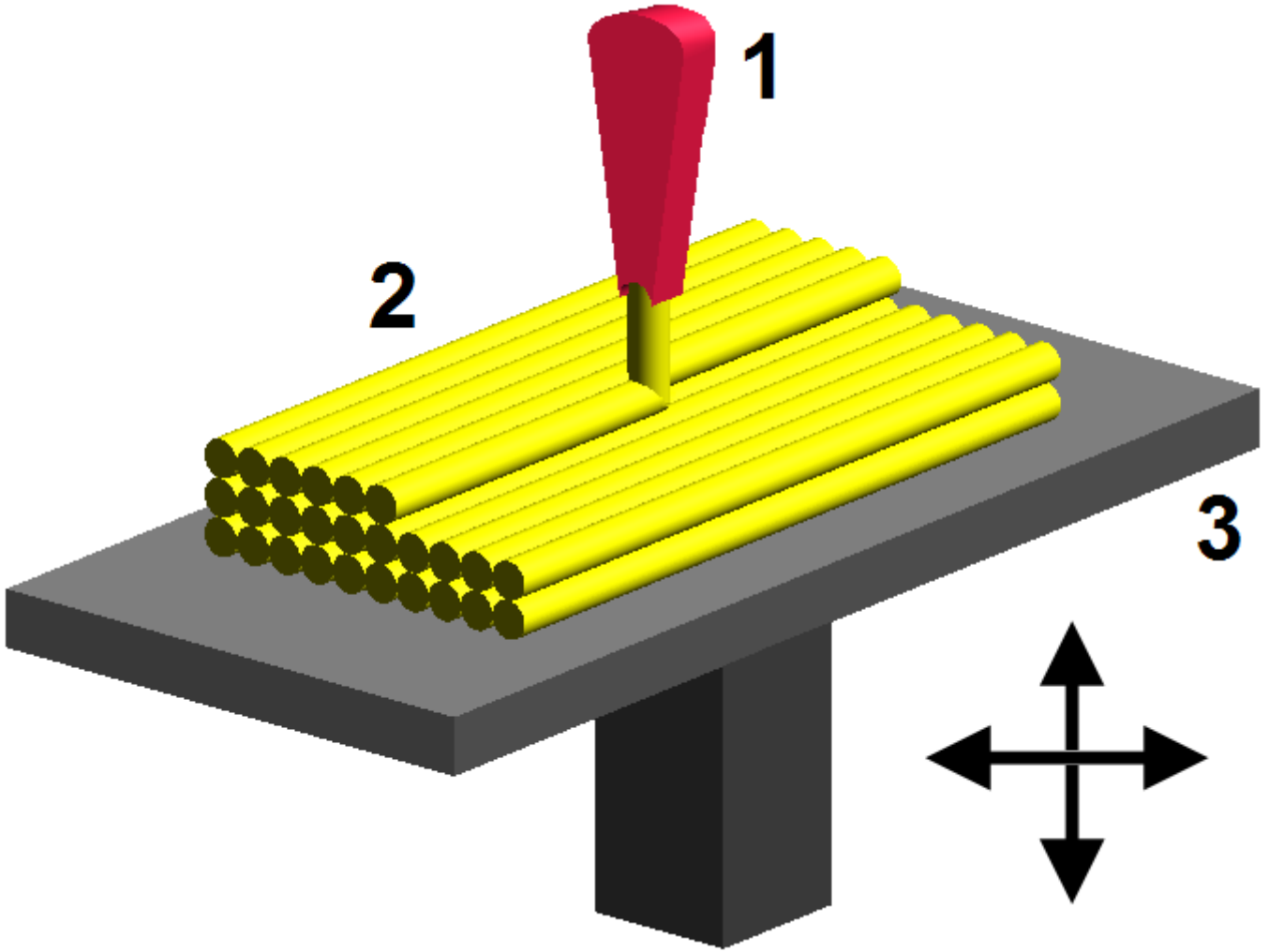
Fig. 39.

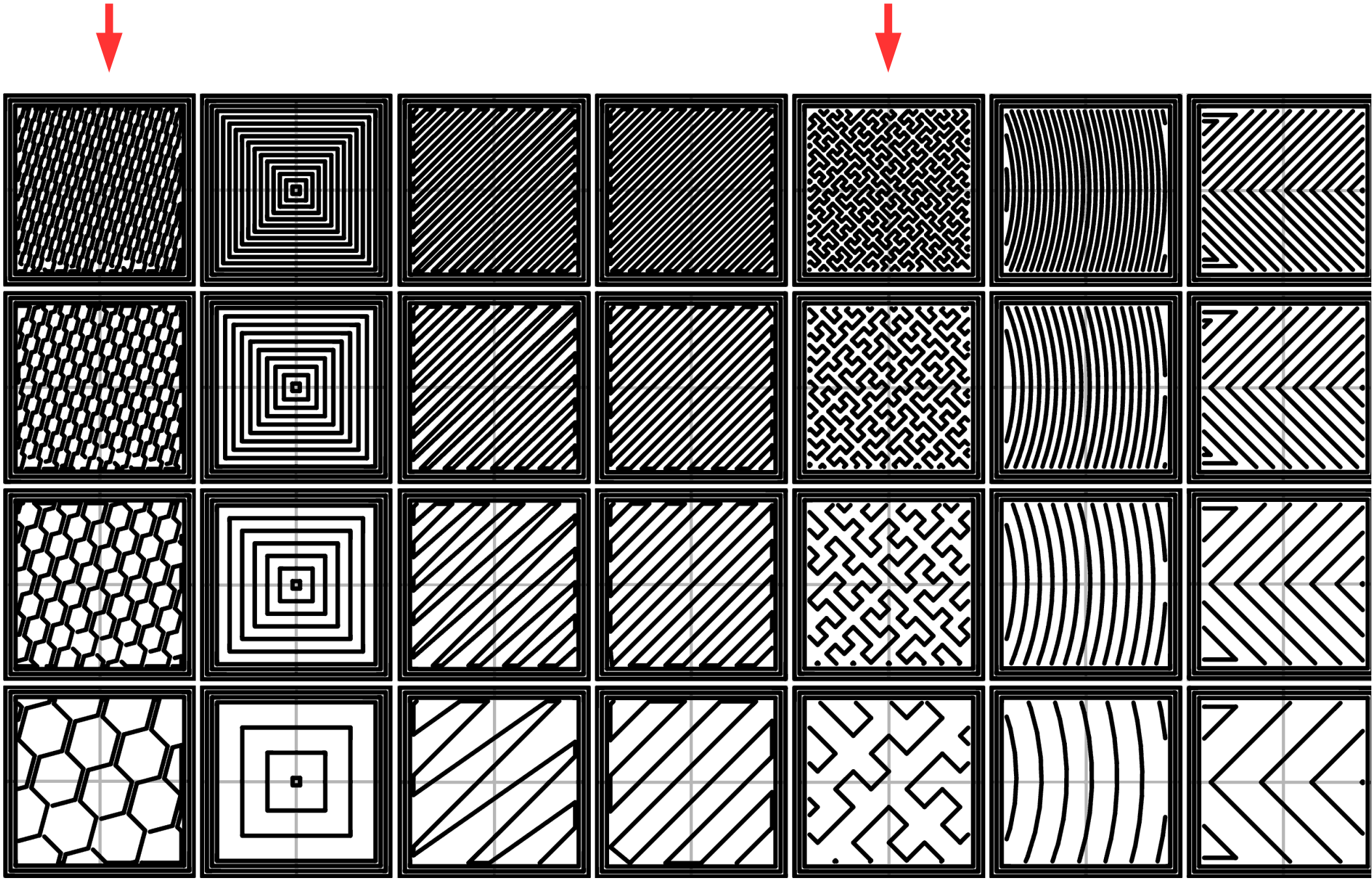


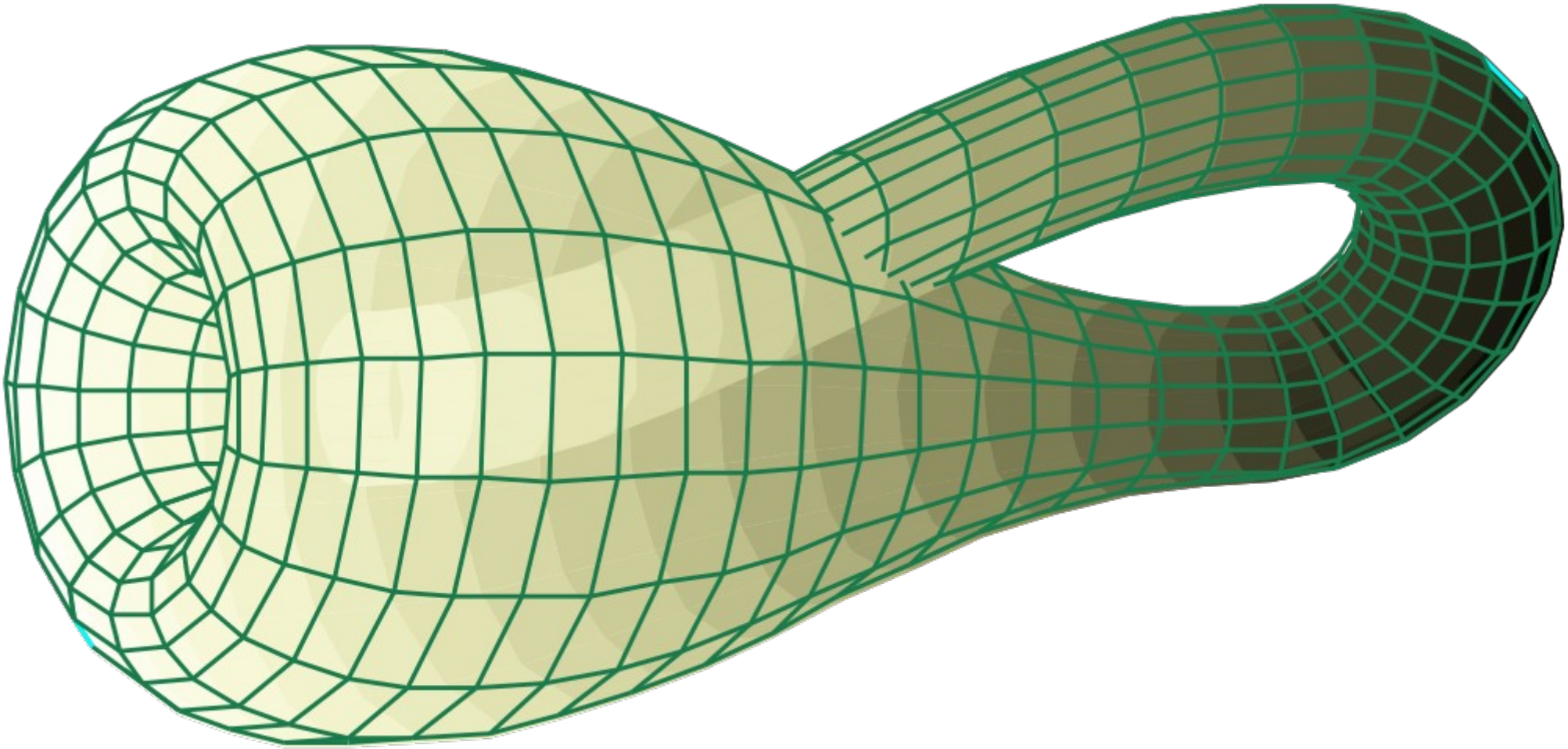
1



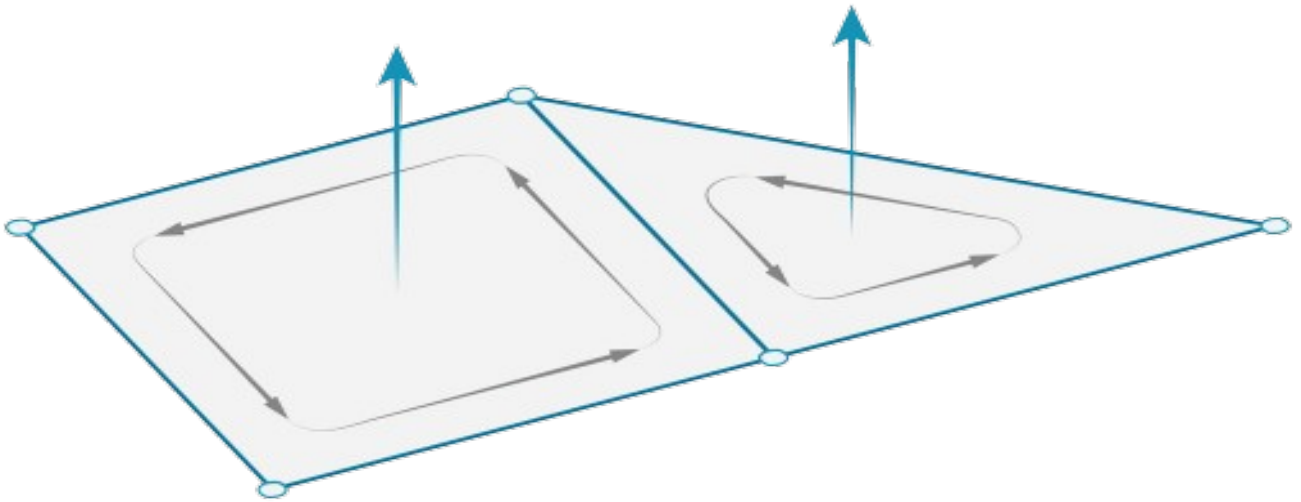
2

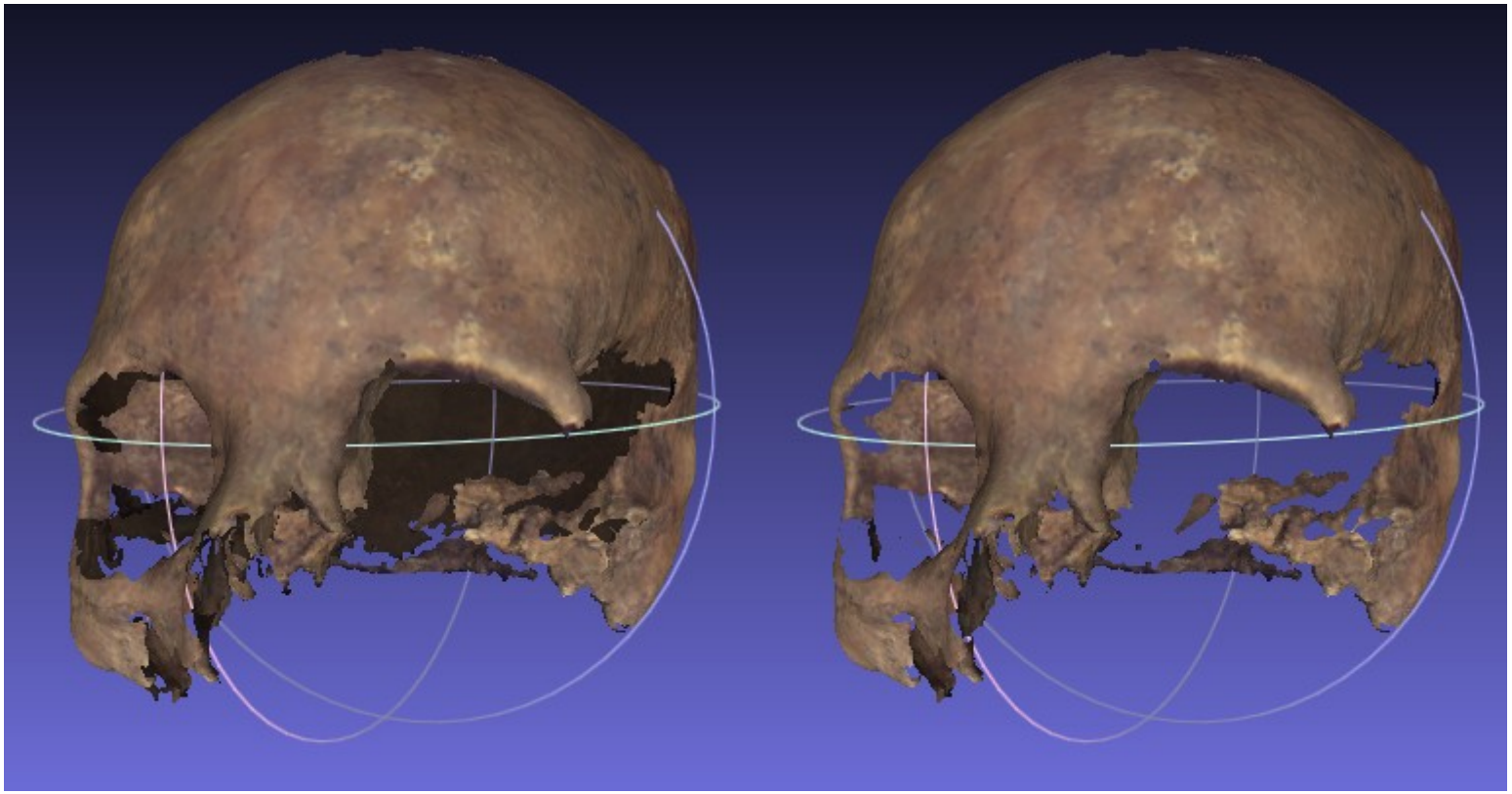

1
2
3

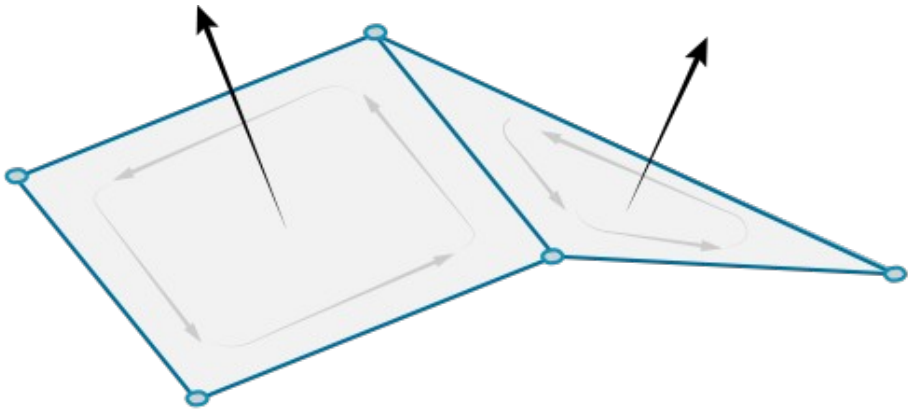




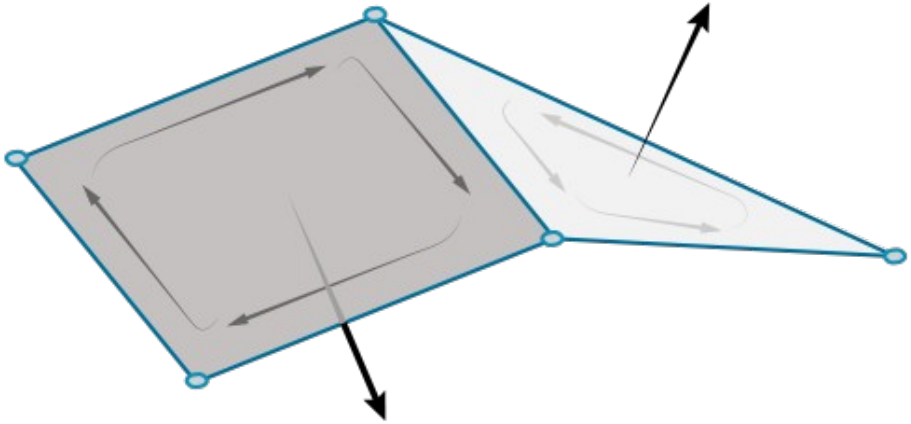
Orientable



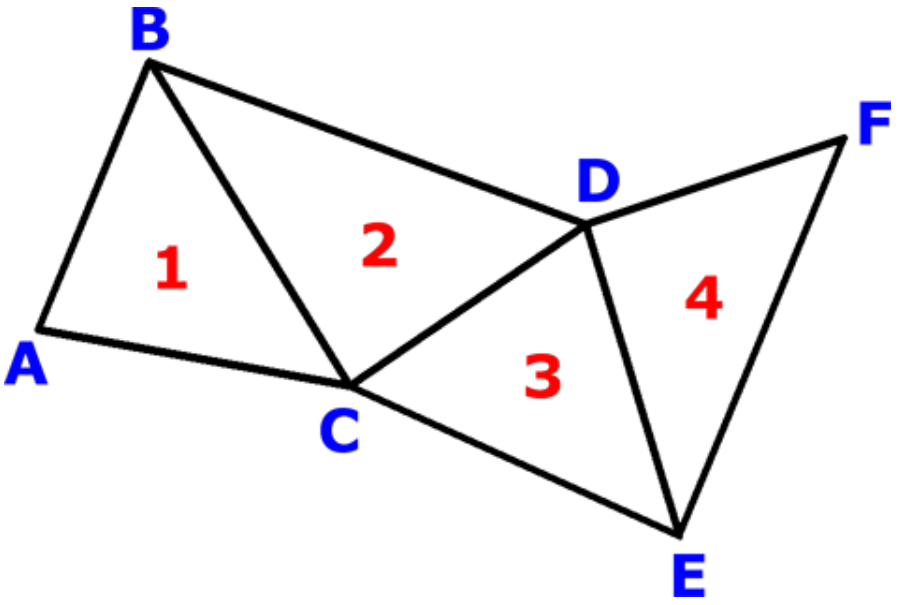


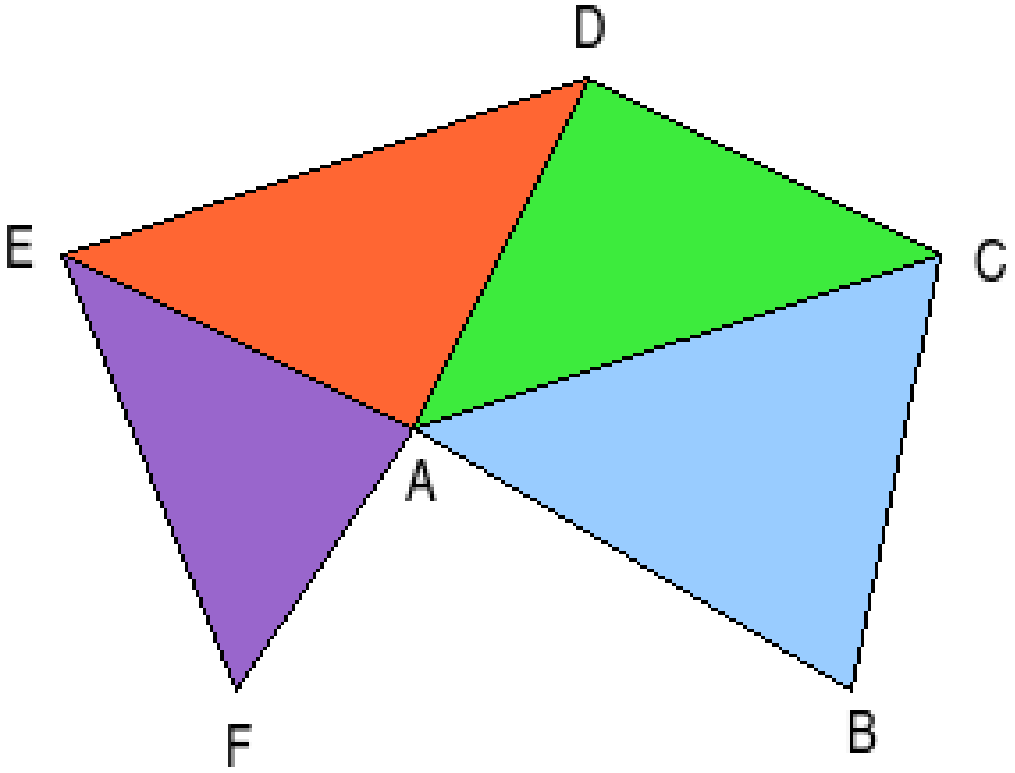


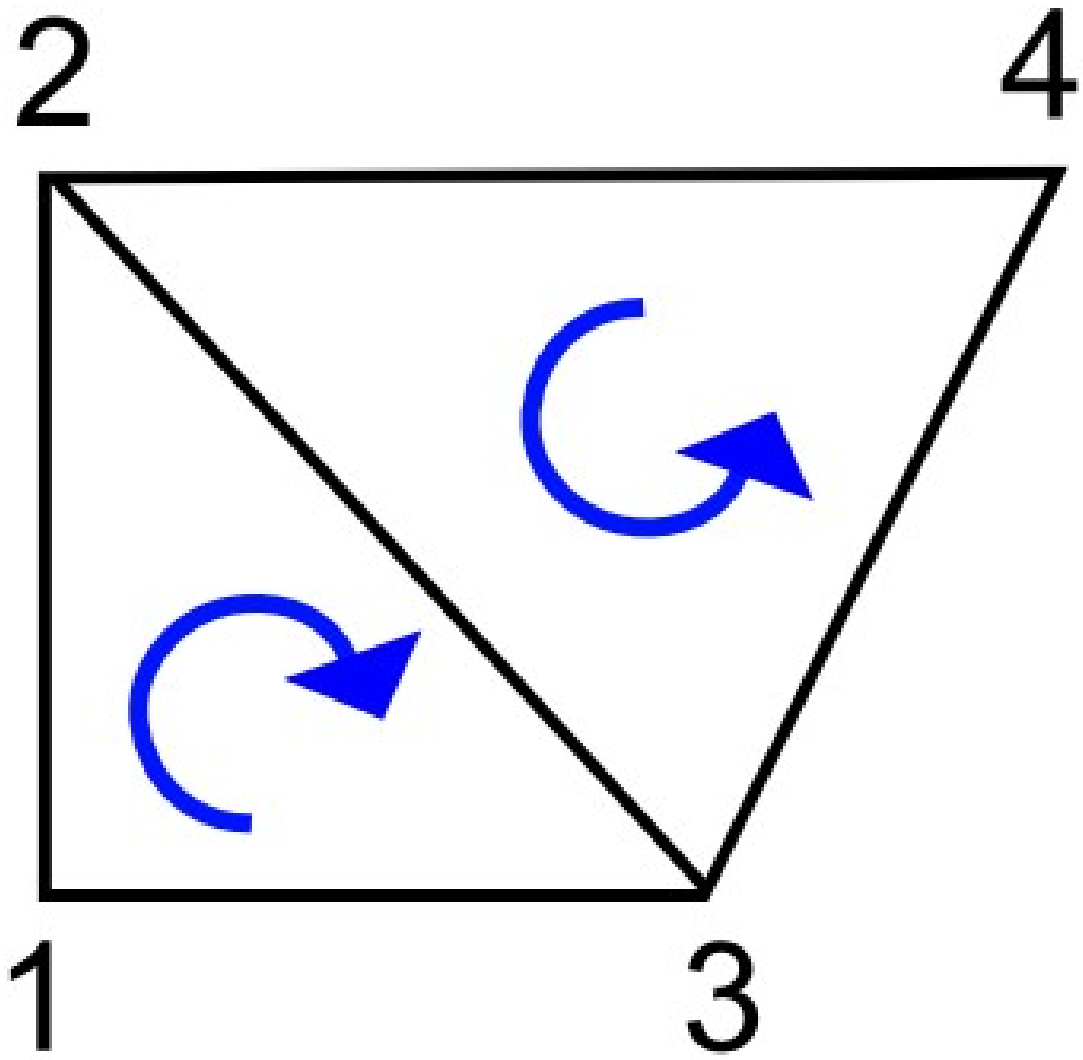
1



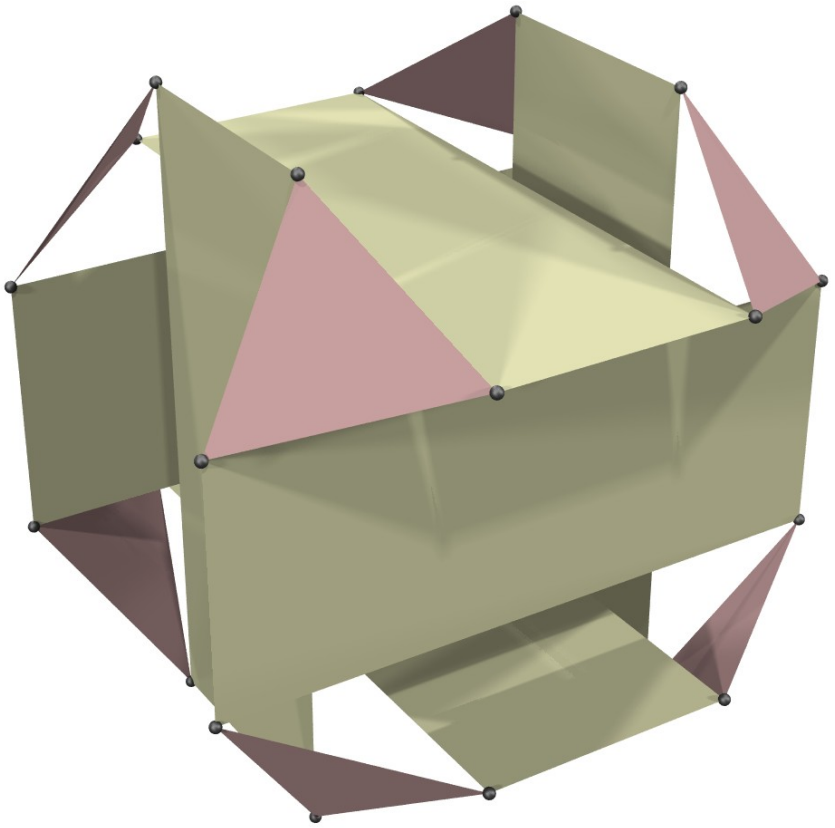
2

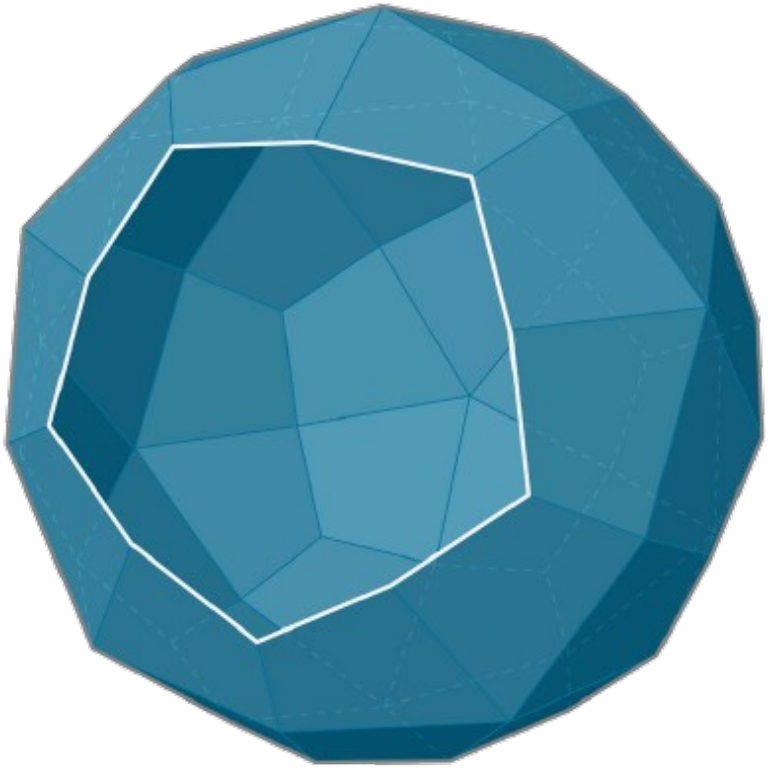

B
F
D
1
2
4
A
3
C
E


D
E
C
A
F
B

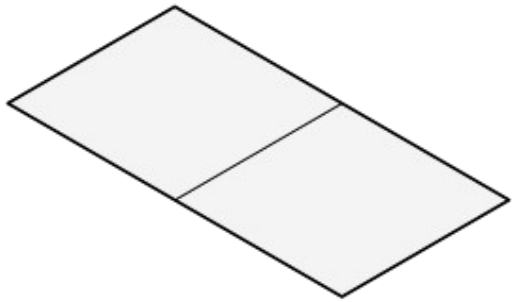

2
4
1
3

Closed/ Boundary-less

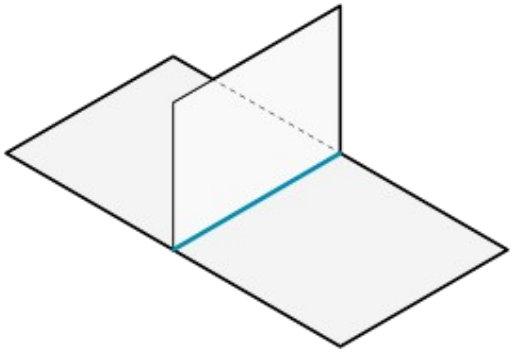




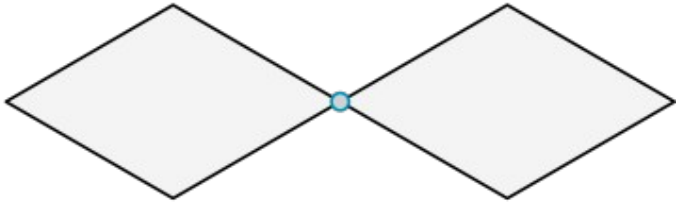
2-manifold



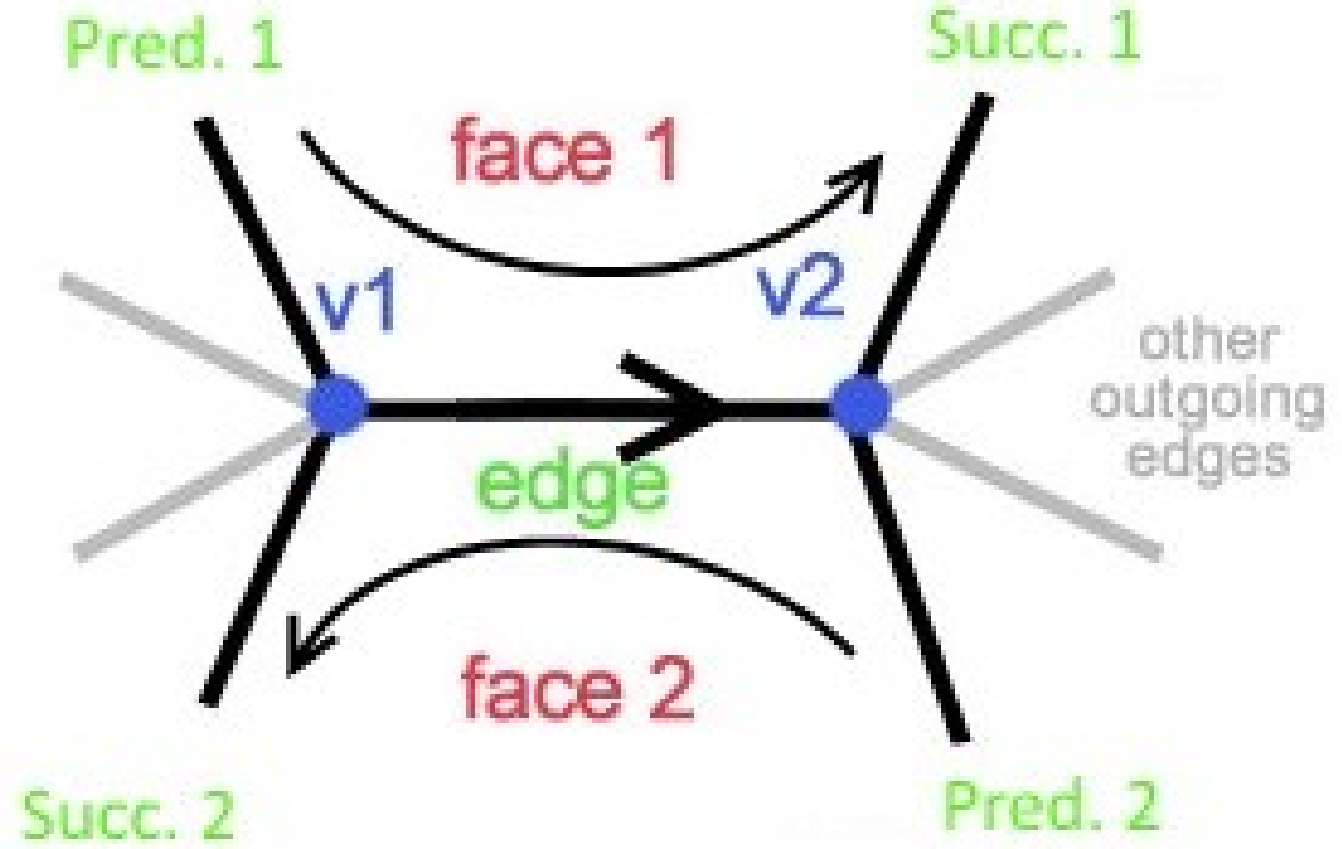
1



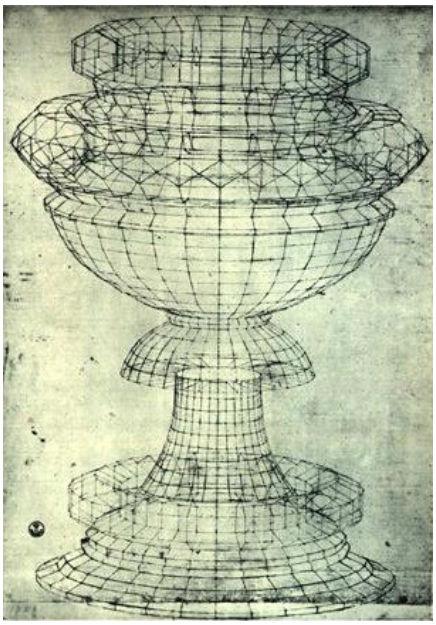
2



3



Winged Edge Structure



This wireframe study of a chalice was painted by the Italian Renaissance artist Paolo Uccello around 1460. Today 3D polygonal meshes are the dominant representation of 3-dimensional objects for a range of uses from Computer Graphics and finite element simulations to 3D printing. Whilst they have great flexibility in their ability to represent arbitrary objects they generally can only *approximate* the geometry of an object. If we're concerned about 3D printing then it's not enough to have just a bunch of polygons. They really do need to define a volume in space.

Image credit https://commons.wikimedia.org/wiki/File:Uccello,_Paolo_-_Perspective_Study_of_a_chalice.JPG

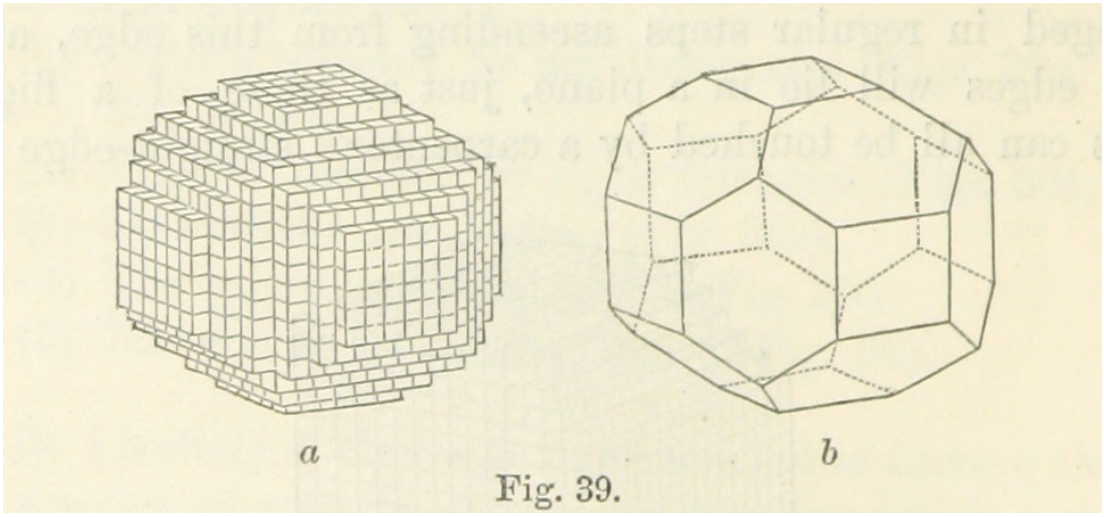
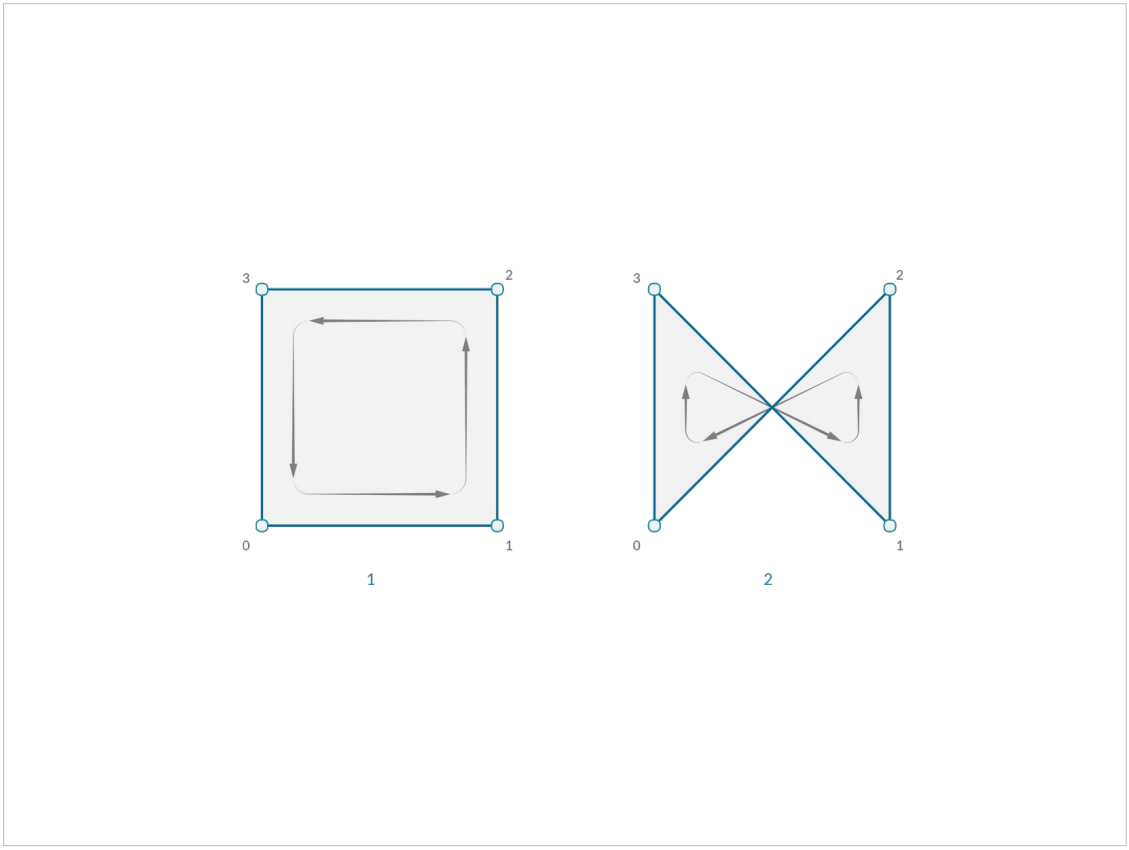


Fig. 39.

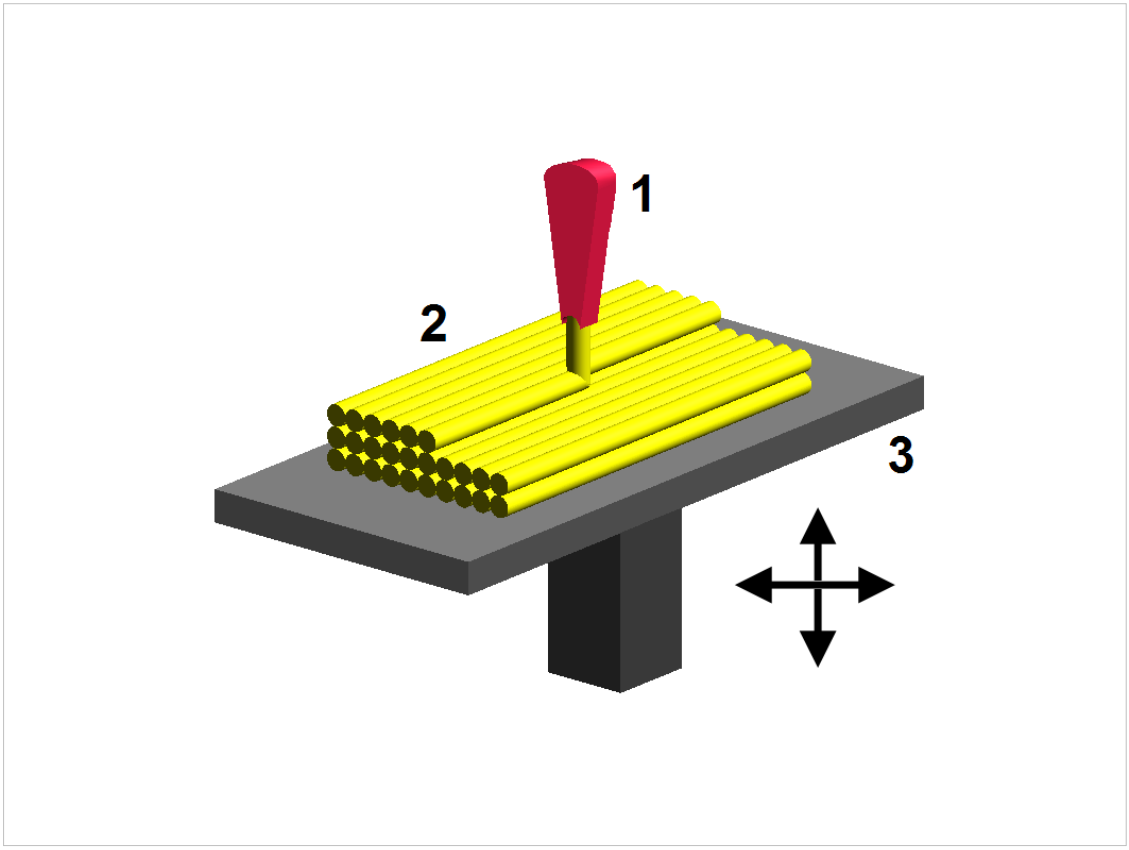
Although we could use a so-called voxel representation, as this illustration from a 19th-century mineralogy book shows, we'd need to store considerably more data than with just the polygon mesh alone.

Image credit
https://www.flickr.com/photos/britishlibrary/11103932105
Public domain



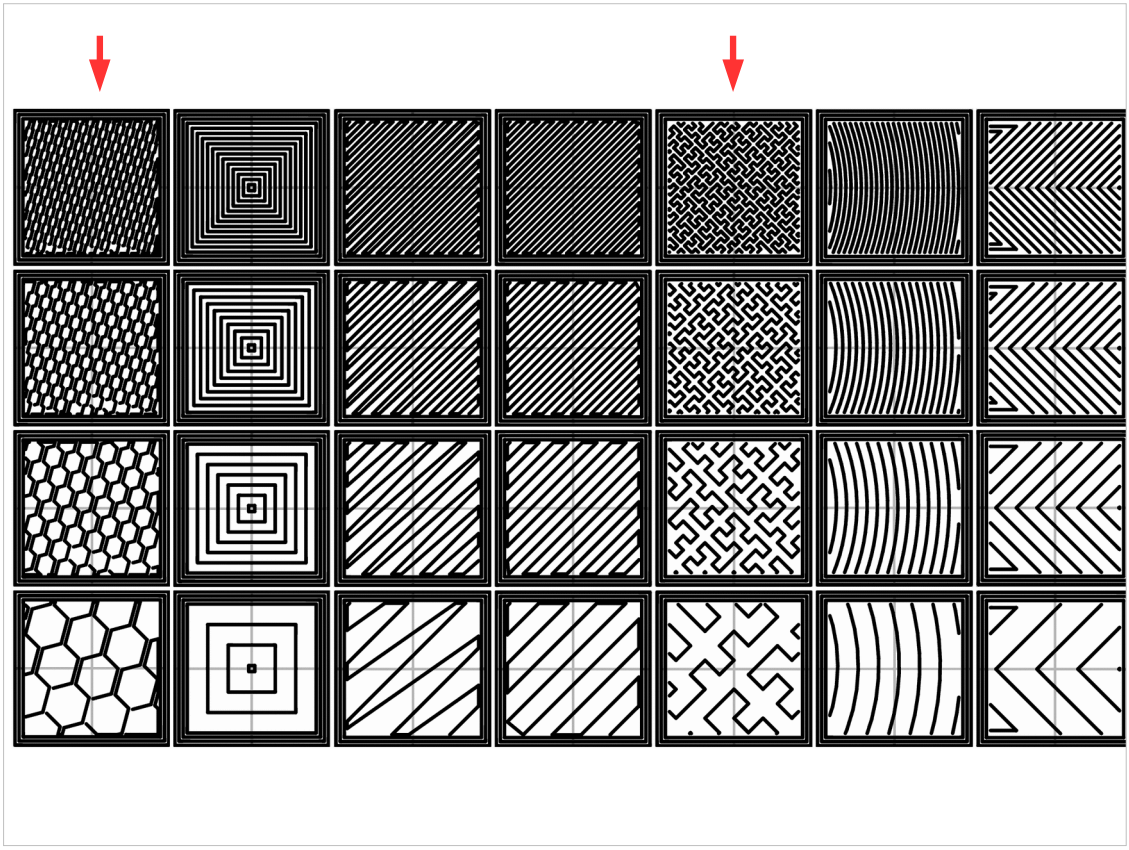
Given all we have is a bunch of flat objects to represent a volume, we need to ensure we follow rules that prevent us from making silly mistakes and defining nonsense meshes they don't define a volume. A basic constraint we can apply is to ensure that rather than arbitrary polygons we only allow triangles. This guarantees we only have planar polygons and avoids 'degenerate' polygons such as that shown on the right. (Let's ignore the possibility that the vertices might be co-linear for now!)

Image credit https://github.com/modelab/grasshopper-primer/tree/master/en



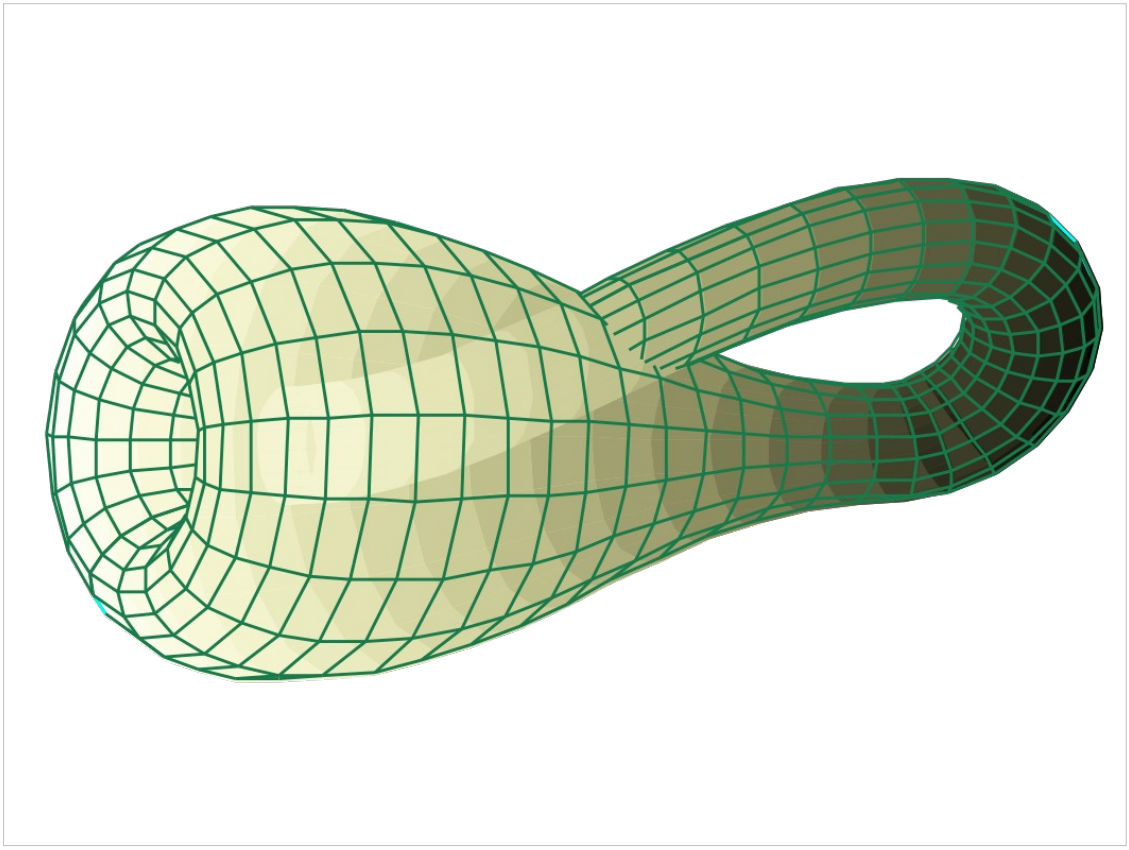
When you load your mesh into your 3D printer and press 'print', the software that runs your printer will first check your mesh to ensure that it really is printable, i.e. that it really defines a volume in space. It needs to do this since it will have to plan where to place material when it extrudes out filament to build up layers. If the mesh is known to enclose a volume completely then it can take slices through the mesh and fill in material from one edge of the boundary to the other.

Image credit
https://en.wikipedia.org/wiki/File:FDM_by_Zureks.png



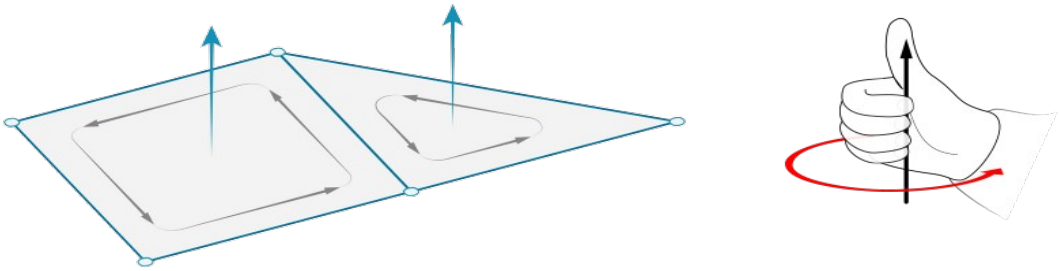
In practice a 3D printer will fill in a volume with a pre-defined pattern such as the honey-comb or Hilbert curve examples shown here. These allow the finished piece to maintain strength whilst still being light and reducing the amount of material that needs to be used. Of course if there is no boundary on the other side then we'll encounter a problem. So how then does the 3D printing software know just from the set of triangles which meshes are printable?

Image credit
http://manual.slic3r.org/expert-mode/infill



Consider an object that definitely cannot be printed – the Klein bottle. The reason is not just that the mesh is self-intersecting. Indeed, unless the triangles forming the self-intersecting part share edges, any check would depend on the particular coordinates of the vertices. Intuitively the property of being printable should be invariant of the particular vertex coordinates. Although it looks on first glance like it could contain a printable volume a quick check reveals the property it lacks is having an inside and an outside – it's a non-orientable surface.

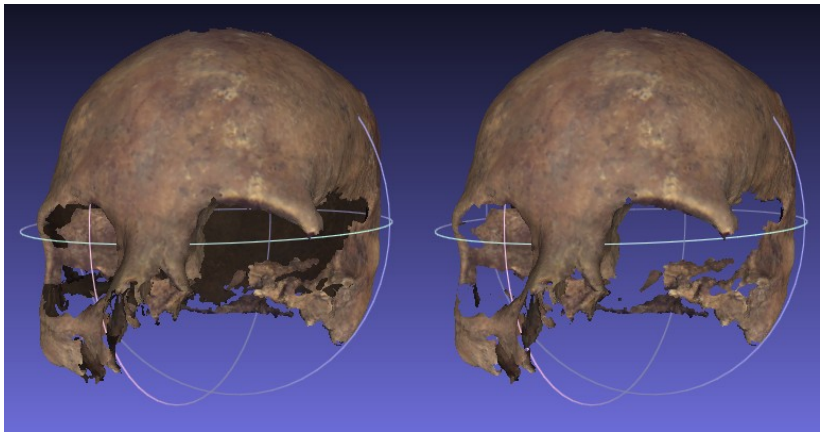
Orientable



This is our first criterion for any surface to be 3D-printable. How we can trivially check this property?

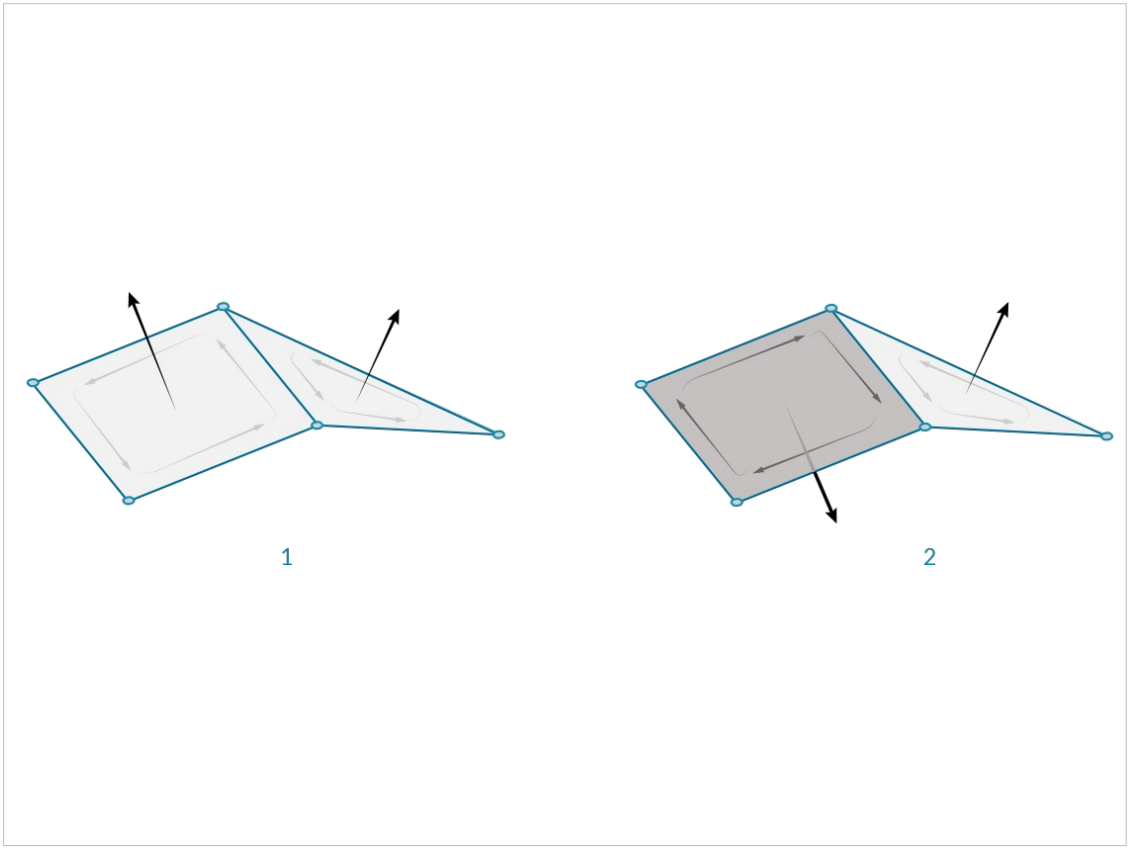
There is a convention in computer graphics libraries called the “right hand rule” that if vertices are ordered in an anti-clockwise direction, outward facing the surface normal is along the direction the thumb points. This is known as the 'winding order' and corresponds to the usual interpretation of vector cross products.

Image credit https://github.com/modelab/grasshopper-primer/tree/master/en



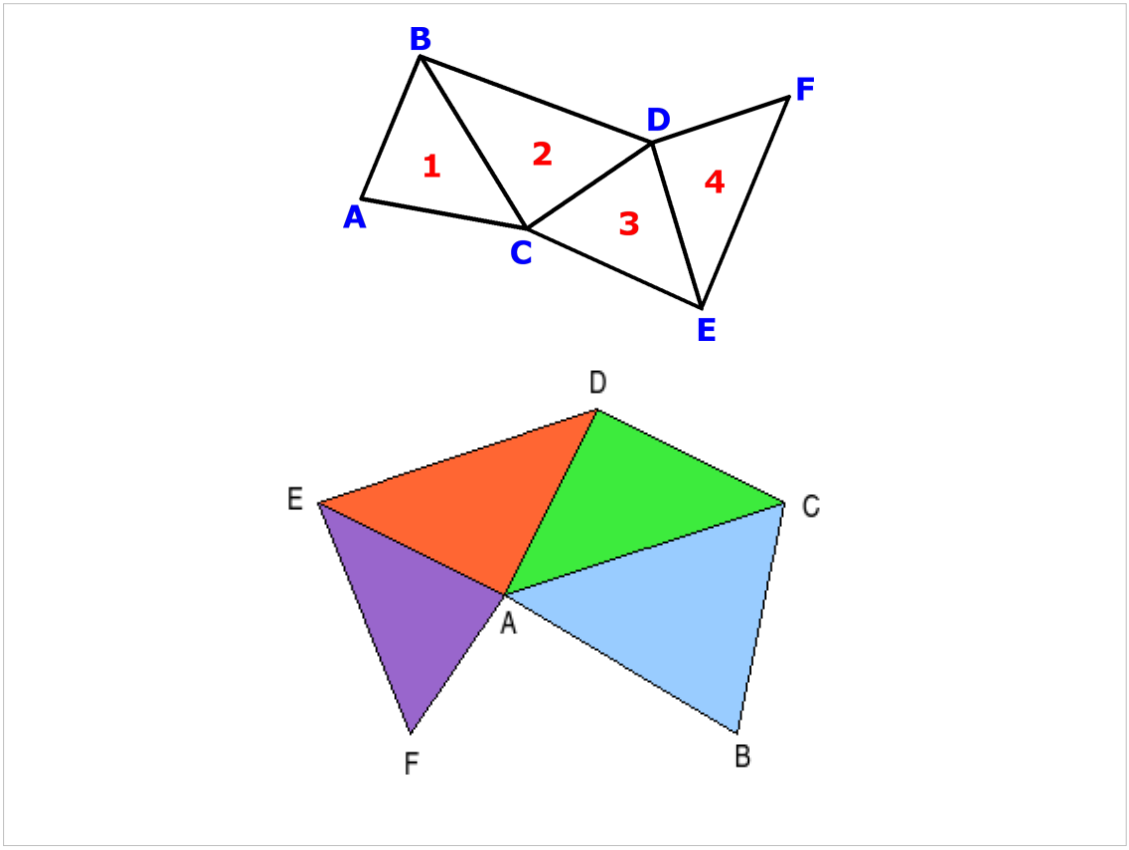
Although it is usually ambiguous which of two ways a surface normal points, using a convention like the 'right-hand rule' makes life easier in computer graphics when we're working with meshes that represent solid objects since we can easily perform so-called 'back face culling' to ignore any surfaces that point away from the camera (such as the right image here).

Image credit
https://en.wikipedia.org/wiki/File:Back_face_culling_skull_example.png



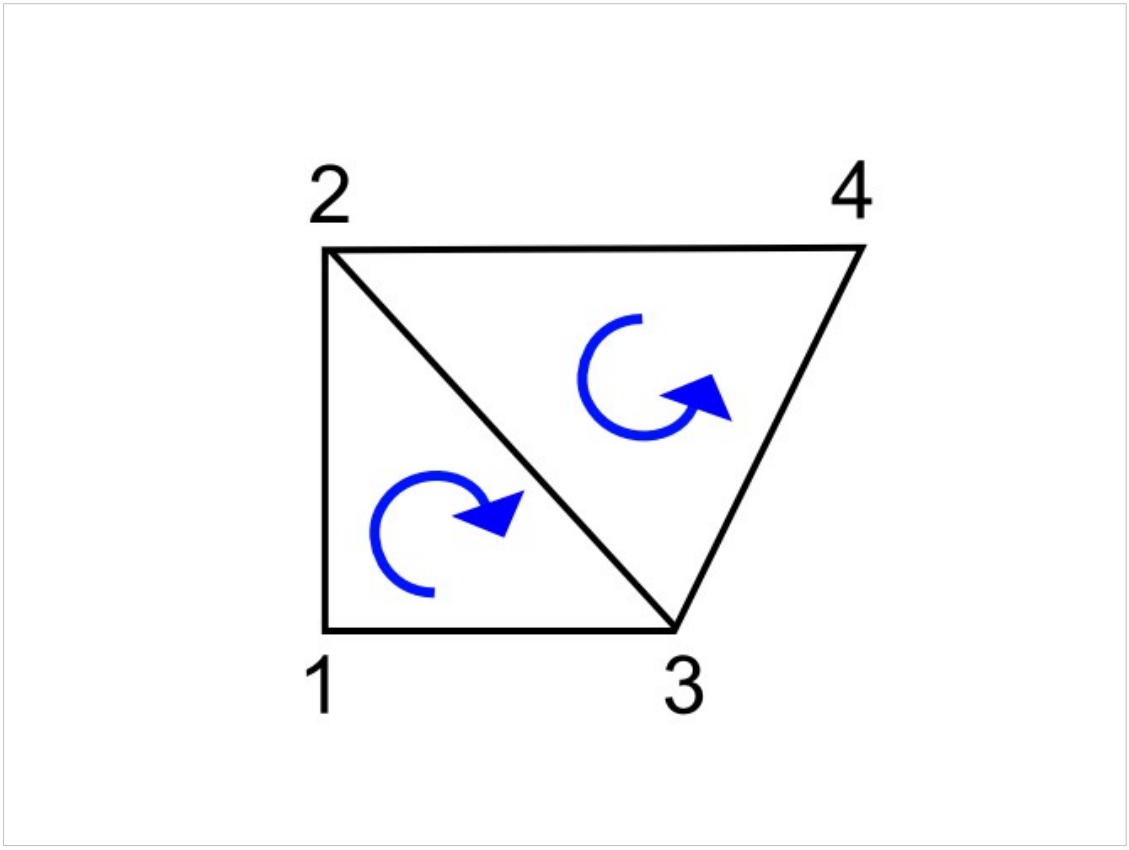
If we want to check that our mesh is orientable it turns out this same rule can be used. Since we want all the normals to point away from the enclosed volume we just check that we can assign a winding order to each triangle that ensures the winding orders on each side of a common edge are in opposite directions – a property that is easily checked from the definition.

Image credit
https://github.com/modelab/grasshopper-primer/tree/master/en



If we're dealing with very large triangle meshes we can speed up the time to process computations such as coordinate transformations by ensuring we only compute a value for a given vertex once. Graphics libraries allow us to reuse two previously specified vertices so that adding new vertices forms either a triangle strip (top) or triangle fan (bottom). In the latter case the initial vertex is taken to be the centre of the fan.

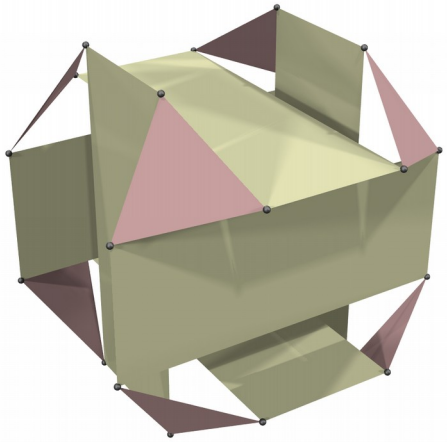
Image credits
https://en.wikipedia.org/wiki/File:Trianglefan.png
CC-BY-SA-3.0
https://en.wikipedia.org/wiki/File:Triangle_Strip.png
Public Domain

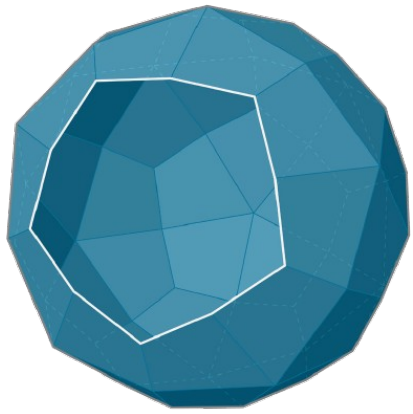


However, there is a problem with naively generating triangles from the strip representation. If you notice carefully the winding order actually alternates as you go along the strip. This is why it's important that the 3D printing software doesn't place any trust in the vertex order specified in the original file but actually checks for itself.

Image credit
https://en.wikipedia.org/wiki/File:Triangle_Strip_In_OpenGL.svg

Closed/ Boundary-less





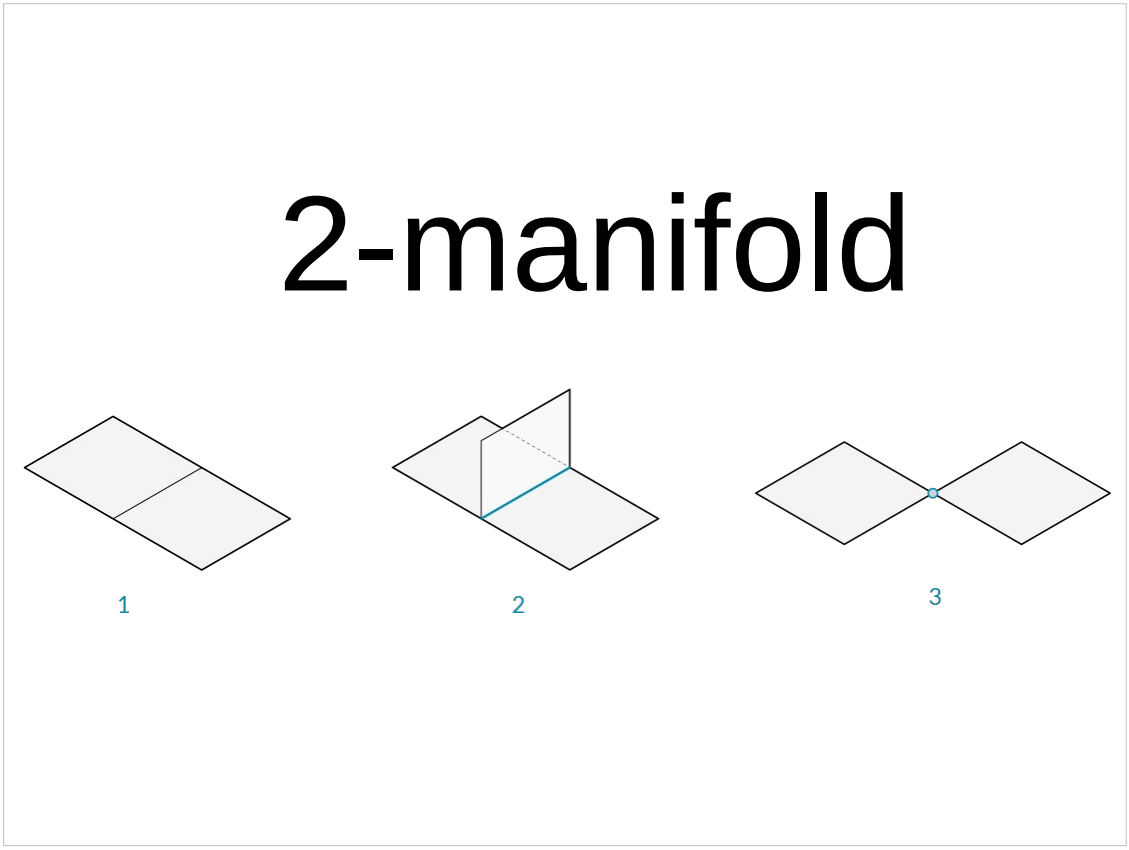
However, orientability is not the complete story, as this example shows. We can assign winding orders to each of the surfaces in these meshes such that all the normals point outwards though there are clearly some surfaces that do not properly enclose a volume. The property these meshes lack is that of being closed.

To check if a mesh is closed all edges must only have exactly two adjacent faces. You might like to check that the Klein Bottle is indeed closed.

Image credit
https://commons.wikimedia.org/wiki/File:Polyhedron_with_no_vertex_visible_from_center_(open).png

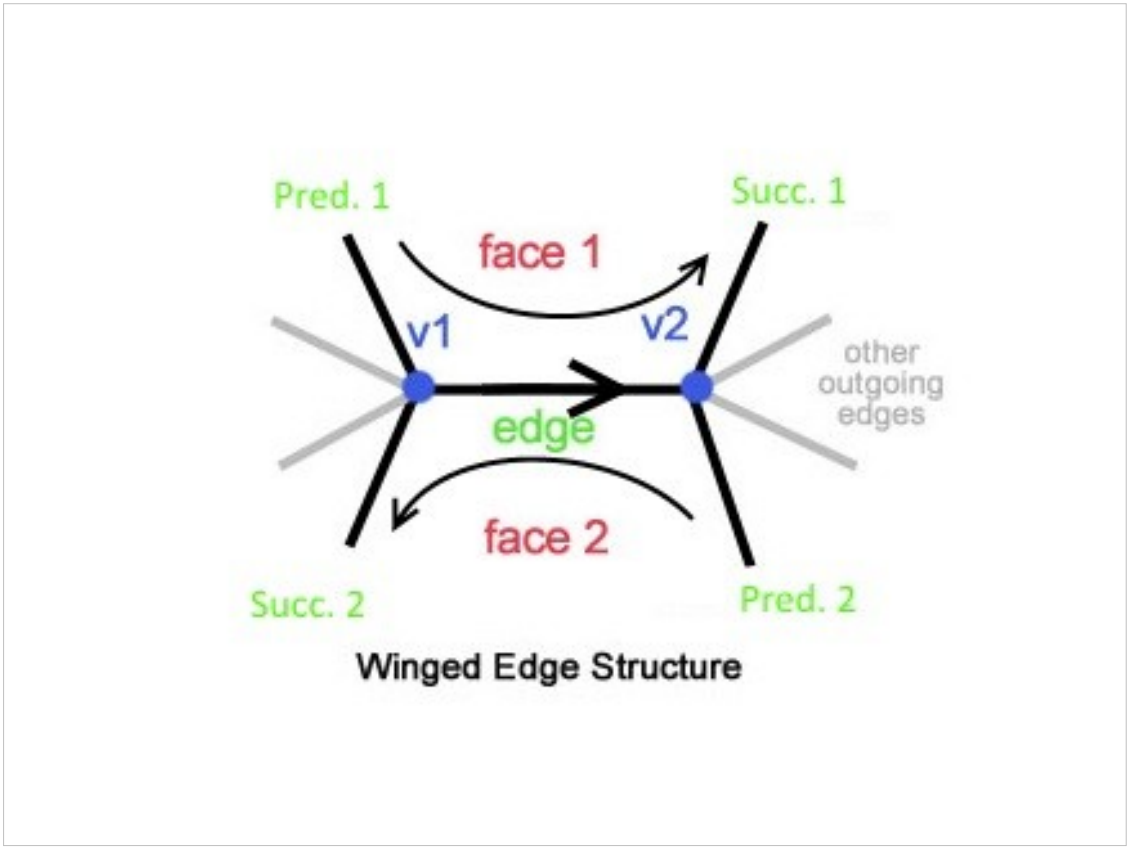
Public Domain

https://github.com/modelab/grasshopper-primer/tree/master/en



A surface with one or two faces adjacent to an edge (example left) is called manifold. A surface with three or more faces adjacent to an edge or where surfaces meet at a point is non-manifold. Any 2-dimensional surface (more commonly known as 2-manifold), embedded in 3D space that has the two properties of being orientable and closed (or boundary-less) will be 3D printable.

Image credit
https://github.com/modelab/grasshopper-primer/tree/master/en



Since both the rules depend on properties of edges it is natural to use a data structure that neatly wraps up these rules, called the 'Winged-Edge Data Structure'. The structure consists of eight items: the two vertices, two adjacent faces each with a predecessor and successor edge. We leave it as an exercise to verify that checking a mesh for 3D print-ability is equivalent to converting the mesh to this data structure.

Image credit (modified version of)
https://en.wikipedia.org/wiki/File:Mesh_we2.jpg
CC-BY-SA-3.0