Advanced Computer Graphics

Lecture-02 3D geometry

Tzung-Han Lin
National Taiwan University of Science and Technology
Graduate Institute of Color and Illumination Technology
e-mail: thl@mail.ntust.edu.tw

TAIWAN TECH National Taiwan University of Science and Technology

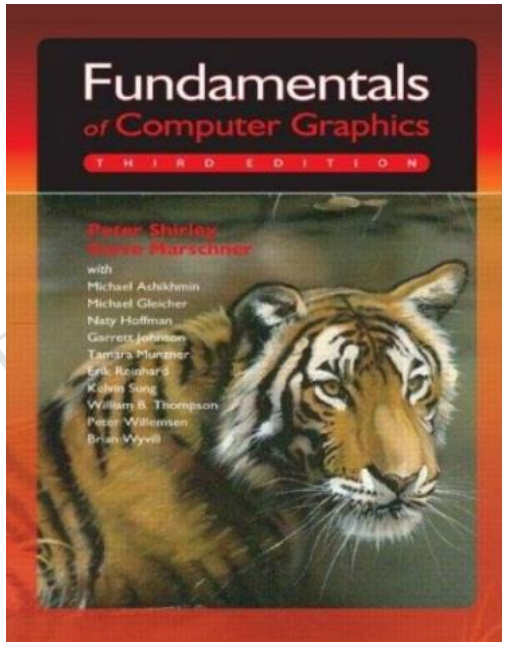
CIT 色彩與照明科技研究所 Graduate Institute of Color and Illumination Technology

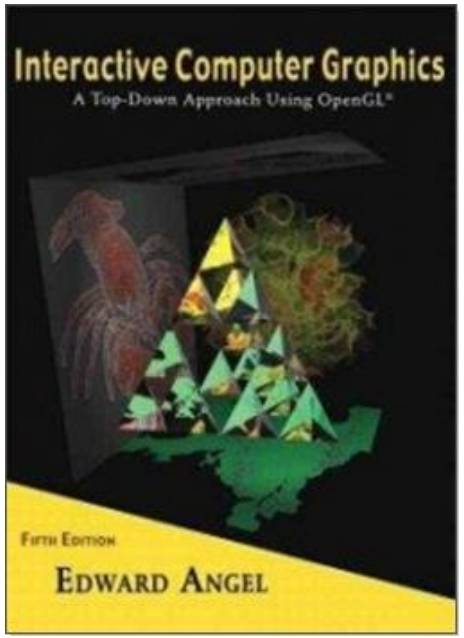
3D Geometry

- Simple 3D processing in Graphics
- Geometrical / topological definition
- Graphics pipeline
- 3D data format (STL/OBJ/PLY)

Content from textbook

- Fundamentals of computer graphics: chapter 3.
- Interactive computer graphics a top-down approach using OpenGL: Chapter 1&3&4.
- Online resource.





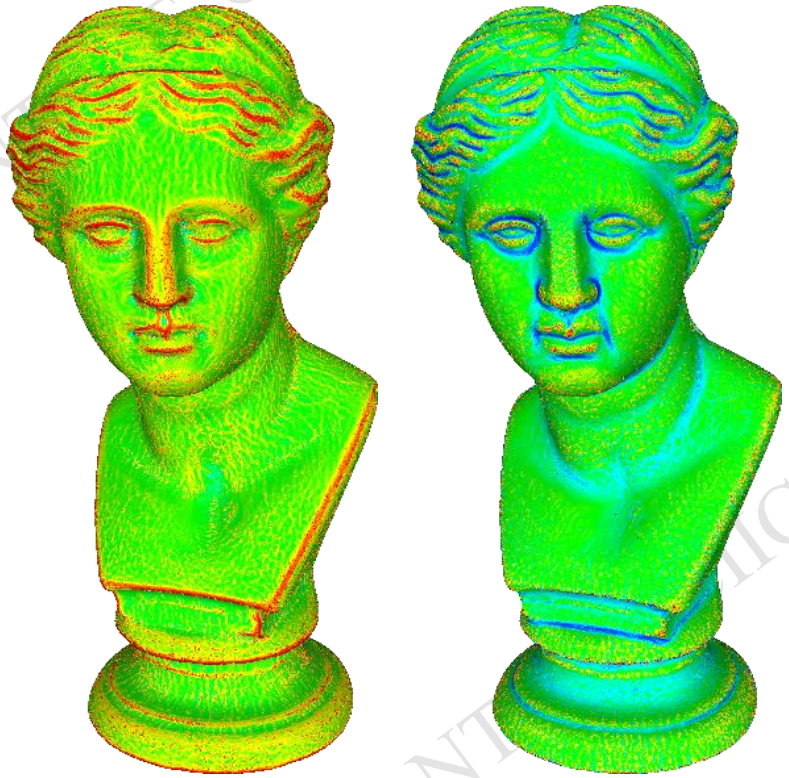
Data format in computer graphics-1

- Mathematical definition
- 2D / 3D point
  - Not occupy an area, no volume, and zero in size.
  - Not able be divided, no able to merge.
- 2D / 3D line (curve)
  - No width, but has specific length.
  - Able be divided into infinite lines, and able to merge.
  - 1D curvature (regional differential).
- 2D / 3D plane (surface)
  - No thickness, no volume, but has specific area.
  - 2D curvature (2x2 tensor).

Data format in computer graphics-2

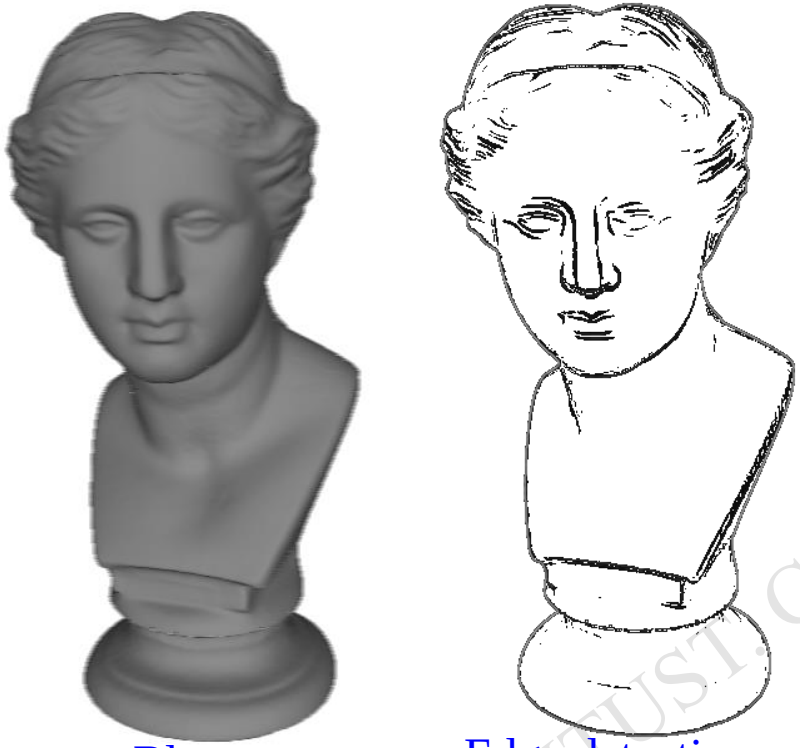
- Computer rendering on screen
- 2D / 3D point
  - The minimum area is 1 pixel on display.
- 2D / 3D line (curve)
  - Minimum width is 1 pixel on display.
  - 1D curvature (finite difference).
- 2D / 3D plane (surface)
  - Could have thickness.
  - Consist of finite lines (or points).
  - 2D curvature (finite difference).

Comparison between vector and pixel



First eigenvalue　　Second eigenvalue

Surface curvature
estimation from vectors



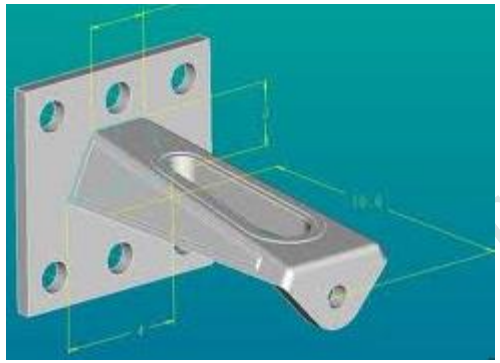
Blur
Gaussian filter

Edge detection
(curvature)

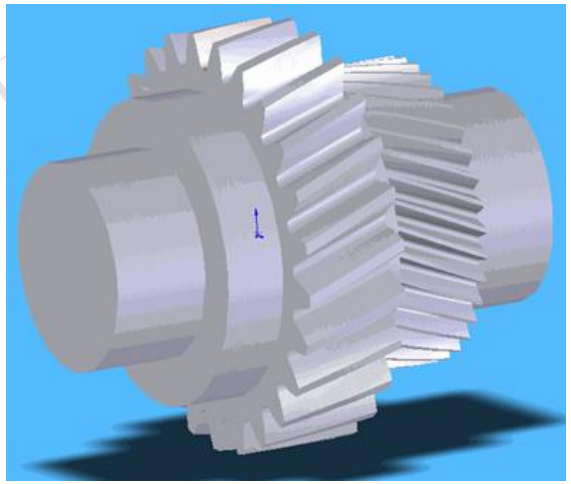
Process on frame-buffer
(or so-called image processing)

Data definition in “Solid model” field

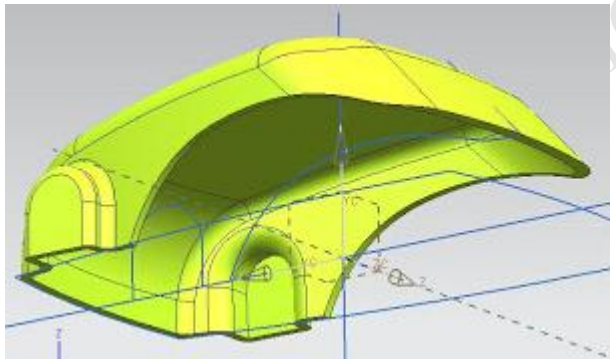
- Represent a volume (consist of enclosed face).
- Physical topology.
- Able to process boolean, division, merge.
- Physical property: weight, inertia, un-isotropic material, density, et. al.



ProE



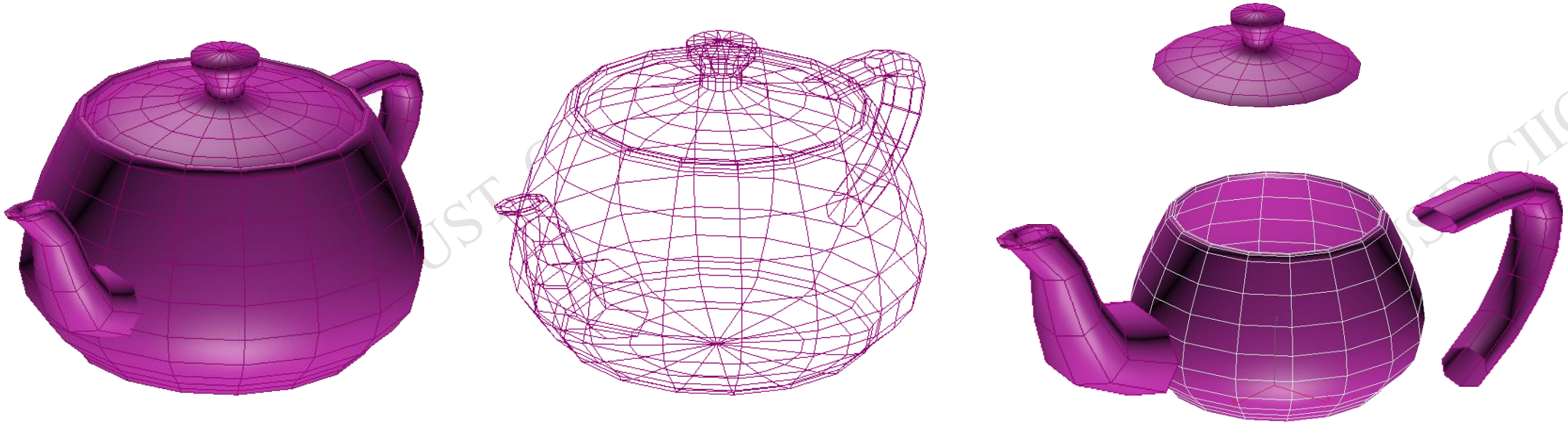
Solidworks



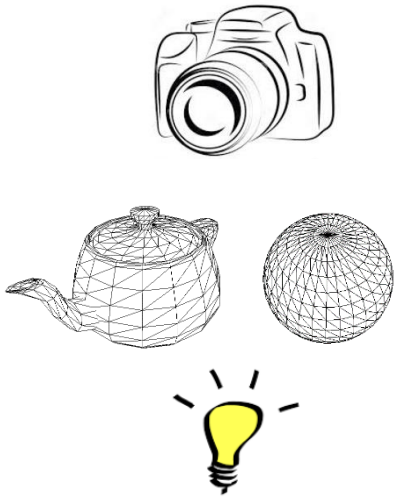
UnitGraphics / CATIA

Vector data in “Computer graphics” field

- Geometry base including vertex, line(curve), surface.
- Consider ONLY the “surface” (very thin shell).
- Usually do NOT have property definition on physical behavior.



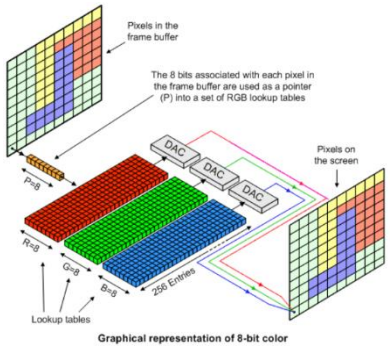
Rendering 3D scene



Models, light, camera, and screen

- Transform
- Illuminate
- Transform
- Clip
- Projection
- Viewport
- Rasterize
- Buffering

Processing



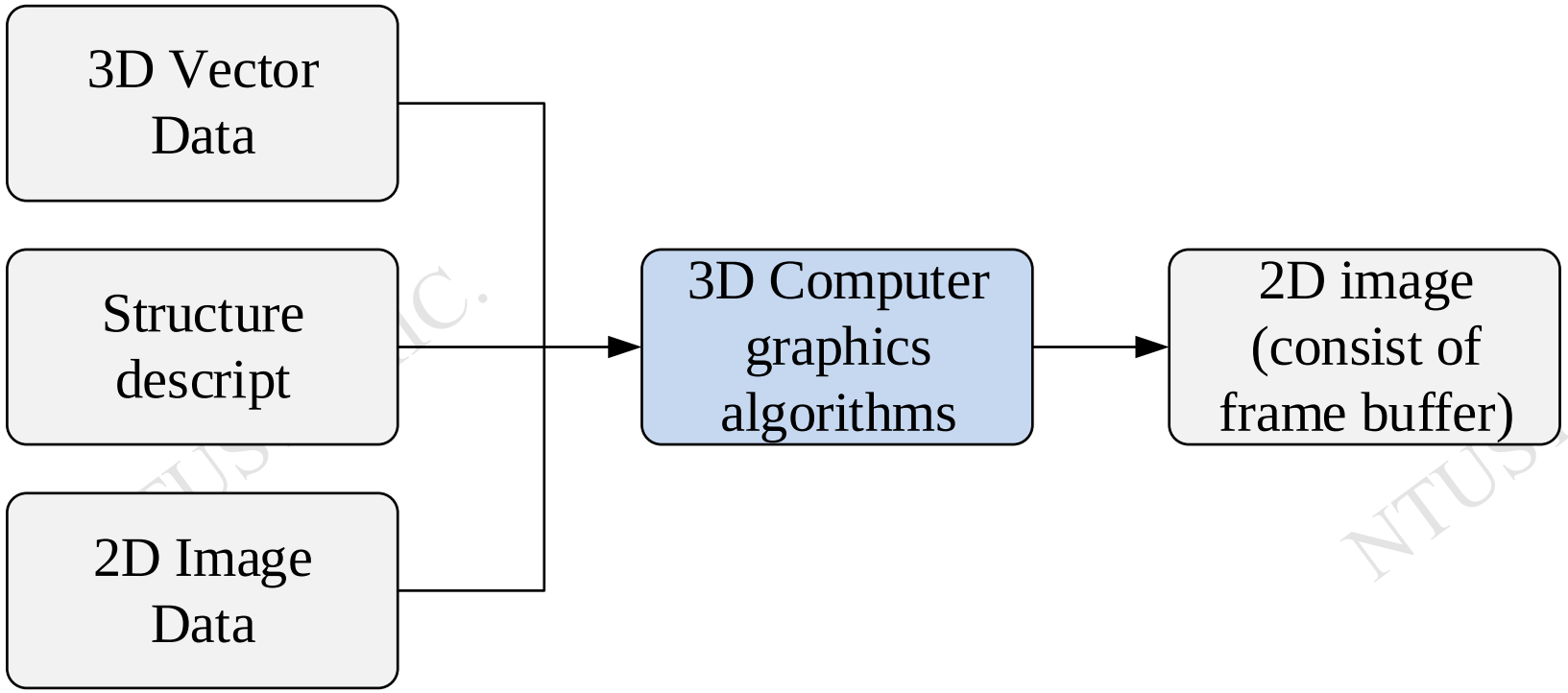
Frame buffer

Display

http://www.clivemaxfield.com/diycalculator/popup-h-console.shtml

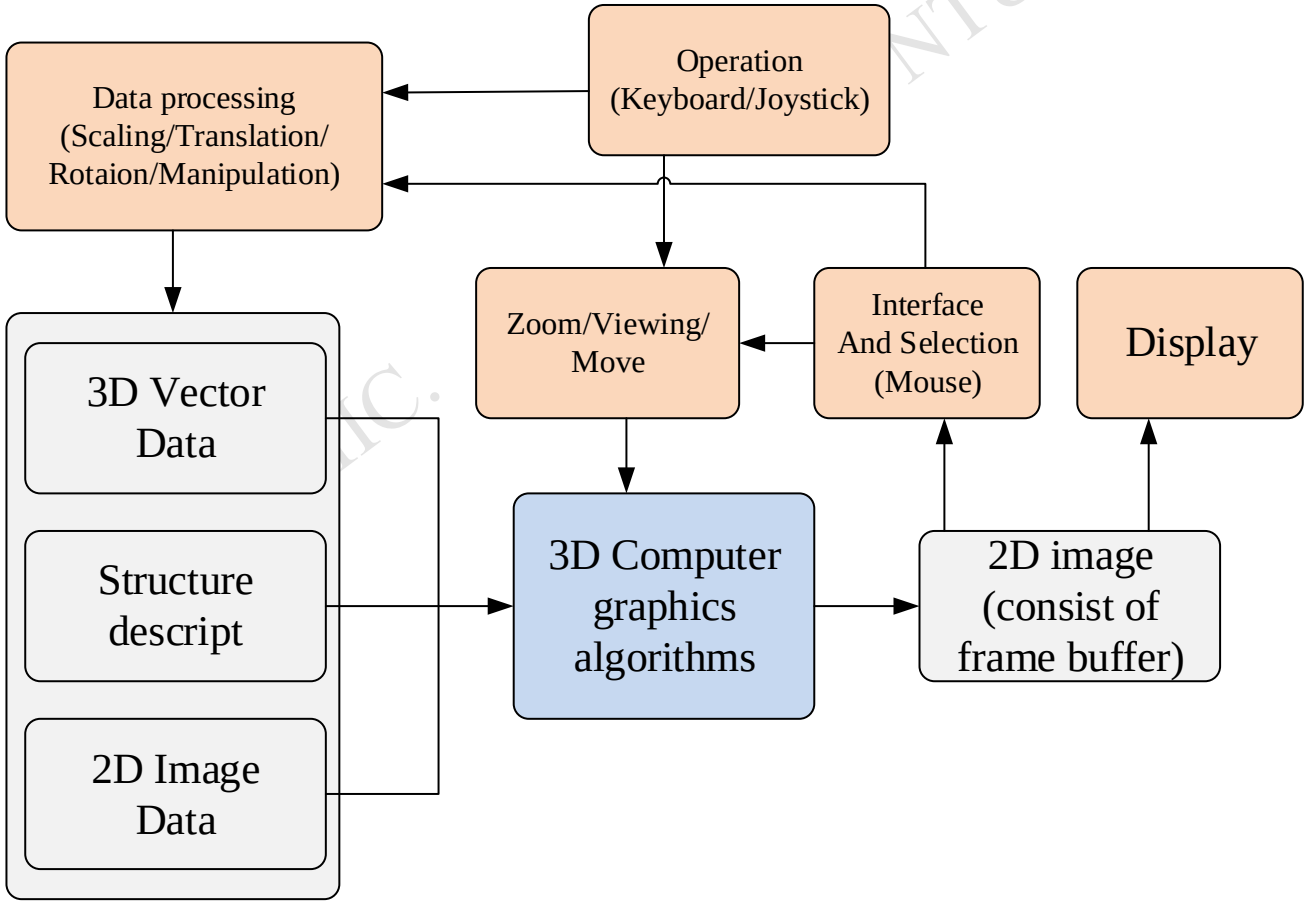
Rendering 3D scene (data type)

- In summary, the “computer graphics algorithm” converts vector data into “frame buffer”, then draw on the screen.



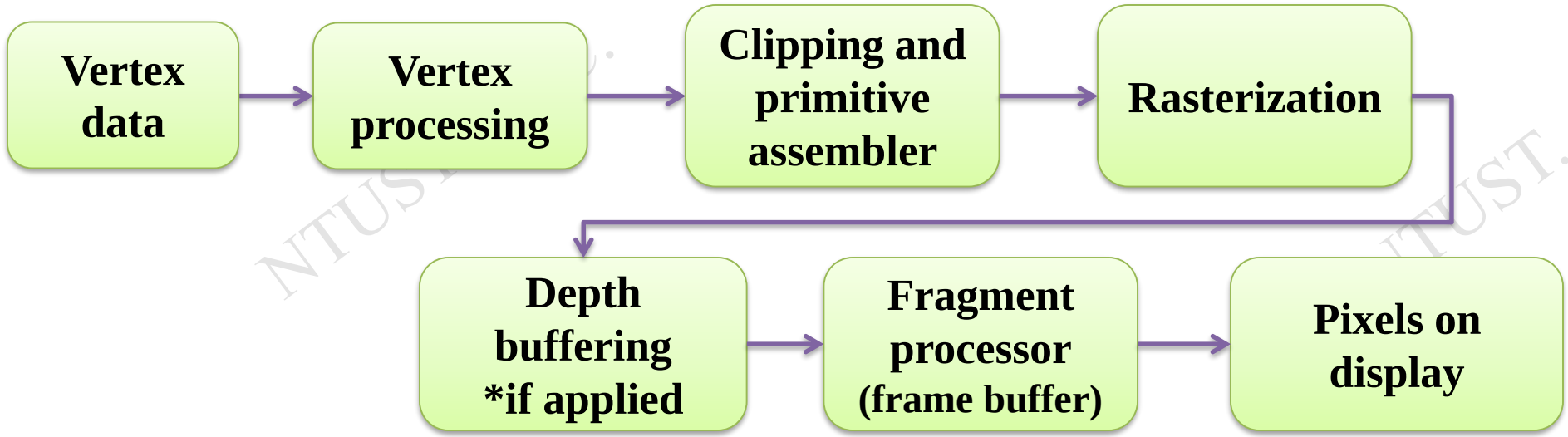
Interaction with rendering 3D scene (data type)

- Keyboard, mouse or event could trigger an action to update current scene.



A simple pipeline for rendering vertices

- Vertex data: the vectors we want to draw on screen
- Vertex processing: to transform (usually a 4x4 matrix including translation, rotation, scale, etc.) vertexes to new positions, and then, to transform to the camera coordinate.

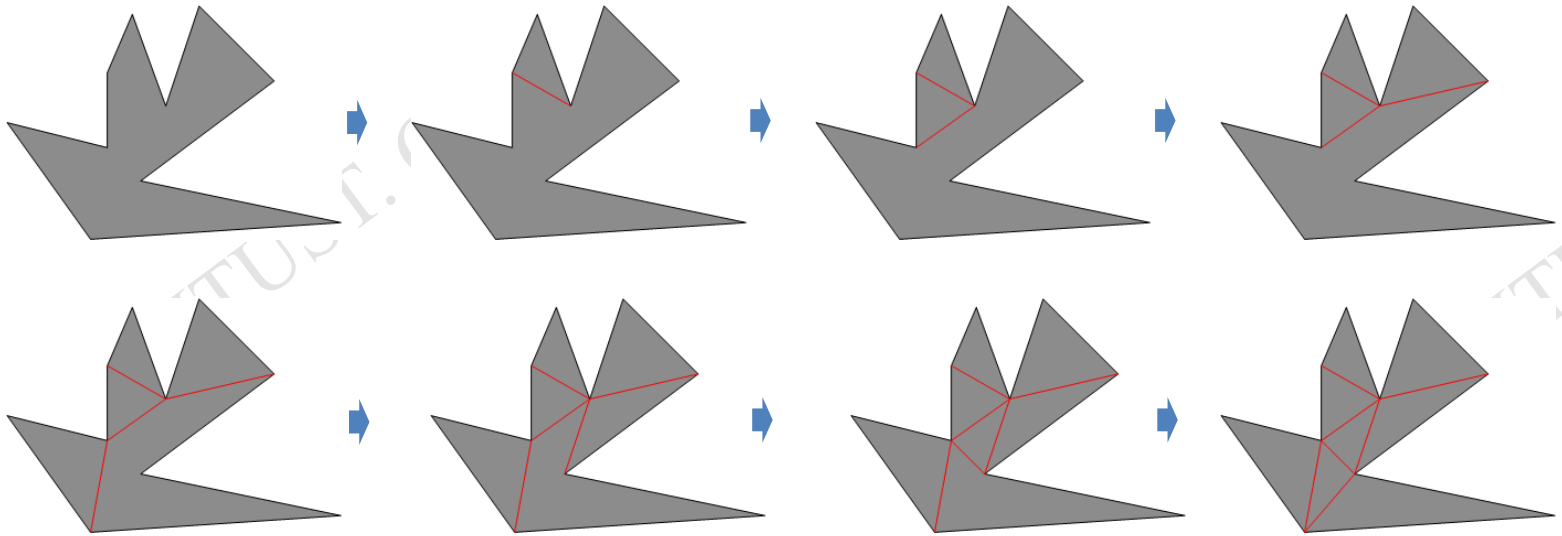


A simple pipeline for rendering vertices—cont.

- Clipping and primitive assembler: remove the part outside the “viewing volume”.
- Rasterization: to convert vectors into integrals (or pixels).
- Depth buffer: in “Rasterization”, the distance comparison to the camera is usually required for representing 3D priority
- Fragment processor: combine with other “image processing” methods for texture or other effects.

2D Tessellation (polygon triangulation)

- Two ears theorem (also called art gallery problem or museum problem)
  - Travel all vertexes of the polygon
  - If current vertex is concave, connect to next-next concave vertex, then split out this triangle.
  - Continue to split a triangle until the polygon becomes a triangle



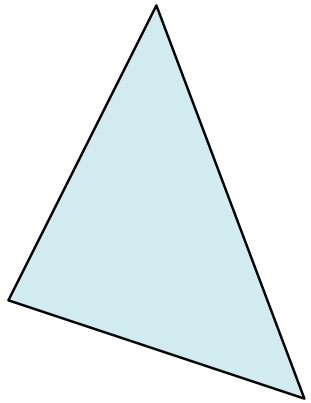
G. H. Meisters, Principal Vertices, Exposed Points, and Ears, *Amer. Math. Monthly* 87, 284-285, 1980

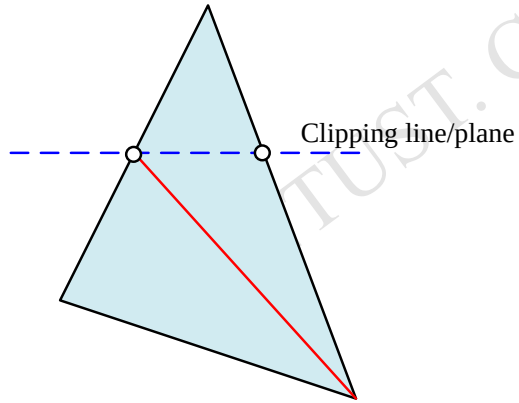
Why triangulation?

- A triangle is the most compact “convex” shape. Any “simple polygon” can be divided into triangles.
- Easy to implement in sub-division, merge.
- Can define unique plane.
- Convex shape, easy to fill color.

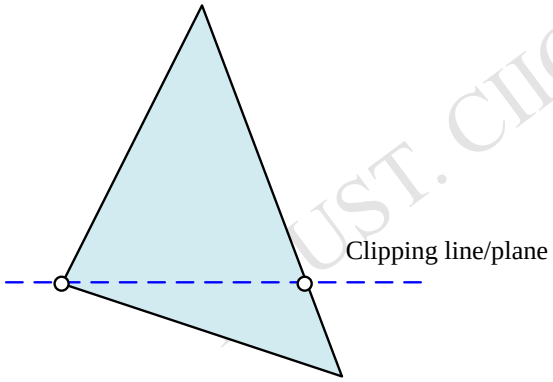
Clipping a triangle (as well as slicing in 3D printing)

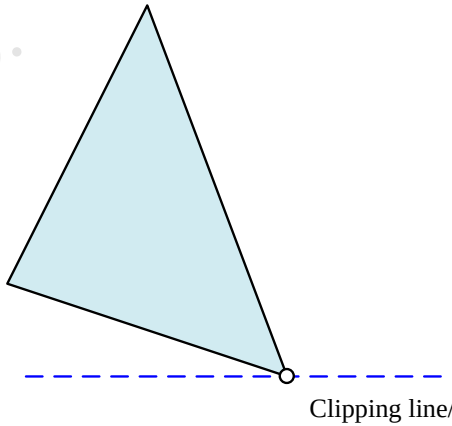
- General case

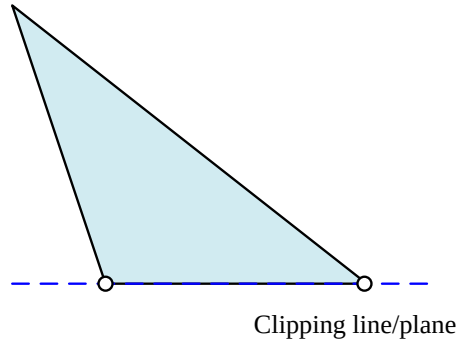




- Special case

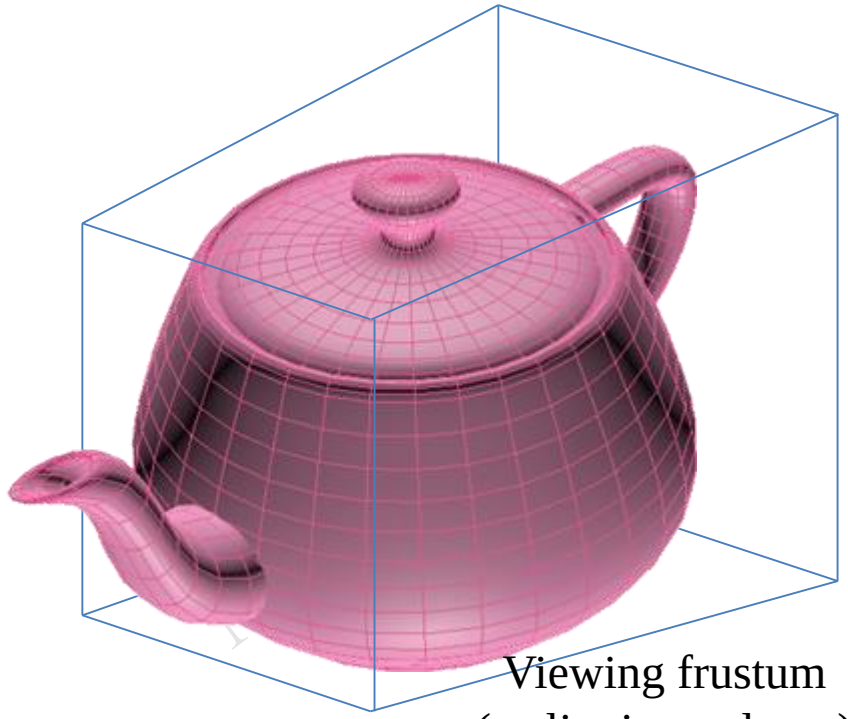






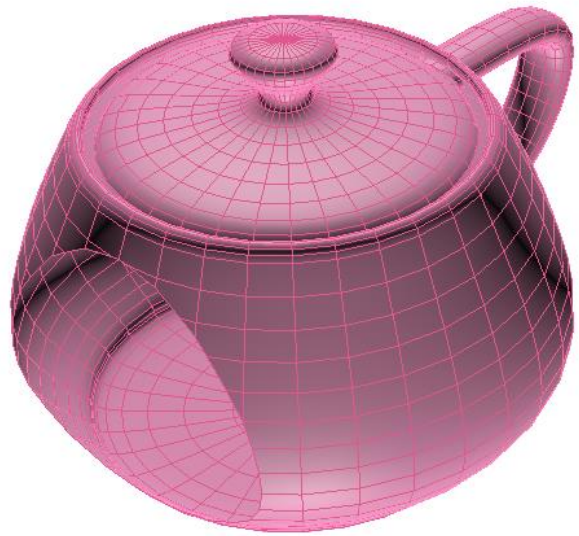
Clipping triangles of 3D model

- Example



Viewing frustum
(a clipping volume)

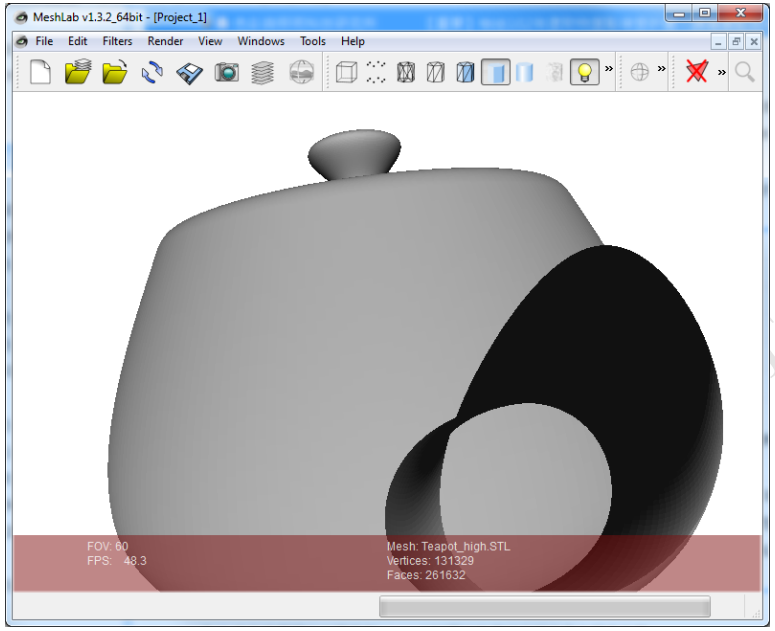
Before clipping

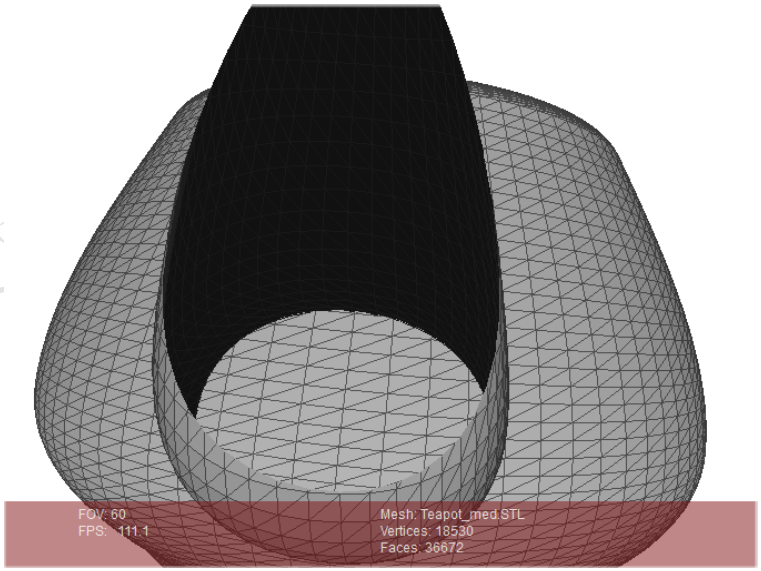


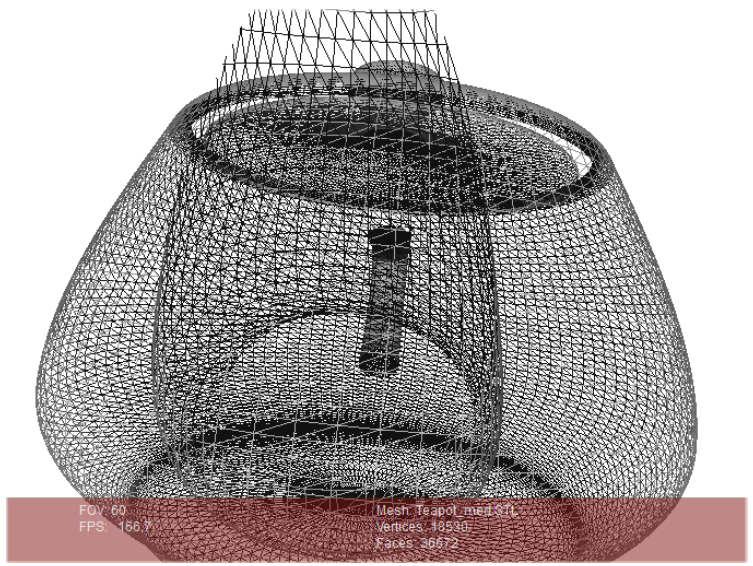
After clipping

Clipping triangles of 3D model

- Example (meshlab software)

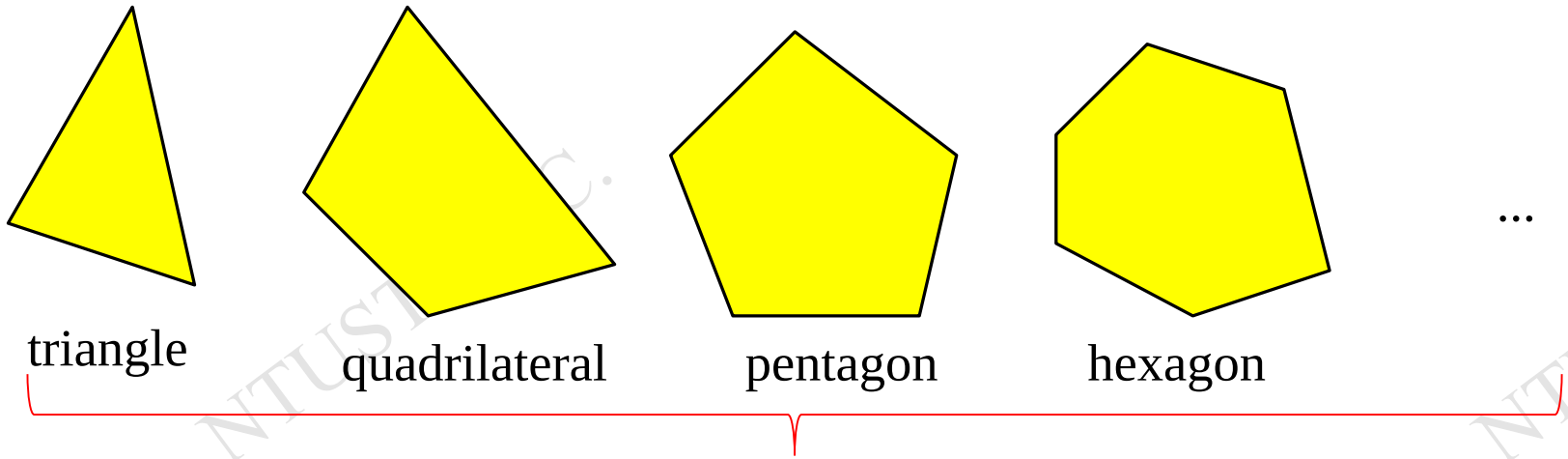






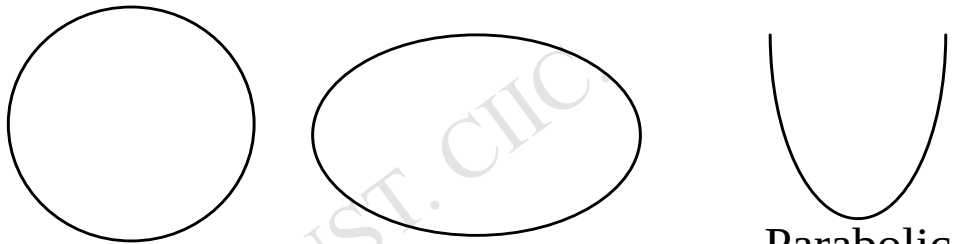
2D geometry: how a shape defined?

- Polygon: consist of finite vertexes which are stored in sequence.

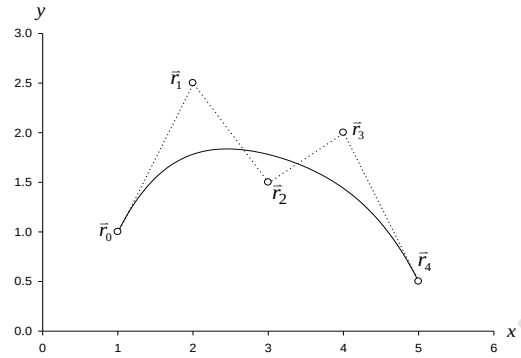


2D geometry: how a shape defined?

- The shape is dominated by equations or function.
  - Implicit function representation: $x^2 + y^2 = 1$
  - Explicit function representation: $y = x^2 + 2x - 1$



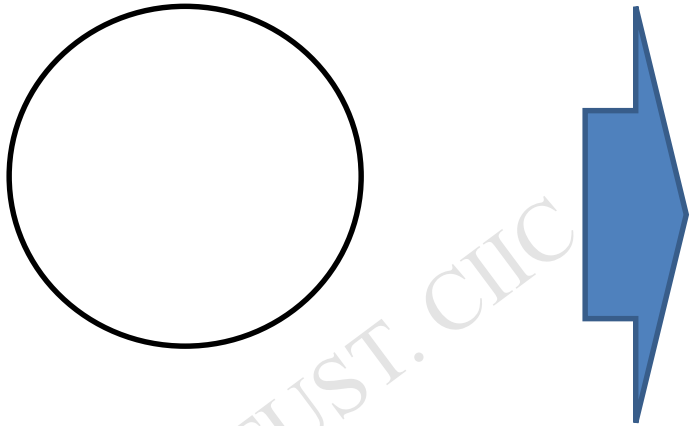
circle ellipse Parabolic curve



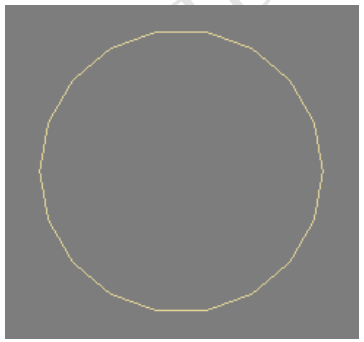
.... Bezier

Mathematical function(Implicit / Explicit)

2D geometry in computer graphics



1. Proper piecewise lines



For example: octadecagon

2. Governing equation/function (math.)
Explicit form:

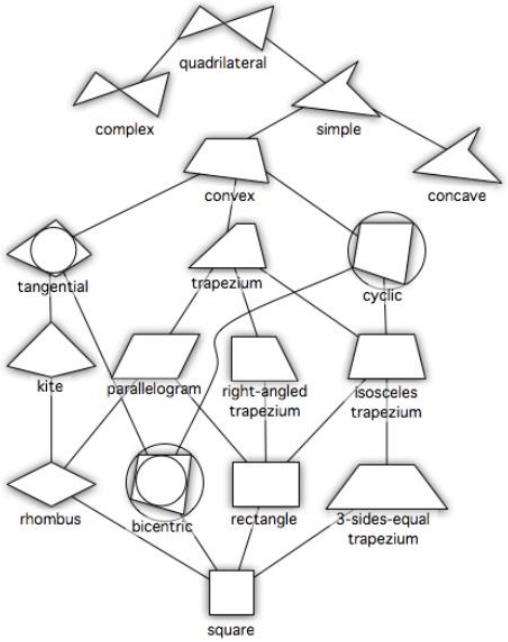
$$x = \cos\theta$$

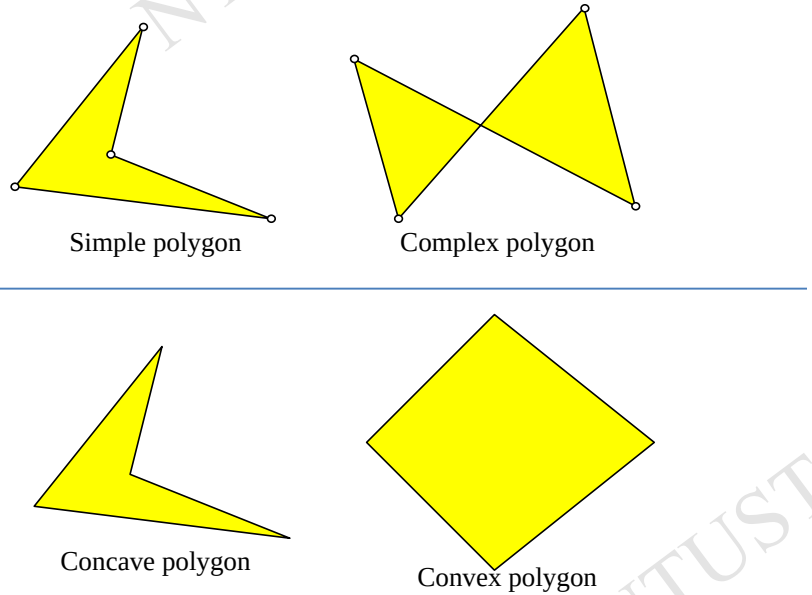
$$y = \sin\theta$$

$$\theta = -\pi \sim \pi$$

2D geometry: Polygon definition

- Simple polygon vs Complex polygon



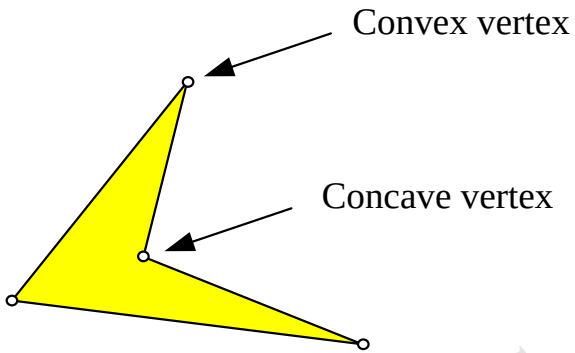


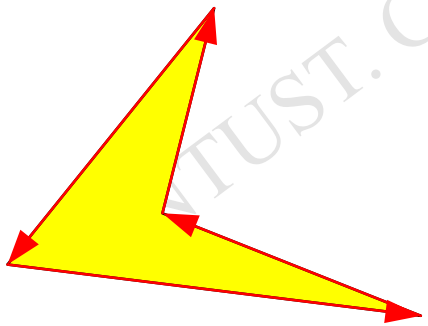
Triangulation for convex polygon is without problem.

http://en.wikipedia.org/wiki/Quadrilateral

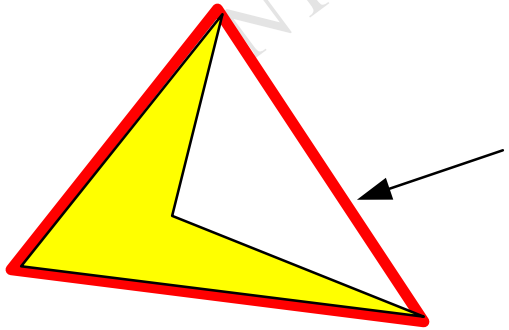
2D geometry

- Definition of convex/concave for 2D

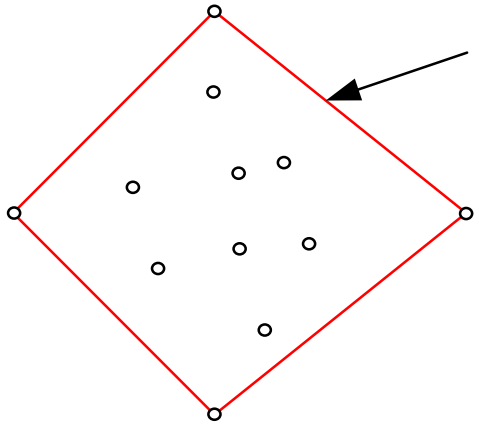




Right hand rule



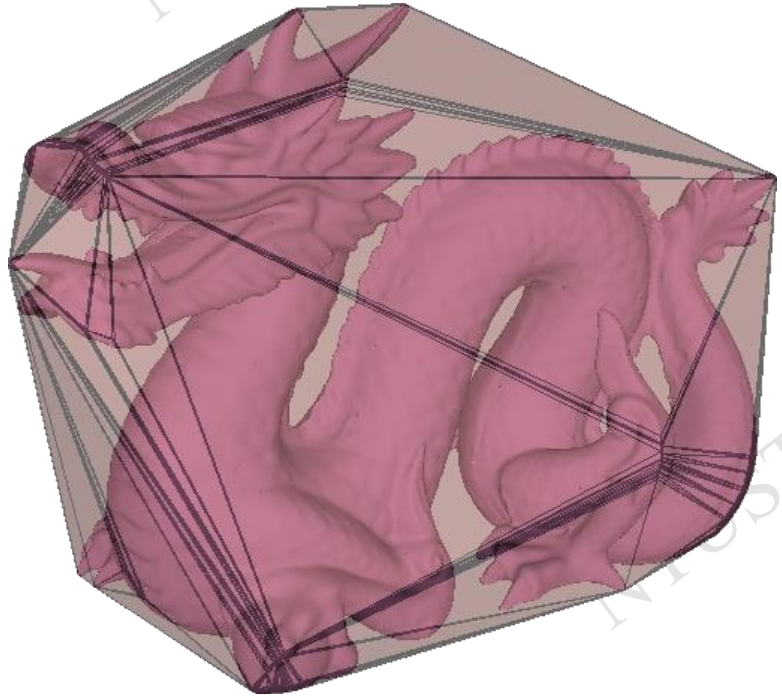
Convex hull

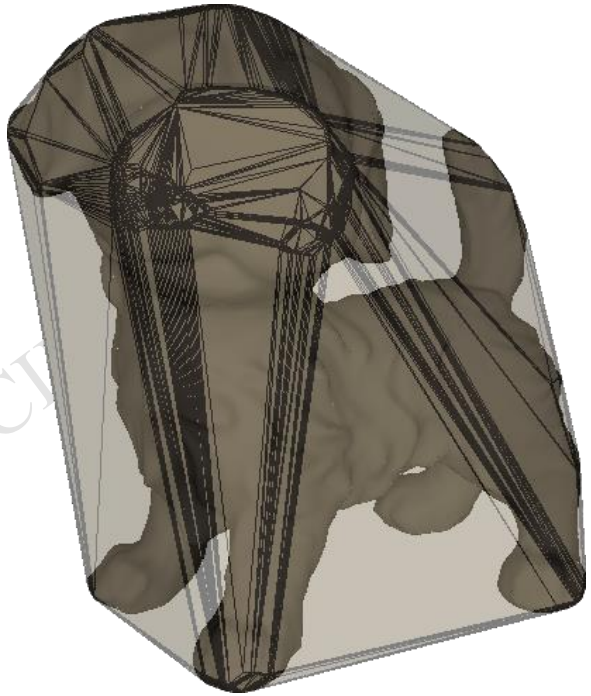


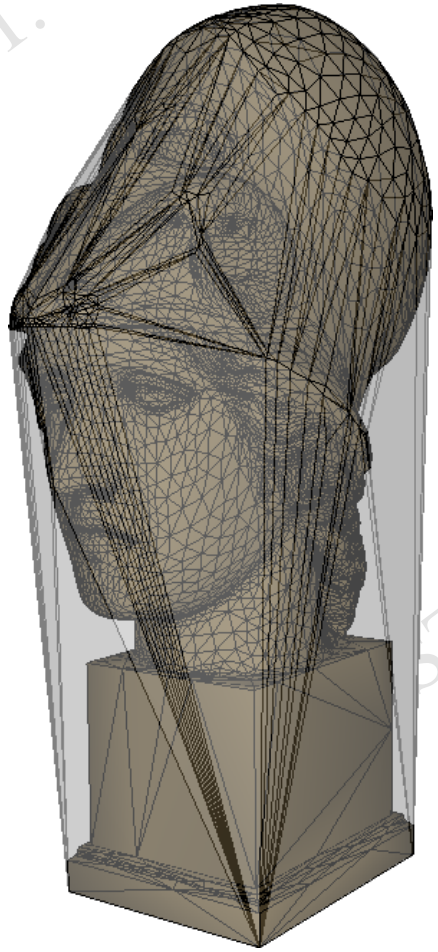
Convex hull

3D geometry, 3D convex hull example

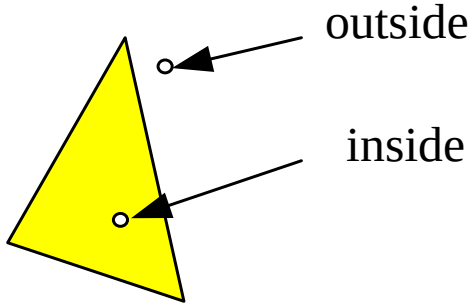
- The smallest enclosed volume.

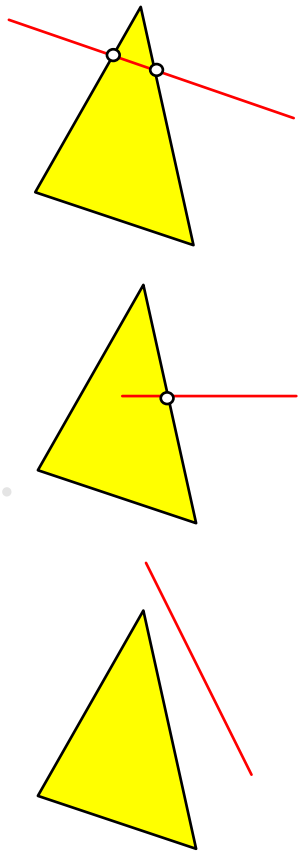


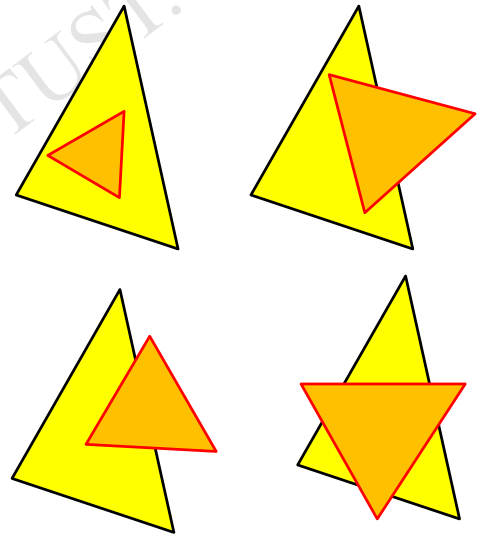




2D geometry intersection





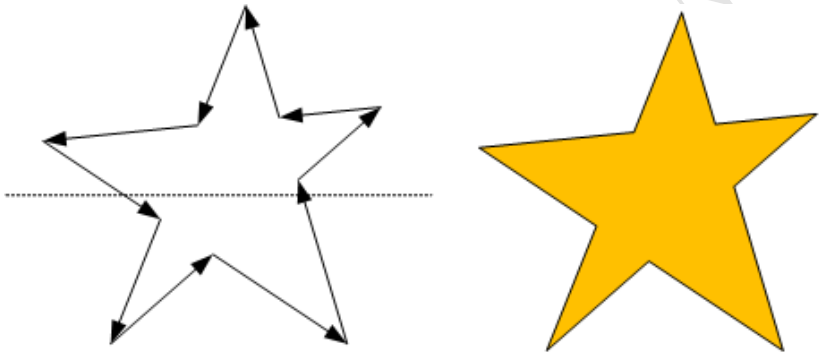


…

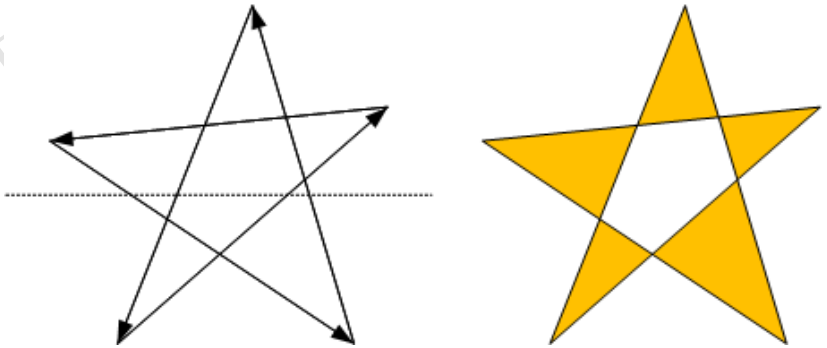
Divided into small triangles, then process.

Scan line rendering

- Switch the filling index by winding number



Simple polygon



Complex polygon

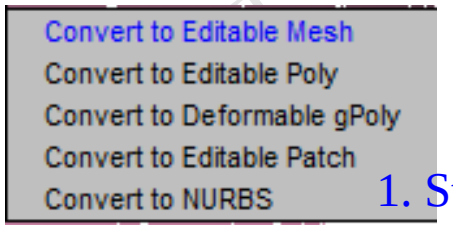
Scan line rendering

- Traditional TEXT v.s. True Type

Basic 3D geometry data structure

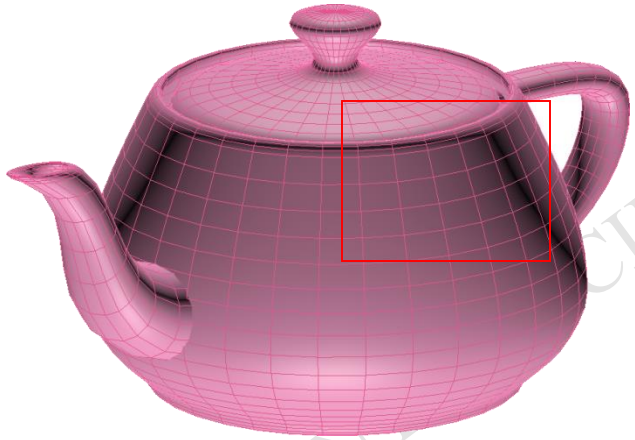
- For Representation?
- For Storage ?
- For Computation ?

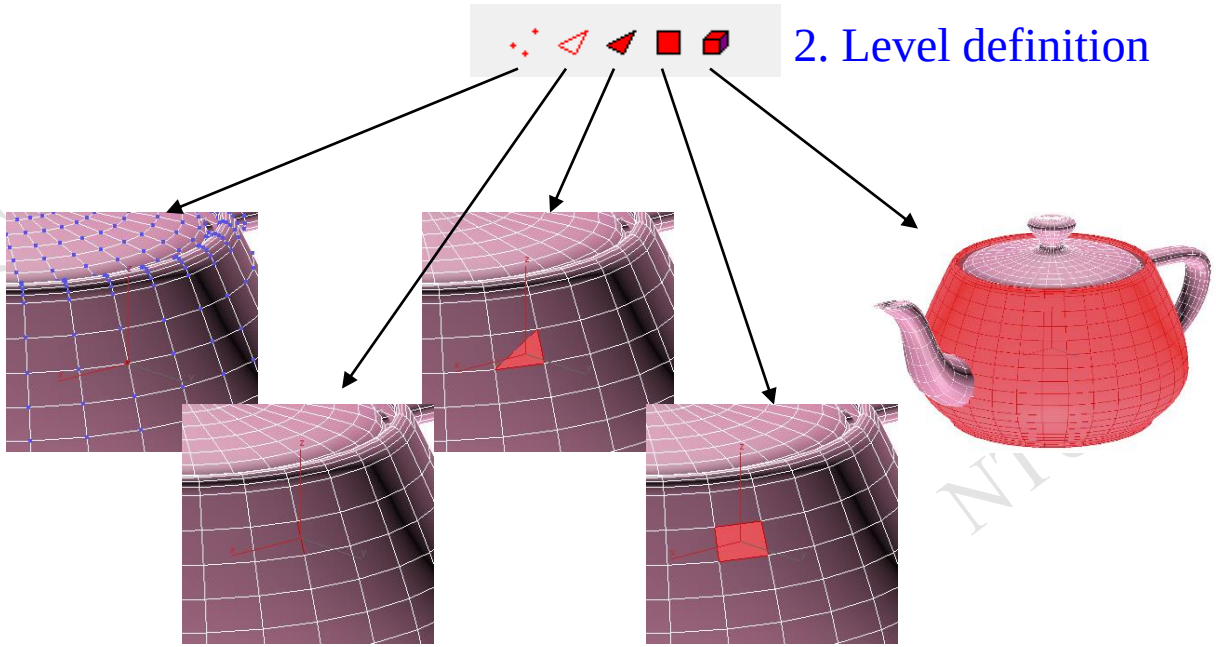
Definition in 3DMax



1. Structure types

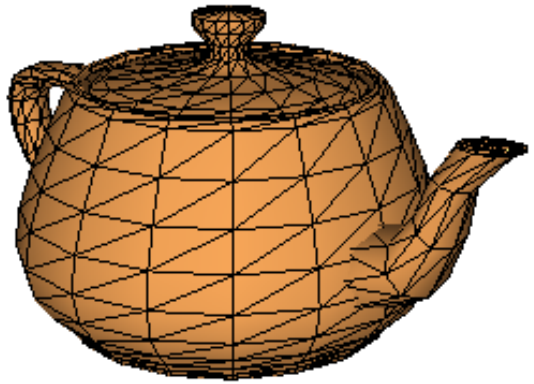
2. Level definition



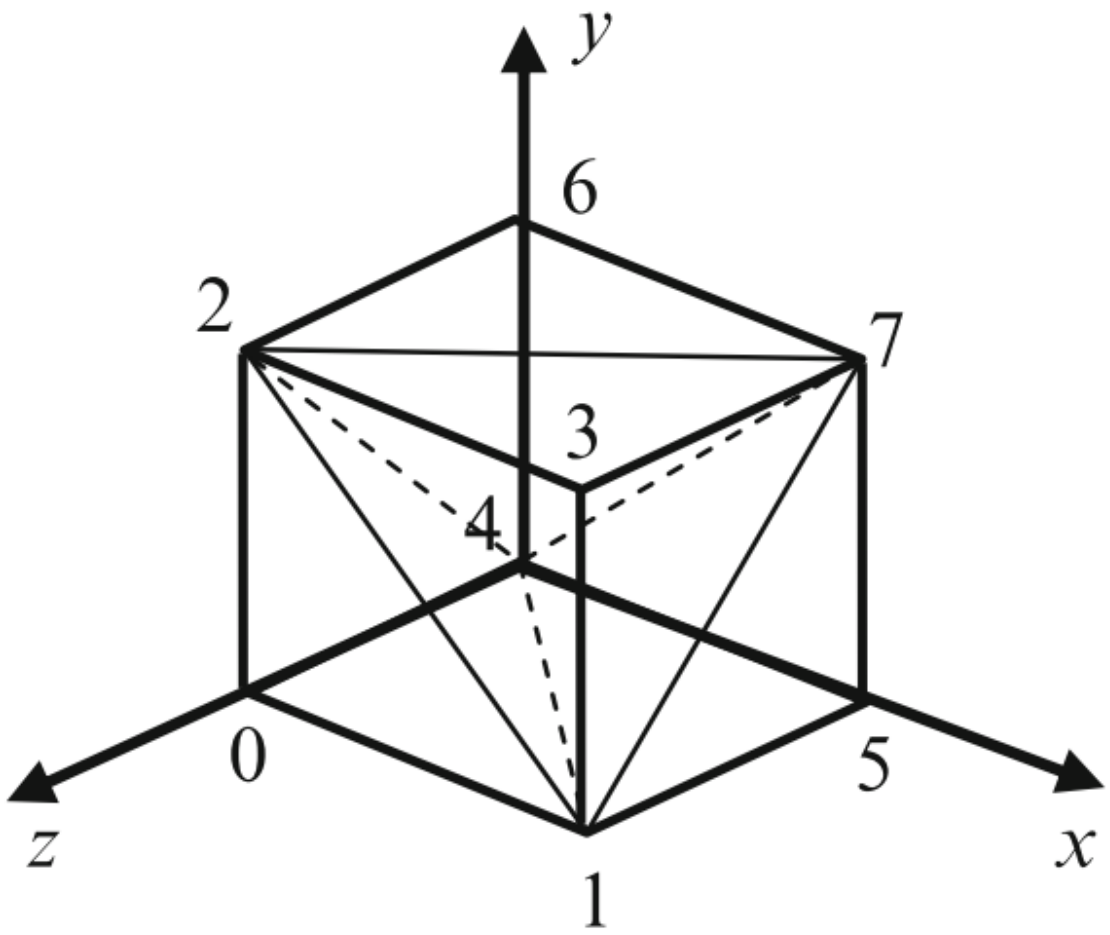


Basic 3D geometry data structure

- The most fundamental definition (for triangle mesh):
- **Vectors of vertexes**—Required
- **Vertex indexes of the triangles**—Required
- Edge / half-edge →for speed up searching—Optional
- Normals of triangles →for render—Optional
- Normals of vertexes→for render—Optional



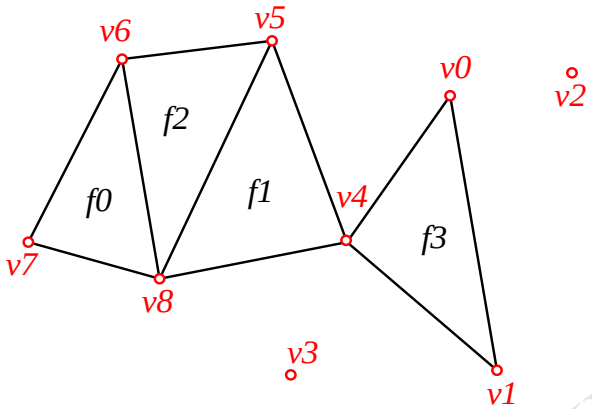
Mesh representation: for most 3D file format



Vertex List		
x	y	z
0.0	0.0	1.0
1.0	0.0	1.0
0.0	1.0	1.0
1.0	1.0	1.0
0.0	0.0	0.0
1.0	0.0	0.0
0.0	1.0	0.0
1.0	1.0	0.0

Triangle List		
i	j	k
0	1	2
1	3	2
2	3	7
2	7	6
1	7	3
1	5	7
6	7	4
7	5	4
0	4	1
1	4	5
2	6	4
0	2	4

3D geometry data structure (example)



Shape appearance

Integer

Float point

f0	[7,8,6]
f1	[8,4,5]
f2	[6,8,5]
f3	[0,4,1]

Face List

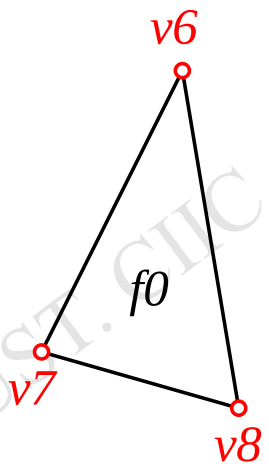
Data format

v0	[x0,y0,z0,w0]
v1	[x1,y1,z1,w1]
v2	[x2,y2,z2,w2]
v3	[x3,y3,z3,w3]
v4	[x4,y4,z4,w4]
v5	[x5,y5,z5,w5]
v6	[x6,y6,z6,w6]
v7	[x7,y7,z7,w7]
v8	[x8,y8,z8,w8]

Vertex List

3D geometry data structure (example) cont.

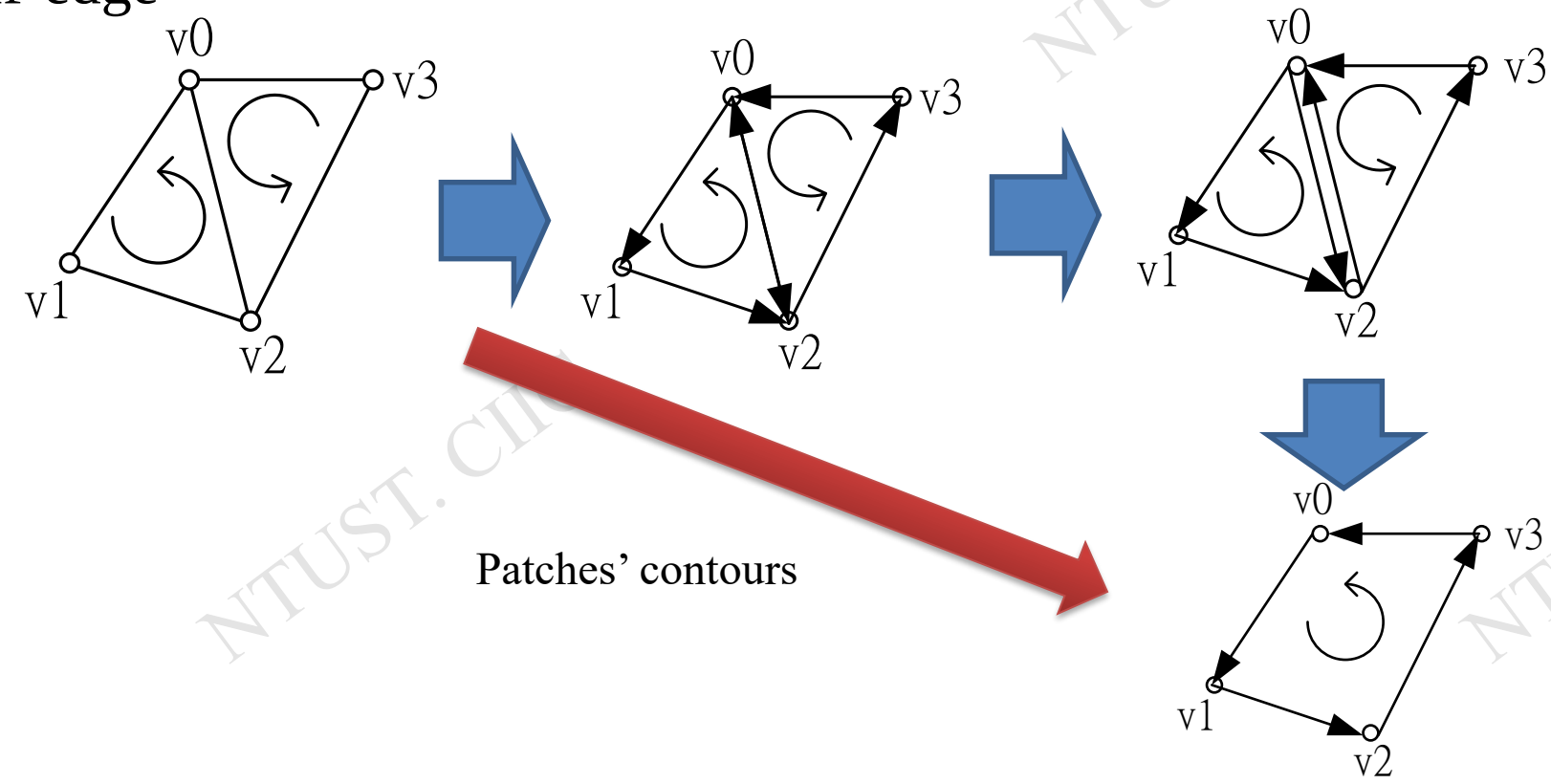
- Index alternation
  - This means the vertex index of triangle can be [7, 8, 6], [8, 6, 7] or [6, 7, 8]. They indicate the exact same triangle.



Ex:
[7,8,6]
[8,6,7]
[6,7,8]

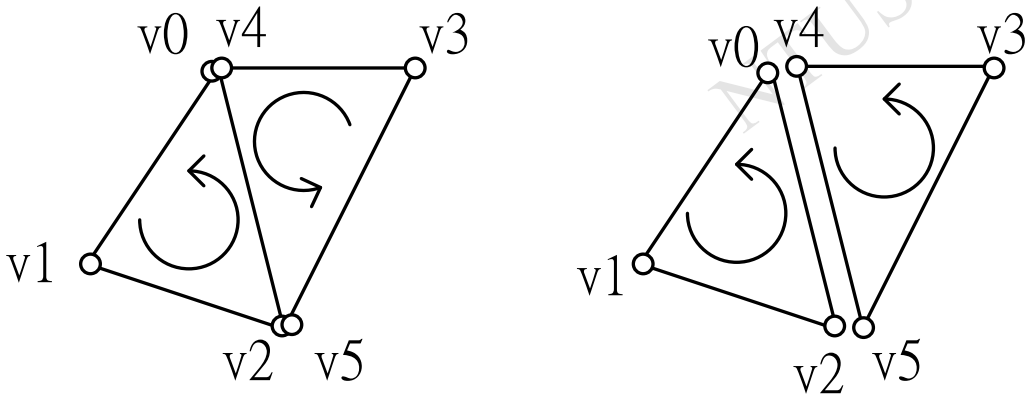
3D geometry data structure

- The connection between neighboring triangles
  - Half-edge



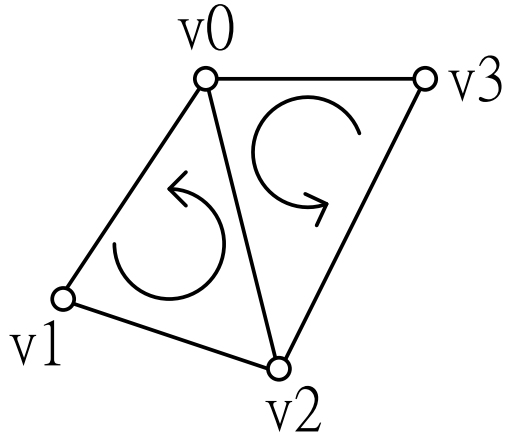
3D geometry data structure

- Isolated triangles vs Connecting triangles

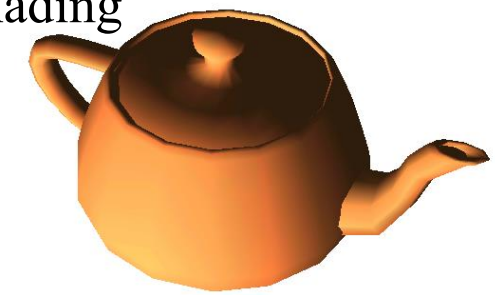


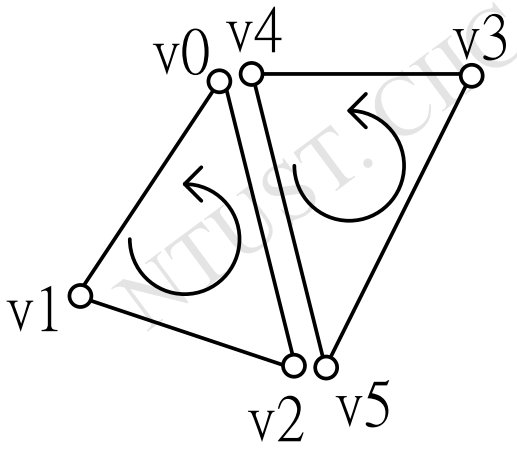
- Vertex V0 and V4 are defined twice for the exact same position. (V2 and V5, as well)

3D geometry data structure—comparison



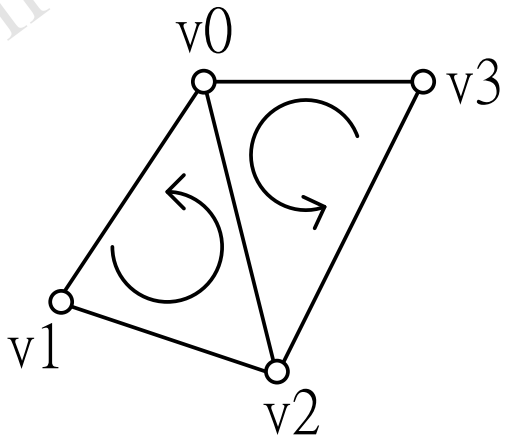
Continuing structure → Considered as single patch. Be able to render as "smooth shading"



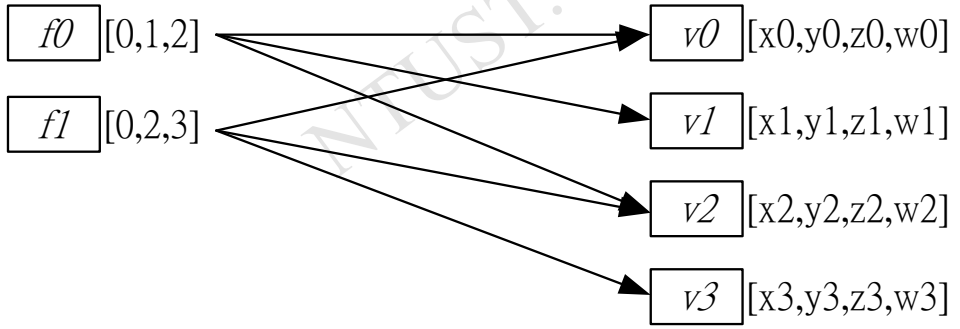


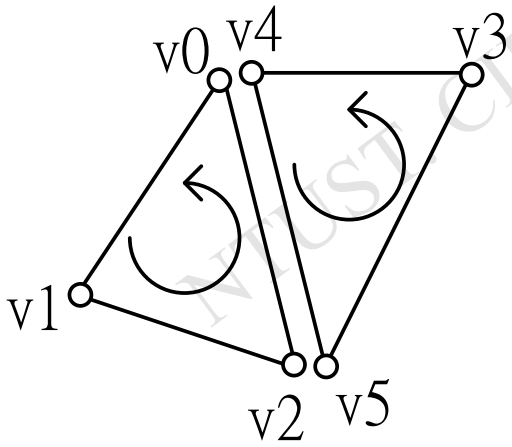
Discontinue structure → be isolated triangles (plane) → Flat shading only

3D geometry data structure—comparison

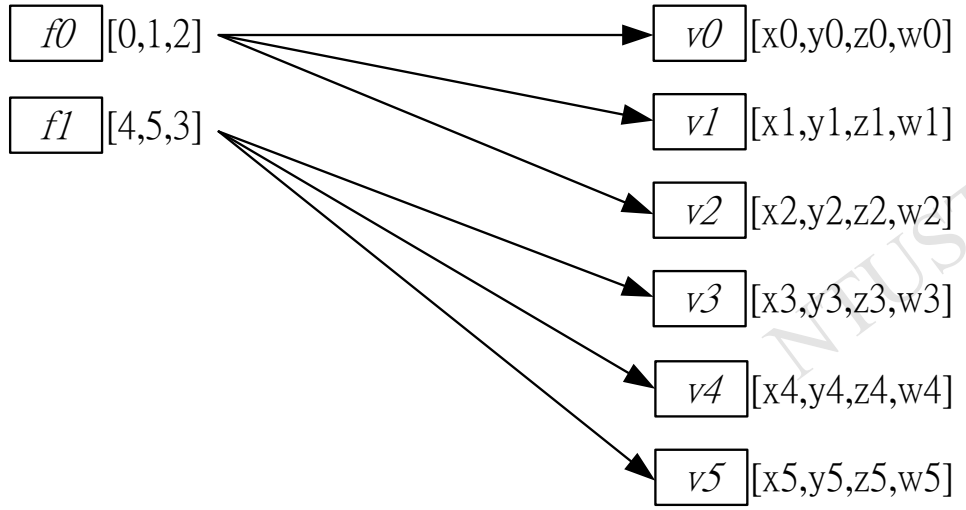


[PLY/OBJ]

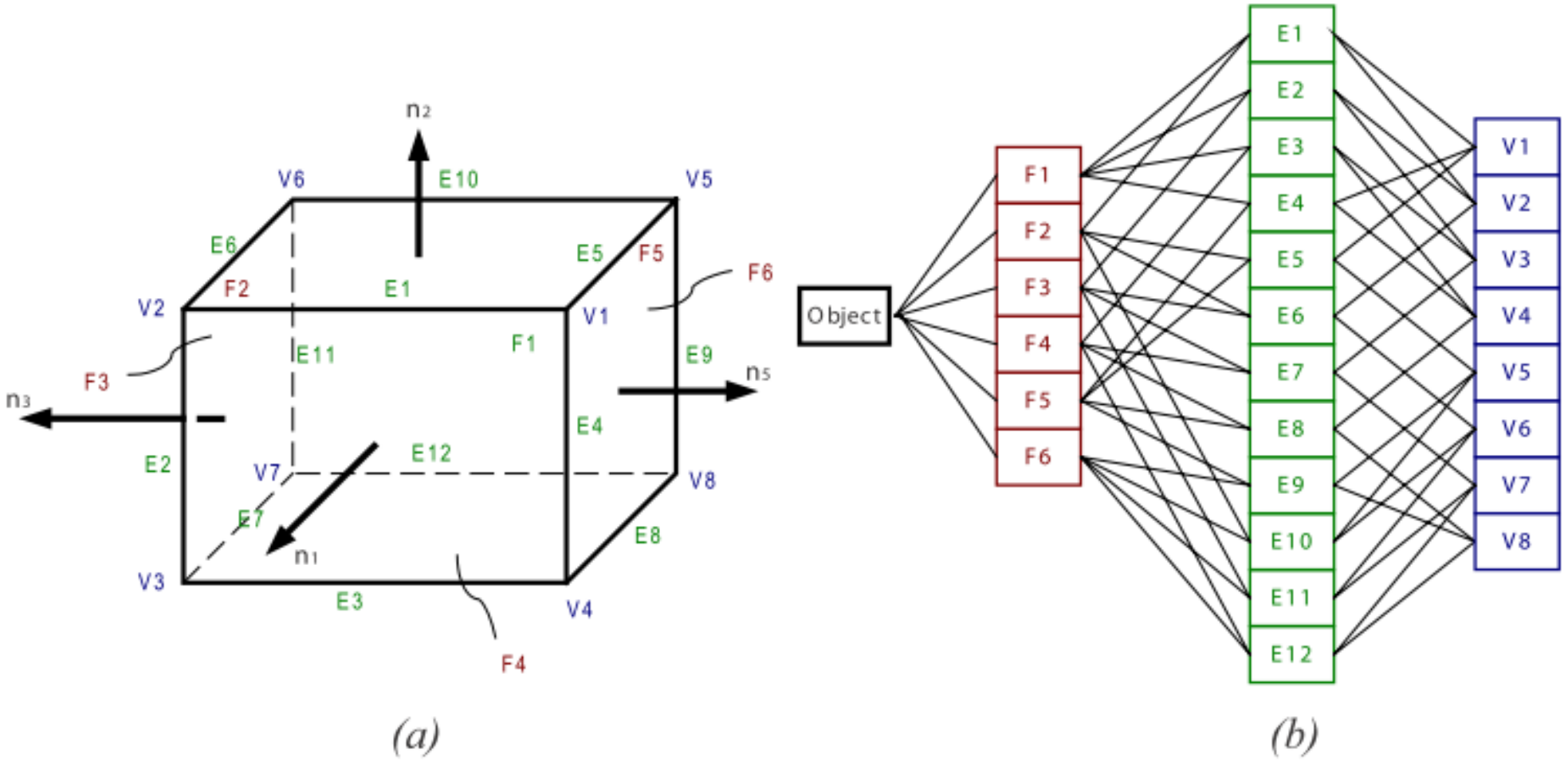




[STL]

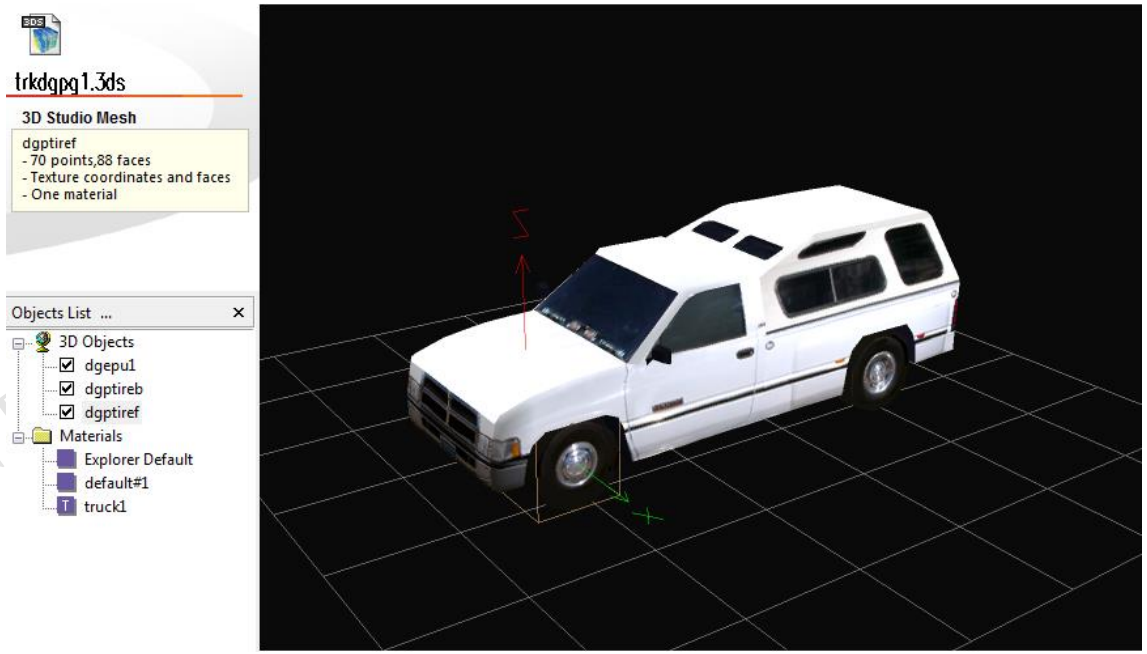


3D geometry data structure: B-Rep type



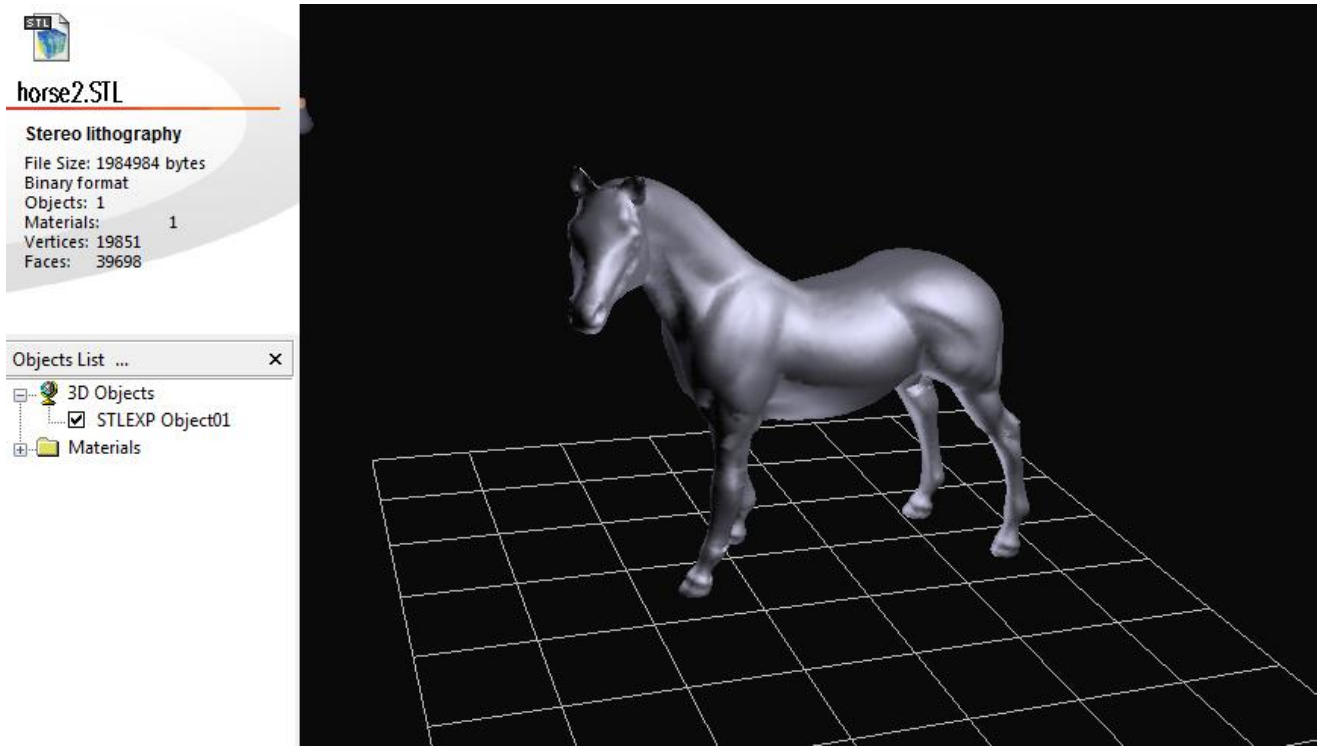
3D geometry data structure

- Hierarchy structure
  - Structure includes “group”, ex *.3ds.
  - Some file may have animation key-frame, *.wrl



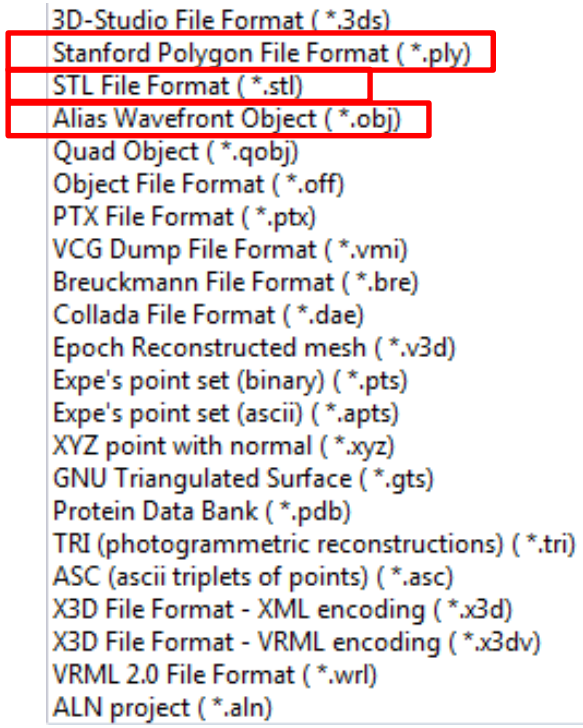
3D geometry data structure

- Single object:
  - For example: stl



3D geometry data structure

- Common format
  - I recommend you that understanding a popular 3D file format as possible(obj is good)
  - And use freeware to convert among the rest formats
- Useful conversion tool: meshlab

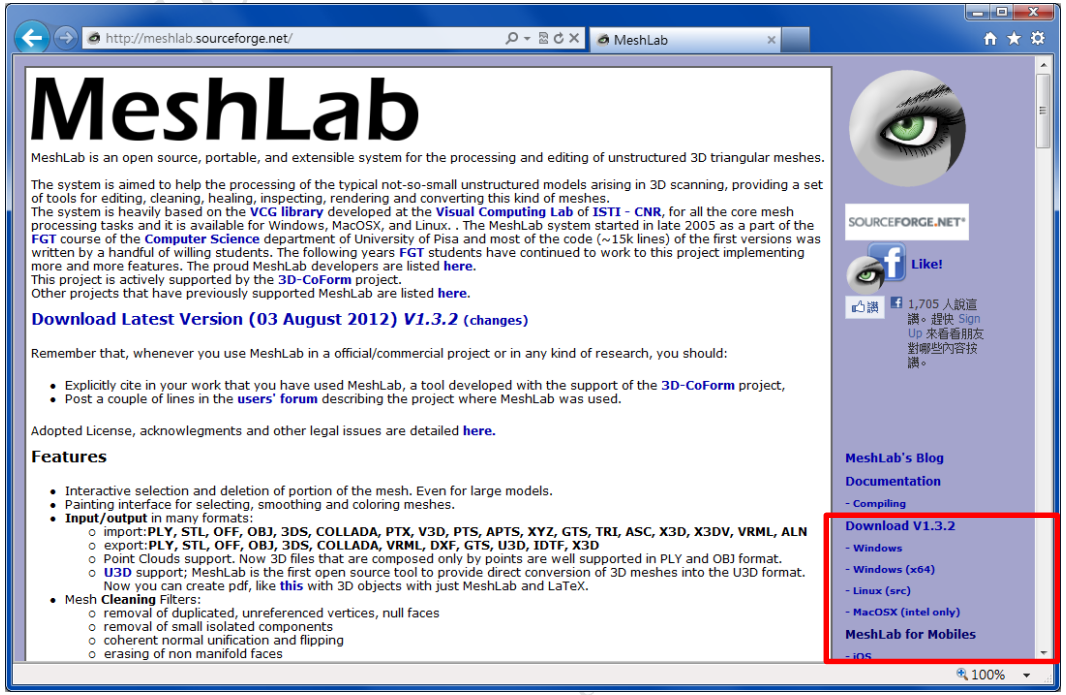


3D File format : Standards

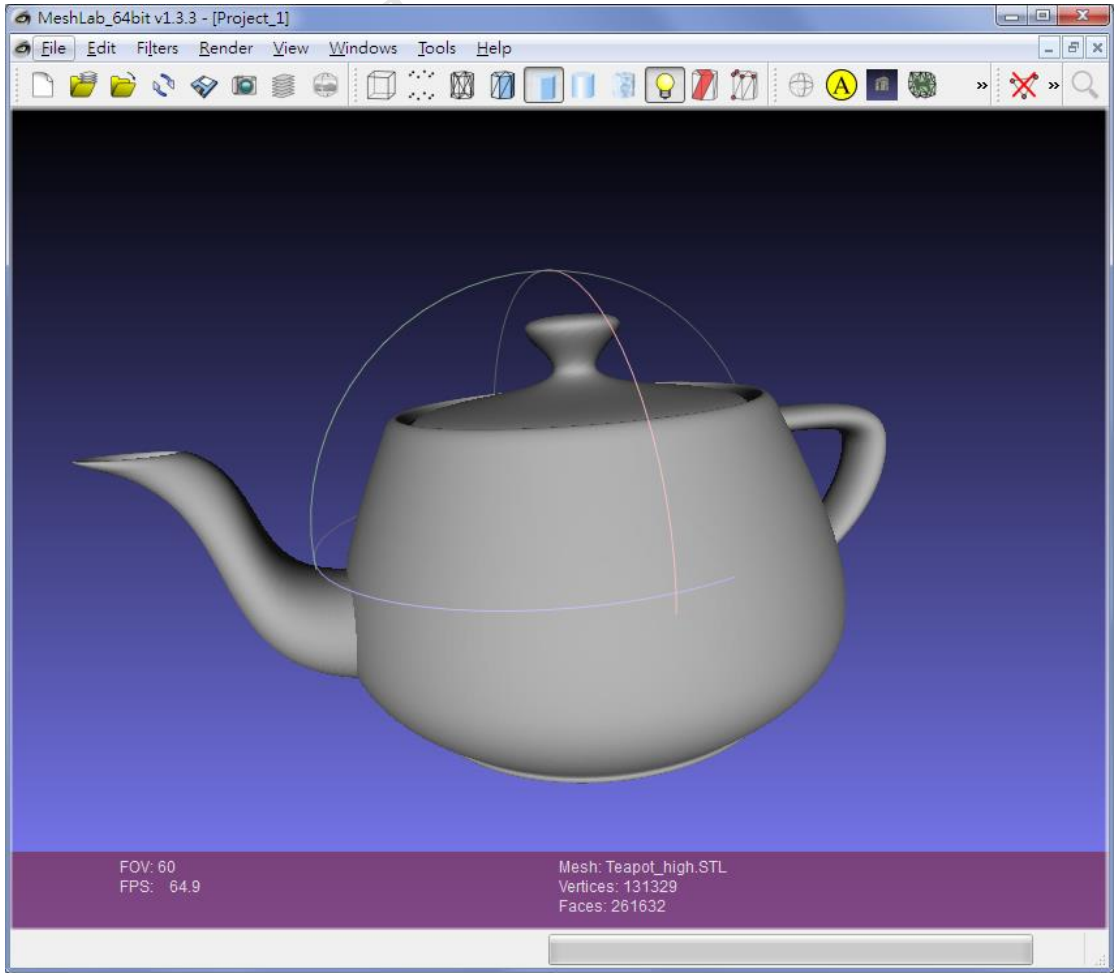
- Triangle / Point / Quad (vertex + boundary)
  - 3MF
  - STL
  - OBJ
  - PLY
  - XYZ
- Animation available
  - 3DS
  - VRML

Meshlab (opensource - free)

Program screen snapshot



Download page

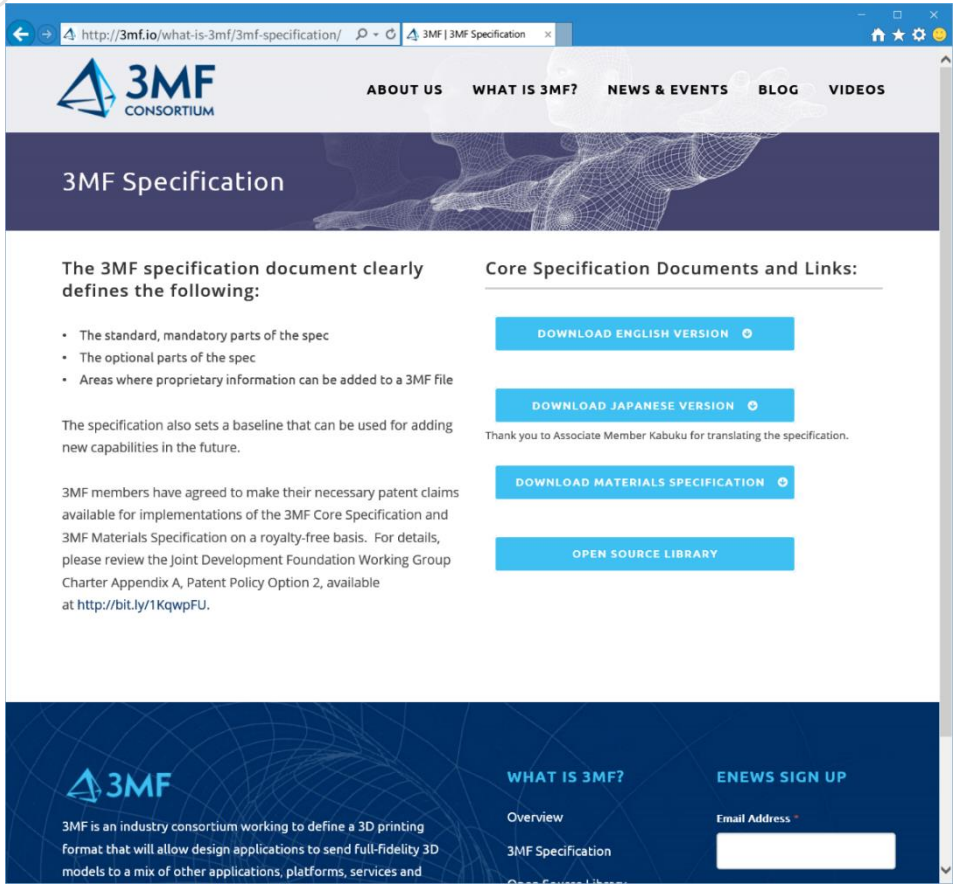


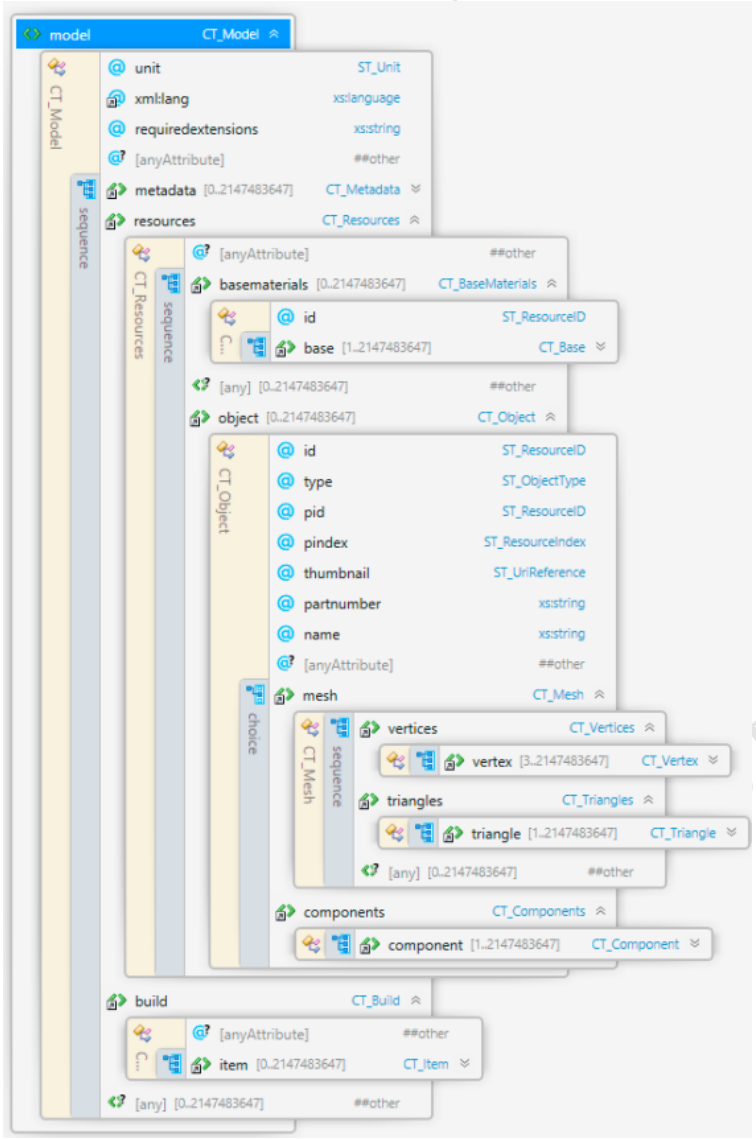
3D file format

File format	Color resolution	Surface color representation	Note
PLY	32 bit on vertex	Interpolate in triangle	**PLY**(Stanford triangle format
OBJ	32 bit on vertex 32 bit texture map	Interpolate in triangle Interpolate in image space	**OBJ**(wavefront object)
STL	16 bit on facet	Uniform color in a triangle	**STL**(stereo lithography)

3MF (3D Manufacturing Format)

- Design for 3D printing





http://3mf.io/what-is-3mf/

3DS (Autodesk 3D Studio) file format

- Binary format
- Hierarchical structure
  - Object (Vertex, Face Description, Material)
    - Local coordinate
  - Light
  - Camera
  - Keyframe

```
0x4D4D // Main Chunk
├─ 0x0002 // M3D Version
├─ 0x3D3D // 3D Editor Chunk
│  ├─ 0x4000 // Object Block
│  │  ├─ 0x4100 // Triangular Mesh
│  │  │  ├─ 0x4110 // Vertices List
│  │  │  ├─ 0x4120 // Faces Description
│  │  │  │  ├─ 0x4130 // Faces Material
│  │  │  │  └─ 0x4150 // Smoothing Group List
│  │  │  ├─ 0x4140 // Mapping Coordinates List
│  │  │  └─ 0x4160 // Local Coordinates System
│  │  ├─ 0x4600 // Light
│  │  │  └─ 0x4610 // Spotlight
│  │  └─ 0x4700 // Camera
│  └─ 0xAFFF // Material Block
│     ├─ 0xA000 // Material Name
│     ├─ 0xA010 // Ambient Color
│     ├─ 0xA020 // Diffuse Color
│     ├─ 0xA030 // Specular Color
│     ├─ 0xA200 // Texture Map 1
│     ├─ 0xA230 // Bump Map
│     └─ 0xA220 // Reflection Map
│        │  /* Sub Chunks For Each Map */
│        ├─ 0xA300 // Mapping Filename
│        └─ 0xA351 // Mapping Parameters
└─ 0xB000 // Keyframer Chunk
   ├─ 0xB002 // Mesh Information Block
   ├─ 0xB007 // Spot Light Information Block
   └─ 0xB008 // Frames (Start and End)
      ├─ 0xB010 // Object Name
      ├─ 0xB013 // Object Pivot Point
      ├─ 0xB020 // Position Track
      ├─ 0xB021 // Rotation Track
      ├─ 0xB022 // Scale Track
      └─ 0xB030 // Hierarchy Position
```

https://en.wikipedia.org/wiki/.3ds

STL (Stereo lithography) file format

- STL: binary / ascii → popular in medical image and computer aided design
  - UINT8[80] – Header
  - UINT32 – Number of triangles
    for each triangle
  - REAL32[3] – Normal vector
  - REAL32[3] – Vertex 1
  - REAL32[3] – Vertex 2
  - REAL32[3] – Vertex 3
  - UINT16 – Attribute byte count end (or color information)

http://en.wikipedia.org/wiki/STL_(file_format)

STL (Stereo lithography) file format—cont.

ASCII

```
solid vcg
  facet normal 0.000000e+000 0.000000e+000 -1.000000e+000
    outer loop
      vertex  -2.395210e+001 -2.371260e+001 0.000000e+000
      vertex  -2.395210e+001 2.227540e+001 0.000000e+000
      vertex  2.395210e+001 2.227540e+001 0.000000e+000
    endloop
  endfacet
  facet normal 0.000000e+000 -0.000000e+000 -1.000000e+000
    outer loop
      vertex  2.395210e+001 2.227540e+001 0.000000e+000
      vertex  2.395210e+001 -2.371260e+001 0.000000e+000
      vertex  -2.395210e+001 -2.371260e+001 0.000000e+000
    endloop
  endfacet
  facet normal 0.000000e+000 0.000000e+000 1.000000e+000
    outer loop
      vertex  -2.395210e+001 -2.371260e+001 3.640720e+001
      vertex  2.395210e+001 -2.371260e+001 3.640720e+001
      vertex  2.395210e+001 2.227540e+001 3.640720e+001
    endloop
  endfacet
  facet normal 0.000000e+000 0.000000e+000 1.000000e+000
    outer loop
      vertex  2.395210e+001 2.227540e+001 3.640720e+001
      vertex  -2.395210e+001 2.227540e+001 3.640720e+001
      vertex  -2.395210e+001 -2.371260e+001 3.640720e+001
    endloop
  endfacet
  facet normal 0.000000e+000 -1.000000e+000 0.000000e+000
```

Binary

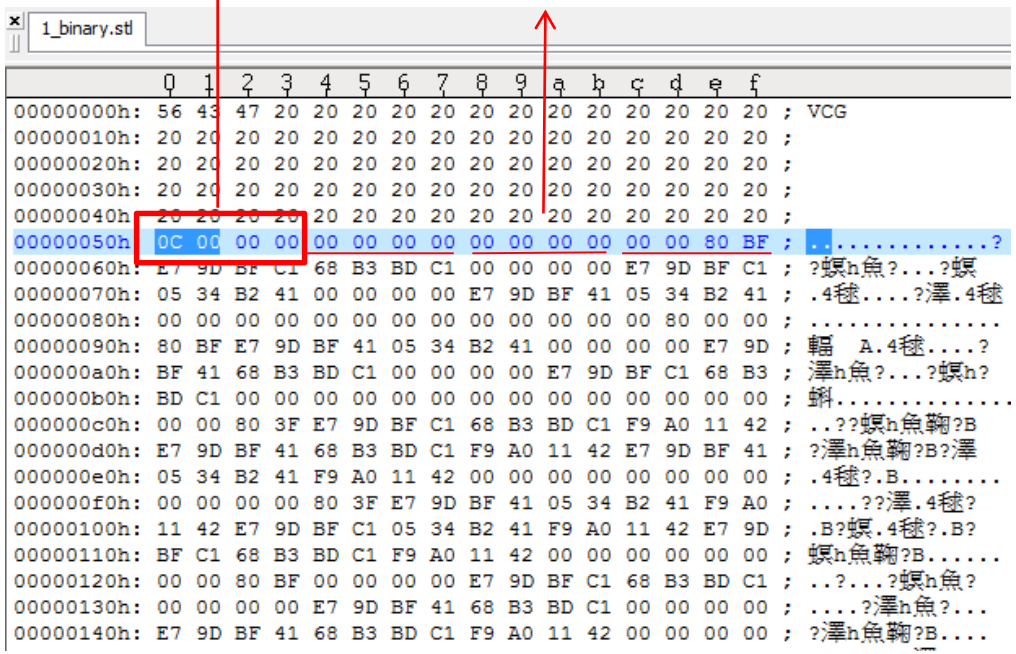
1_binary.stl

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 56 43 47 20 20 20 20 20 20 20 20 20 20 20 20 20 ; VCG
00000010h: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 ;
00000020h: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 ;
00000030h: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 ;
00000040h: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 ;
00000050h: 0C 00 00 00 00 00 00 00 00 00 00 00 00 00 80 BF ; ..............?
00000060h: E7 9D BF C1 68 B3 BD C1 00 00 00 00 E7 9D BF C1 ; ?螟h魚?...?螟
00000070h: 05 34 B2 41 00 00 00 00 E7 9D BF 41 05 34 B2 41 ; .4毬....?澤.4毬
00000080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 ; ................
00000090h: 80 BF E7 9D BF 41 05 34 B2 41 00 00 00 00 E7 9D ; 輻  A.4毬....?
000000a0h: BF 41 68 B3 BD C1 00 00 00 00 E7 9D BF C1 68 B3 ; 澤h魚?...?螟h?
000000b0h: BD C1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; 蝌..............
000000c0h: 00 00 80 3F E7 9D BF C1 68 B3 BD C1 F9 A0 11 42 ; ..??螟h魚鞠?B
000000d0h: E7 9D BF 41 68 B3 BD C1 F9 A0 11 42 E7 9D BF 41 ; ?澤h魚鞠?B?澤
000000e0h: 05 34 B2 41 F9 A0 11 42 00 00 00 00 00 00 00 00 ; .4毬?.B........
000000f0h: 00 00 00 00 80 3F E7 9D BF 41 05 34 B2 41 F9 A0 ; ....??澤.4毬?
00000100h: 11 42 E7 9D BF C1 05 34 B2 41 F9 A0 11 42 E7 9D ; .B?螟.4毬?.B?
00000110h: BF C1 68 B3 BD C1 F9 A0 11 42 00 00 00 00 00 00 ; 螟h魚鞠?B......
00000120h: 00 00 80 BF 00 00 00 00 E7 9D BF C1 68 B3 BD C1 ; ..?...?螟h魚?
00000130h: 00 00 00 00 E7 9D BF 41 68 B3 BD C1 00 00 00 00 ; ....?澤h魚?...
00000140h: E7 9D BF 41 68 B3 BD C1 F9 A0 11 42 00 00 00 00 ; ?澤h魚鞠?B....
```

STL (Stereo lithography) file format—cont.

Triangle num. : 12

(0.0, 0.0, -1.0)



http://en.wikipedia.org/wiki/STL_(file_format)

UINT8[80] – Header
UINT32 – Number of triangles
fore ach triangle
REAL32[3] – Normal vector
REAL32[3] – Vertex 1
REAL32[3] – Vertex 2
REAL32[3] – Vertex 3
UINT16 – Attribute byte count end

float tmp;
fread(&tmp,4,1,stream);
→read binary, then convert into float

(see example code)

OBJ (wavefront object) file format

Article | Discussion | Read | Edit | View history | Search

Wavefront .obj file

From Wikipedia, the free encyclopedia

For other uses, see Obj (disambiguation).

OBJ (or .OBJ) is a geometry definition file format first developed by Wavefront Technologies for its Advanced Visualizer animation package. The file format is open and has been adopted by other 3D graphics application vendors. For the most part it is a universally accepted format.

The OBJ file format is a simple data-format that represents 3D geometry alone — namely, the position of each vertex, the UV position of each texture coordinate vertex, normals, and the faces that make each polygon defined as a list of vertices, and texture vertices. Vertices are stored in a counter-clockwise order by default, making explicit declaration of normals unnecessary.

OBJ geometry format

Filename extension	.obj
Internet media type	application/x-tgif?
Developed by	Wavefront Technologies
Type of format	3D model format

Contents [hide]

http://en.wikipedia.org/wiki/Wavefront_.obj_file

OBJ (wavefront object) file format

- A text script (description) for 3D model
  - Vertex color / texture map
  - Supper material and light shading color
  - Easy to parse and modified
  - Group or hierarchical structure

```
mtllib [external .mtl file name]
...
```

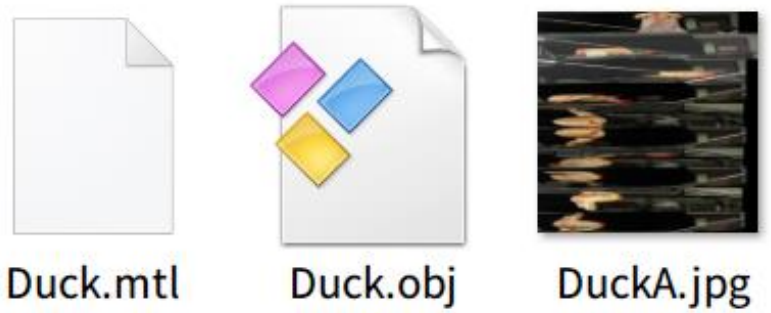
```
o [object name]
...
g [group name]
...
usemtl [material name]
...
```

```
v 0.123 0.234 0.345 1.0
```

```
vt 0.500 -1.352 [0.234]
```

```
vn 0.707 0.000 0.707
```

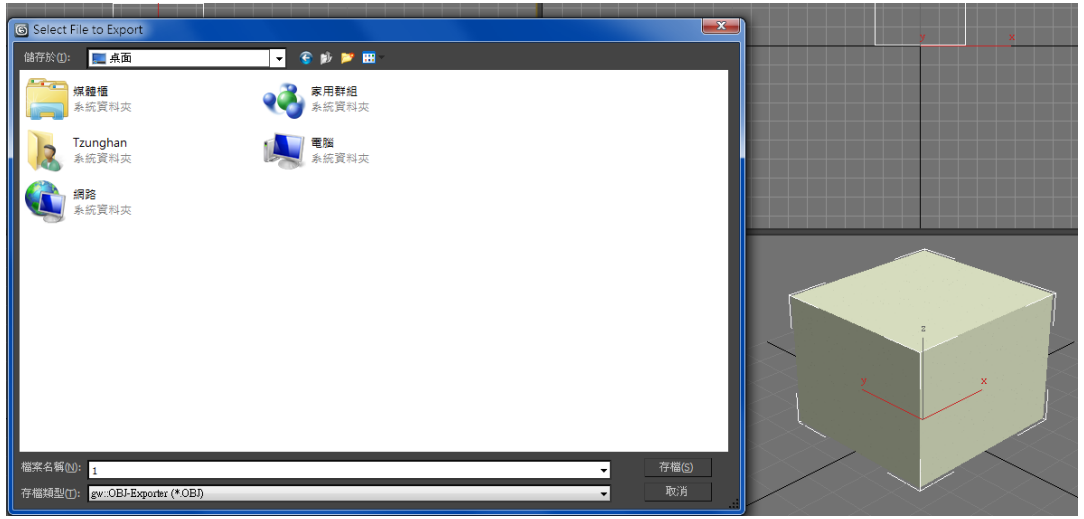
```
f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3 ...
```

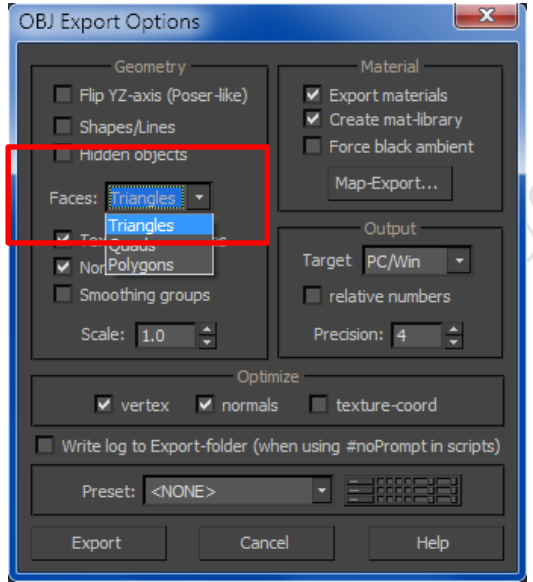


http://en.wikipedia.org/wiki/Wavefront_.obj_file

OBJ (wavefront object) file format—cont.

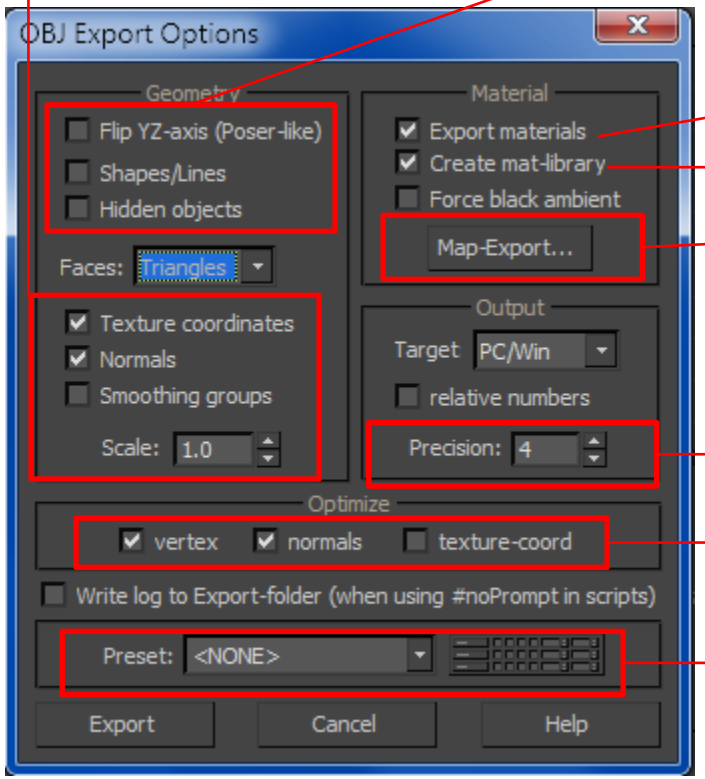
- Practice:
  - Use 3DMax, and draw a box, then export to a Obj file
  - Note the export setting.
  - Use other 3D software to open this file (ex. Meshlab)
  - Use “Text Editor” to manipulate the Obj file, then open by Meshlab program





OBJ (wavefront object) file format—cont.

勾選：匯出材質座標系
勾選：匯出法向量(點&面)
勾選：平滑化物件
模型放大倍率



其他家公司所提供的預設值

勾選：若有貼圖可連同照片匯出

勾選：產生材質說明檔

材質輸出的位置格式大小等設定

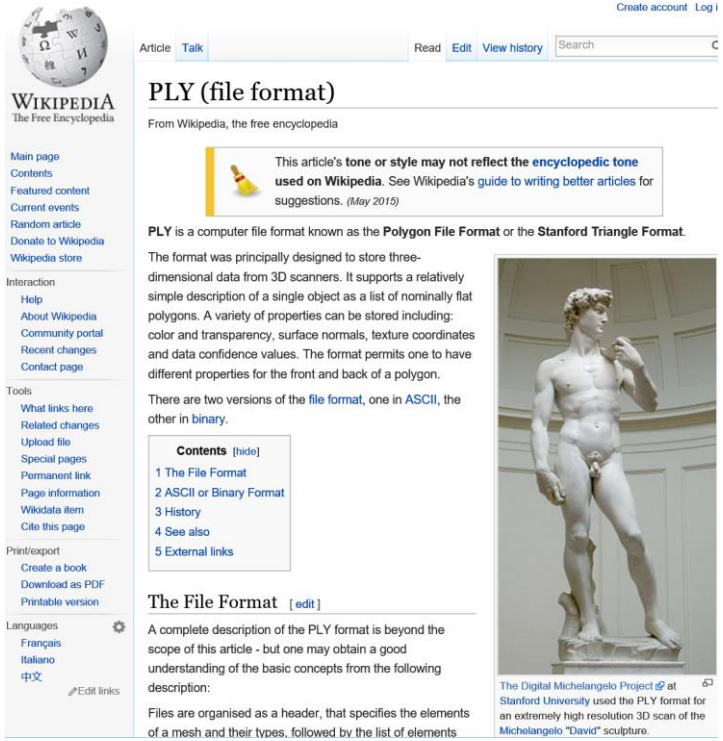
輸出數值的小數點位數

最佳化

其他家公司所提供的預設值

PLY (Polygon File Format)

- So called the Stanford Triangle Format
  - Support “triangle” only
  - Only “vertex color ” render



Create account Log i

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Français
Italiano
中文
Edit links

PLY (file format)

From Wikipedia, the free encyclopedia

This article's **tone or style may not reflect the encyclopedic tone used on Wikipedia**. See Wikipedia's guide to writing better articles for suggestions. *(May 2015)*

PLY is a computer file format known as the **Polygon File Format** or the **Stanford Triangle Format**.

The format was principally designed to store three-dimensional data from 3D scanners. It supports a relatively simple description of a single object as a list of nominally flat polygons. A variety of properties can be stored including: color and transparency, surface normals, texture coordinates and data confidence values. The format permits one to have different properties for the front and back of a polygon.

There are two versions of the file format, one in ASCII, the other in binary.

Contents [hide]
1 The File Format
2 ASCII or Binary Format
3 History
4 See also
5 External links

The Digital Michelangelo Project at Stanford University used the PLY format for an extremely high resolution 3D scan of the Michelangelo "David" sculpture.

The File Format [edit]

A complete description of the PLY format is beyond the scope of this article - but one may obtain a good understanding of the basic concepts from the following description:

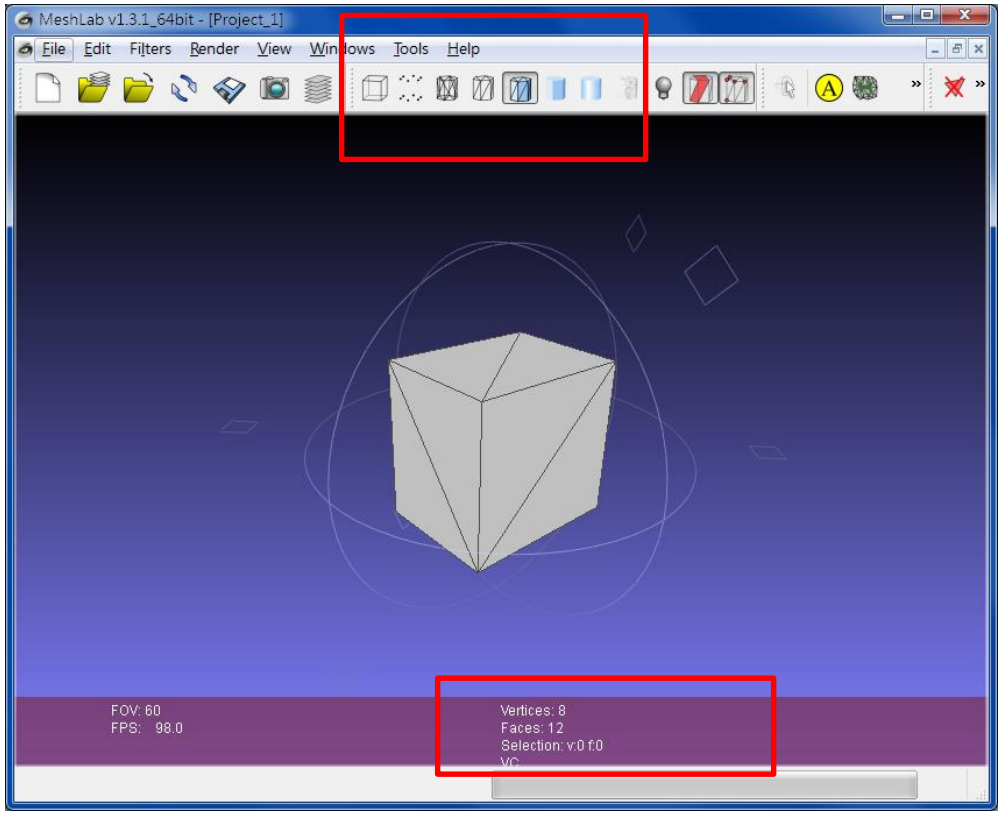
Files are organised as a header, that specifies the elements of a mesh and their types, followed by the list of elements

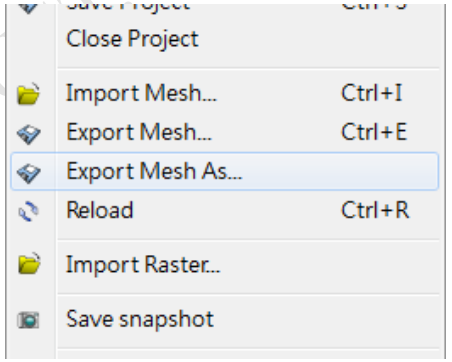
```
ply
format binary_little_endian 1.0
comment Tzunghan Lin (thl@mail.ntust.edu.tw)
comment This data is generated by TH TextureMap Tools
element vertex 138220
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
element face 276436
property list uchar int vertex_indices
end_header
```

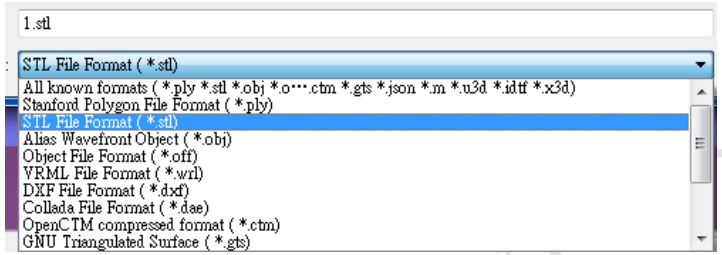

https://en.wikipedia.org/wiki/PLY_(file_format)

Meshlab (3D tools)

Display mode







Model information

3D format (animation): glTF

GLB

Last modified by Farooq Sheikh on 2019/07/17 07:55

What is a GLB file?

GLB is the binary file format representation of 3D models saved in the GL Transmission Format (glTF). Information about 3D models such as node hierarchy, cameras, materials, animations and meshes in binary format. This binary format stores the glTF asset (JSON, .bin and images) in a binary blob. It also avoids the issue of increase in file size which happens in case of glTF. GLB file format results in compact file sizes, fast loading, complete 3D scene representation, and extensibility for further development. The format uses model/gltf-binary as MIME type.

GLB File Format

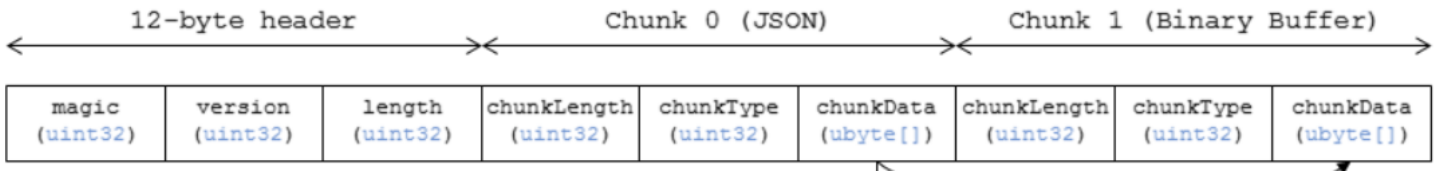
The content delivery methods used by glTF result in extra processing to decode the base-64 encoded binary data and also increases the file size by ~33%. These delivery methods, which contributed to the formation of GLB file format, include:

- glTF JSON points to external binary data (geometry, key frames, skins), and images.
- glTF JSON embeds base64-encoded binary data, and images inline using data URIs.

GLB as a container format was introduced as binary file format for representation of glTF asset in a binary blob to avoid the issues caused by glTF. GLB file format specifications should be referred for any reader/writer implementation of the same for applications development.

File Structure

GLB file format is based on little endian and its structure is as shown below:



https://wiki.fileformat.com/specification/3d/glb/

3D format (animation): glTF

glTF Basics

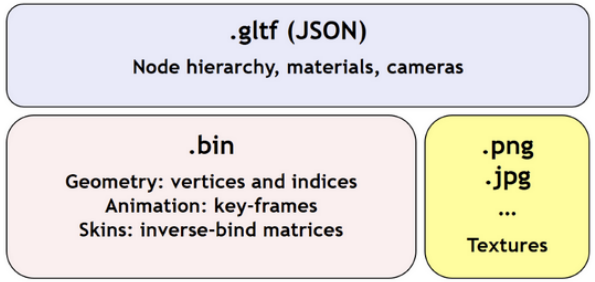
This section is non-normative.

glTF assets are JSON files plus supporting external data. Specifically, a glTF asset is represented by:

- A JSON-formatted file (`.gltf`) containing a full scene description: node hierarchy, materials, cameras, as well as descriptor information for meshes, animations, and other constructs
- Binary files (`.bin`) containing geometry and animation data, and other buffer-based data
- Image files (`.jpg` , `.png`) for textures

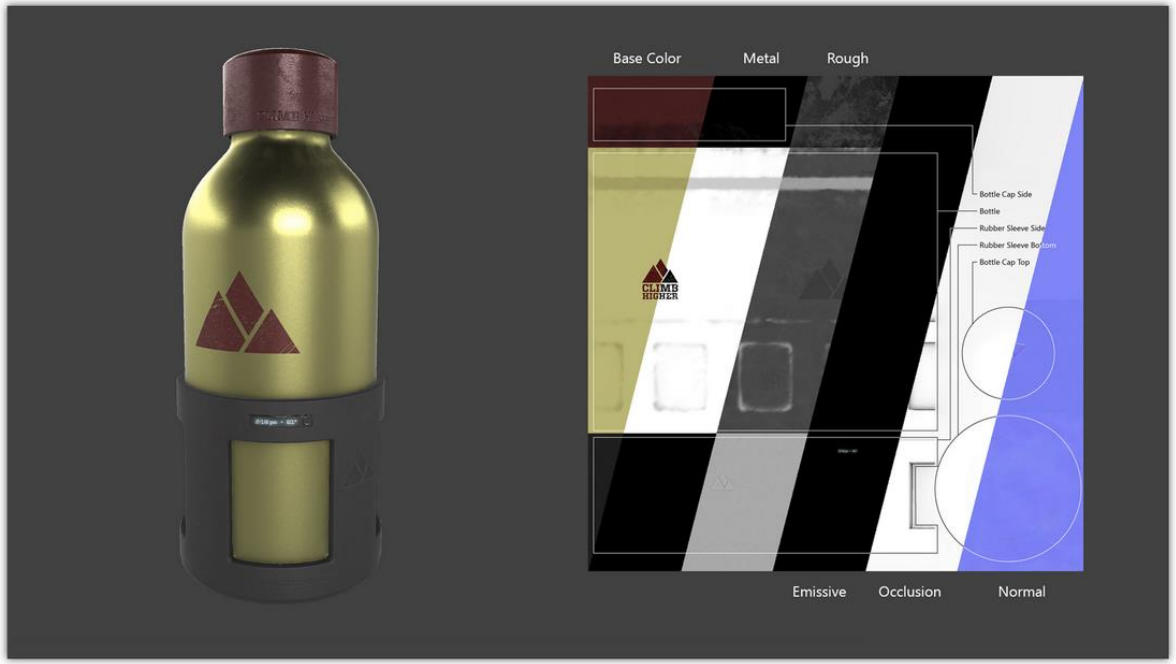
Assets defined in other formats, such as images, may be stored in external files referenced via URI, stored side-by-side in GLB container, or embedded directly into the JSON using data URIs.

Valid glTF asset must specify its version.



Materials

glTF defines materials using a common set of parameters that are based on widely used material representations from Physically-Based Rendering (PBR). Specifically, glTF uses the metallic-roughness material model. Using this declarative representation of materials enables a glTF file to be rendered consistently across platforms.



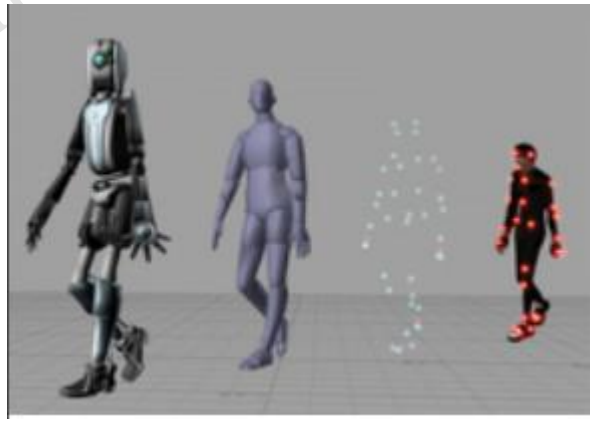
https://github.com/KhronosGroup/glTF/tree/master/specification/2.0

Supplementary material

- Motion capture

From CMU Prof. Kanade

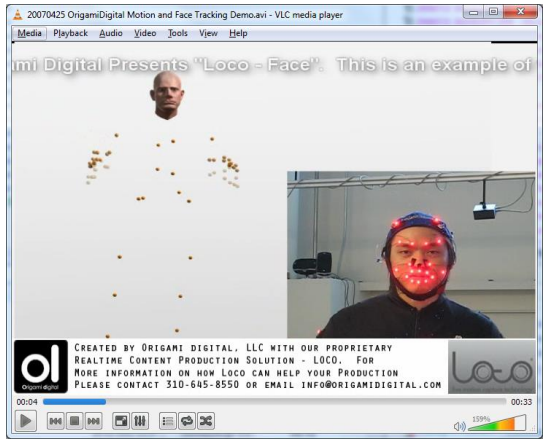
Movie Avatar

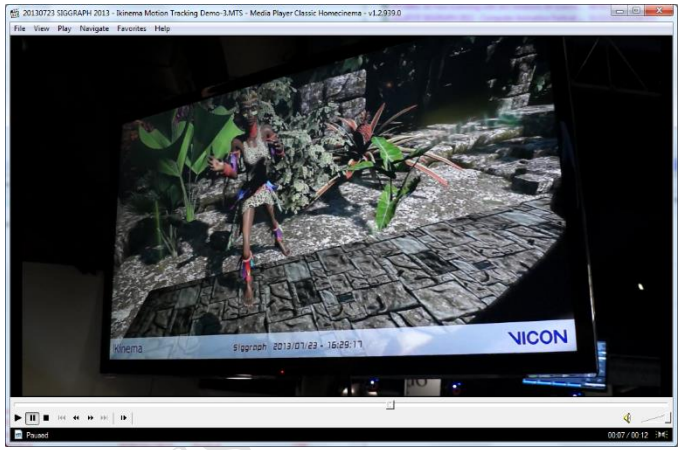


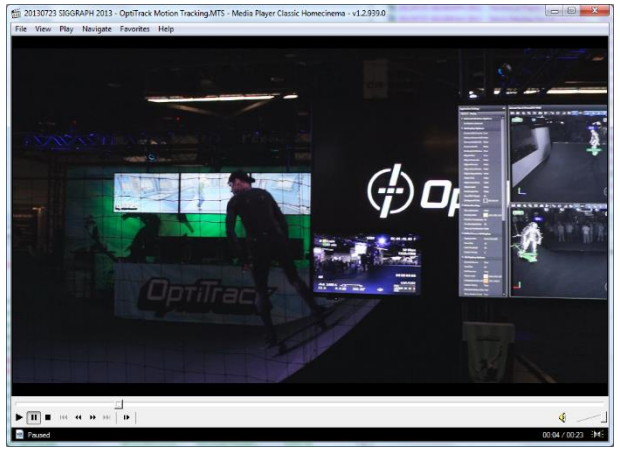
CMU motion capture database, http://mocap.cs.cmu.edu/
http://www.cadengineering.co.in/home-6/products

Supplementary material

- Motion capture







2007 OrigamiDigital Motion and Face Tracking Demo
SIGGRAPH 2013 - Ikinema Motion Tracking Demo-3
SIGGRAPH 2013 - OptiTrack Motion Tracking

Supplementary material

- Numerical Error
- Truncated error (Truncation error):
  - Ex. 3.141592654→3.141592

- Round off error:
  - Ex. 3.141592654→3.141593

Supplementary material: Euler formula

is called a regular mesh. The number of vertices (V), edges (E), and faces (F) in a closed polygonal mesh are related by the Euler-Poincare formula

$$V + F - E = 2(1 - g) \tag{8.1}$$

where g, the genus, denotes the number of holes/handles in the mesh. The right-hand side of the above equation is called the Euler Characteristic. For the torus in Fig. 8.3 and the Möbius strip in Fig. 8.7b, $g = 1$, and hence $V + F = E$. For polyhedral objects without any holes, $V + F = E + 2$. This equation is generally referred to as the Euler's formula. In a triangular mesh without holes, the average valence of a vertex is six, and we can get an estimate of the number of faces and edges in terms of the vertices as

$$F \approx 2V$$
$$E \approx 3V \tag{8.2}$$

Also in a triangular mesh, every face has three edges, and every edge is counted twice while counting the number of faces. Therefore the number of faces and edges are connected by the equation $E = 3F/2$.

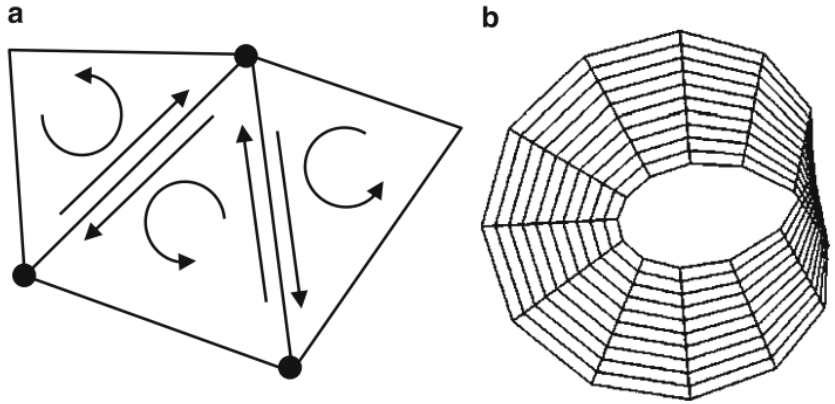


Fig. 8.7 (**a**) Compatible faces in an orientable mesh. (**b**) The Möbius strip is an example of a non-orientable mesh

Supplementary material: Mesh simplification

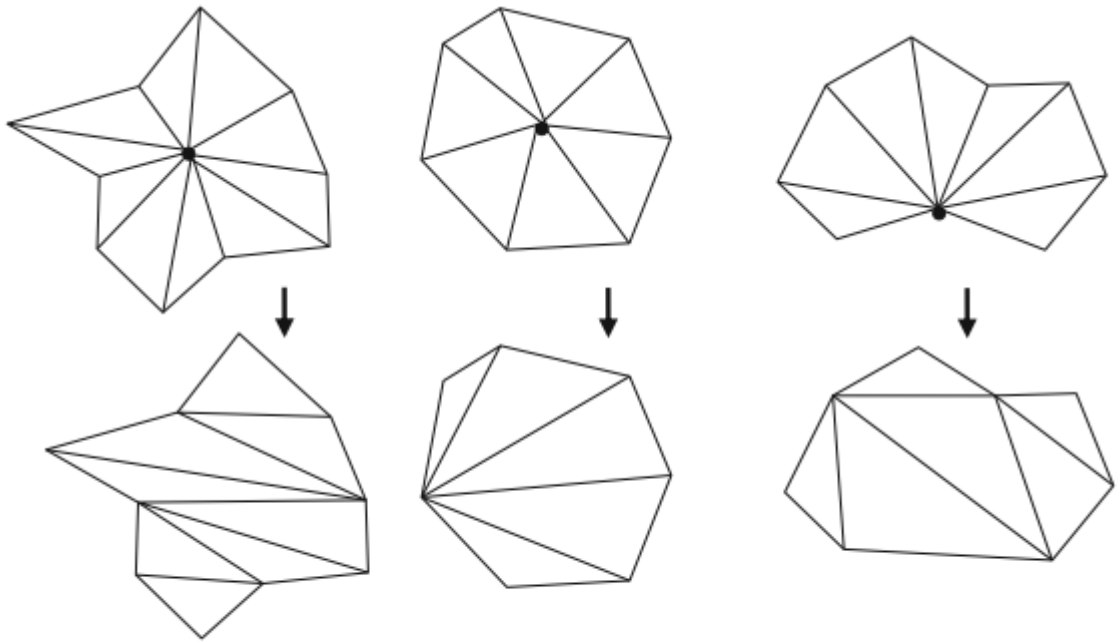


Fig. 8.18 Removal of internal and boundary vertices and the triangulation of the resulting polygons

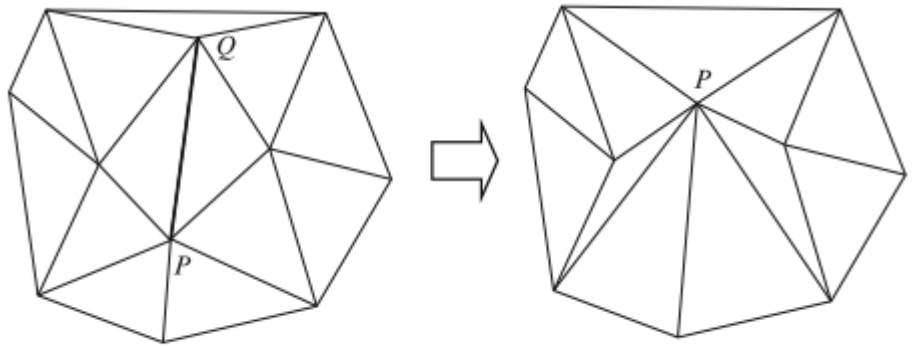


Fig. 8.19 An edge collapse operation performed by moving the vertex P towards Q

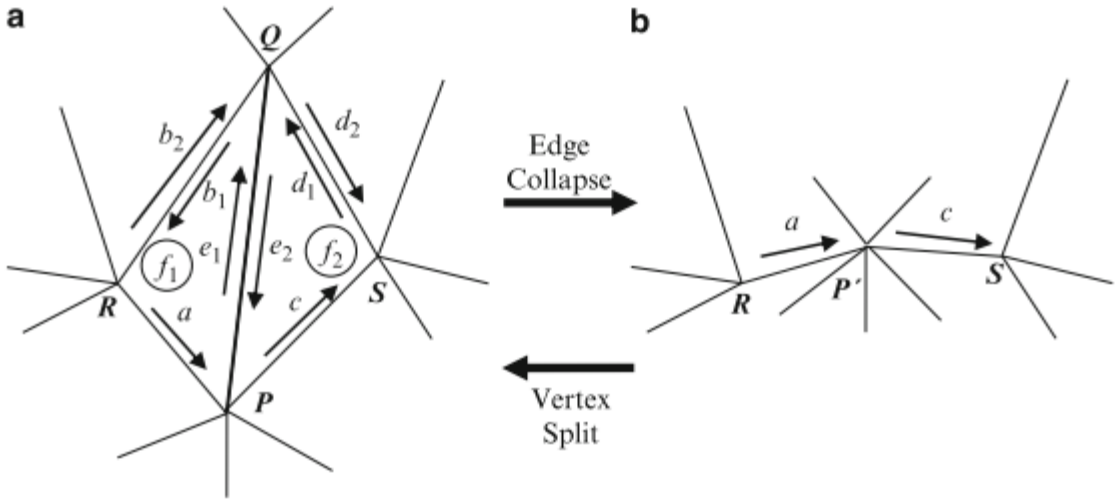


Fig. 8.20 Edge references used by the edge collapse algorithm in Listing 8.9

Supplementary material: Mesh subdivision

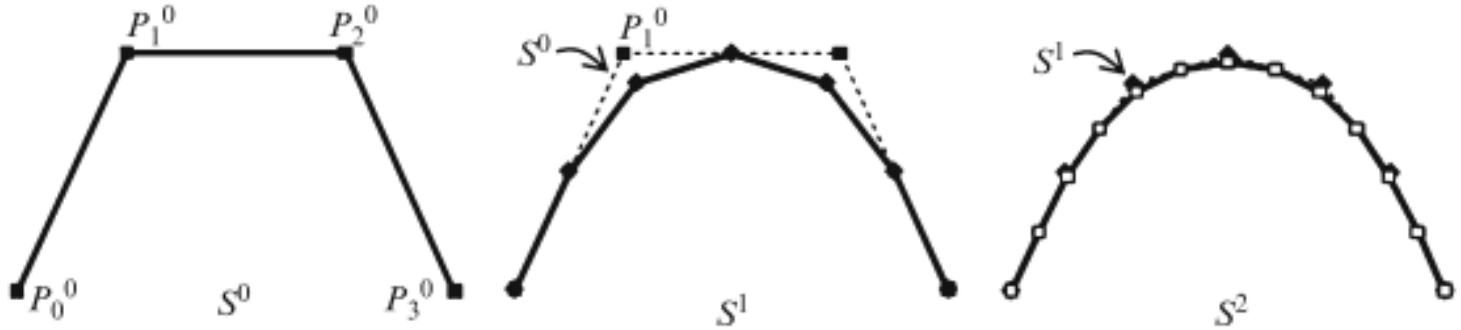
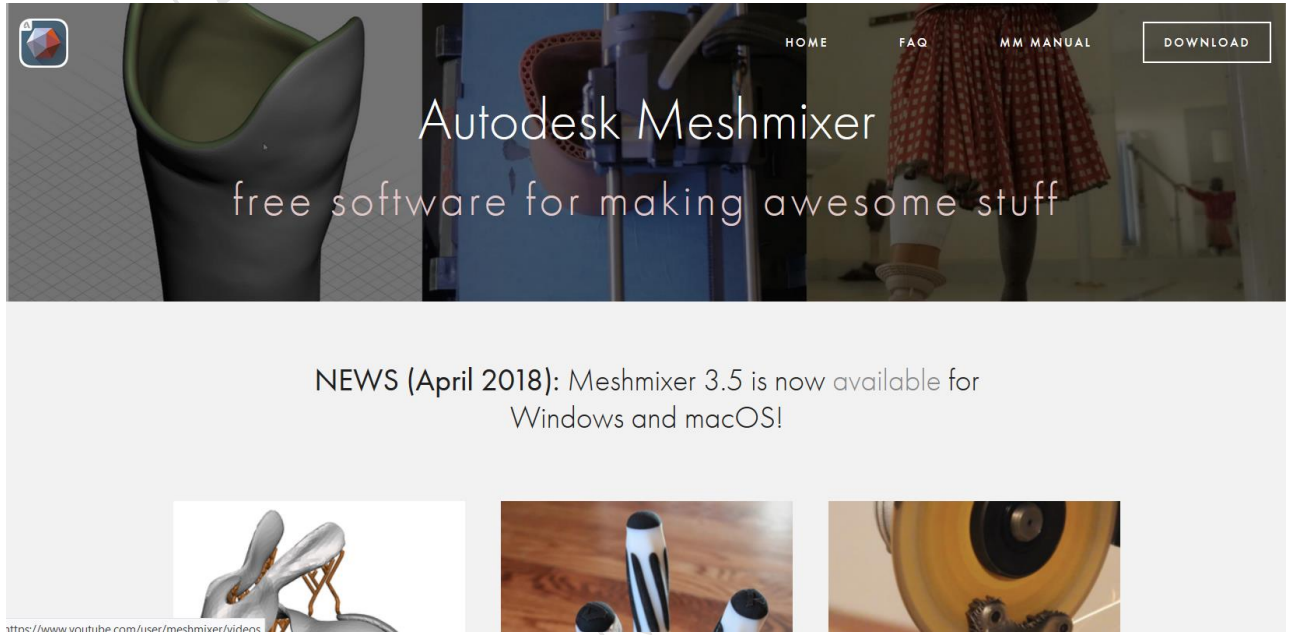
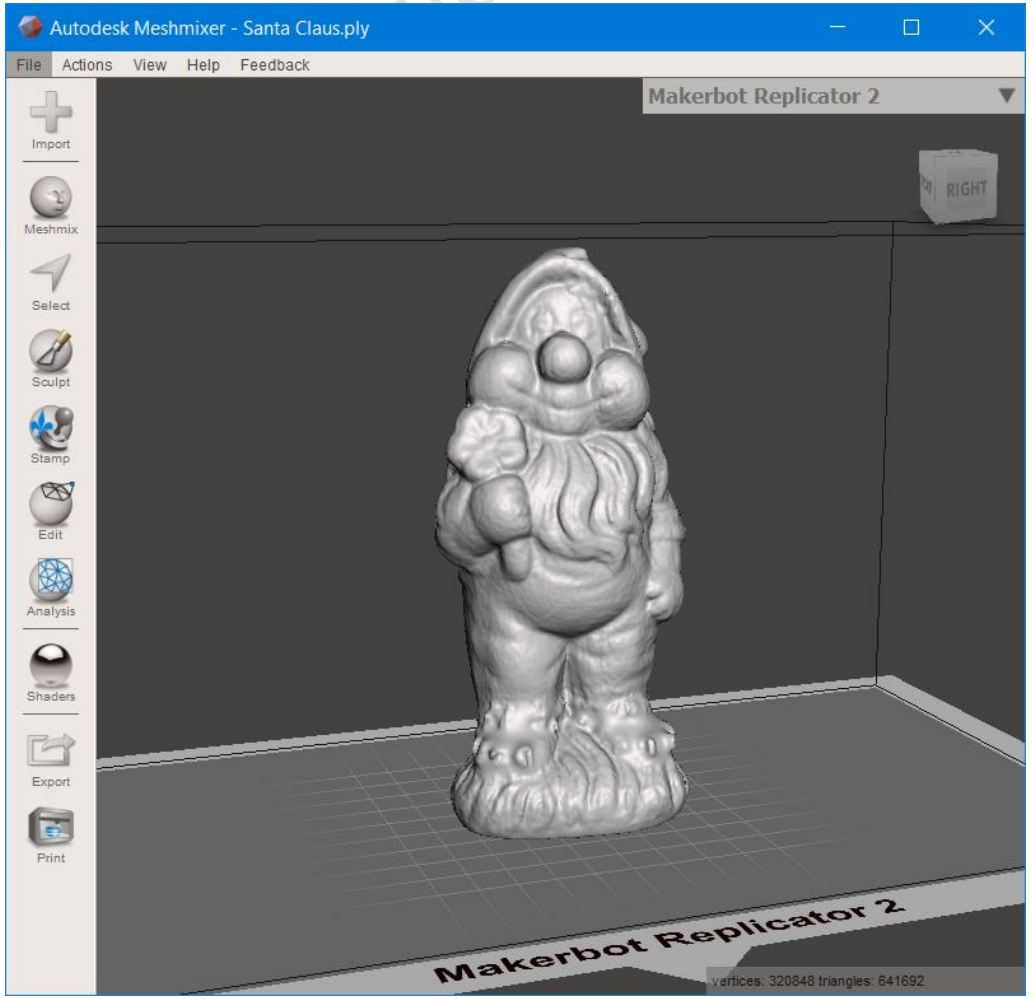


Fig. 8.23 A control polygonal line and the next two levels of its subdivision

Fig. 8.25 Triangular subdivision scheme

Meshmixer (opensource - free)

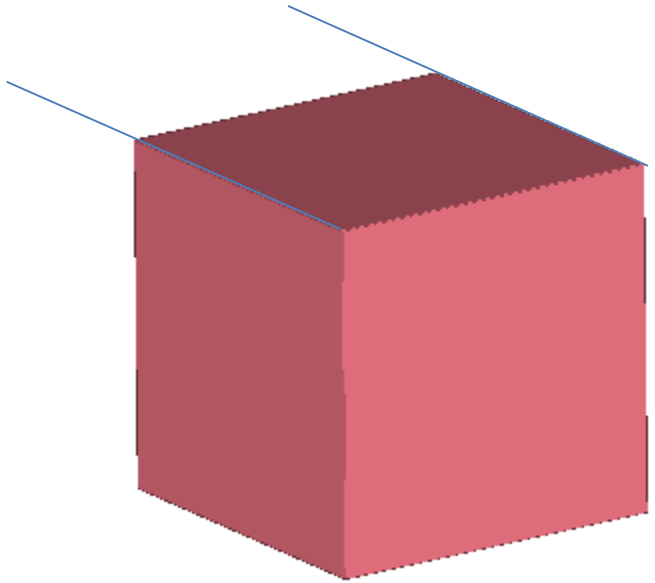




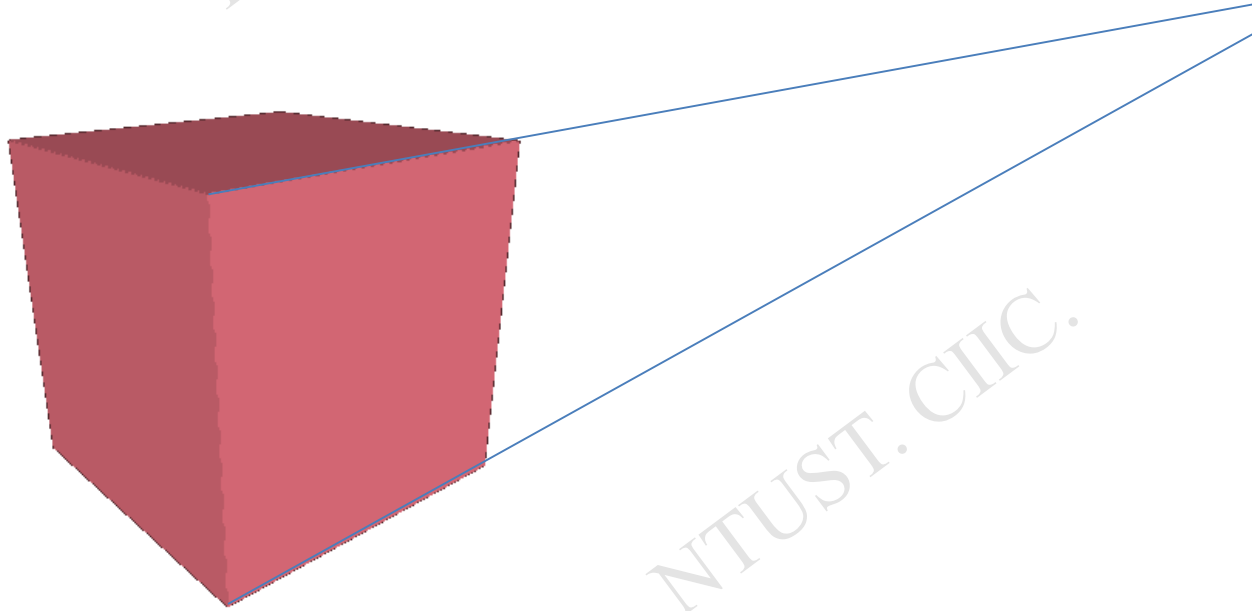
http://www.meshmixer.com/

Parallel projection & Perspective projection

- Note: In graphics, pinhole camera belongs to perspective projection

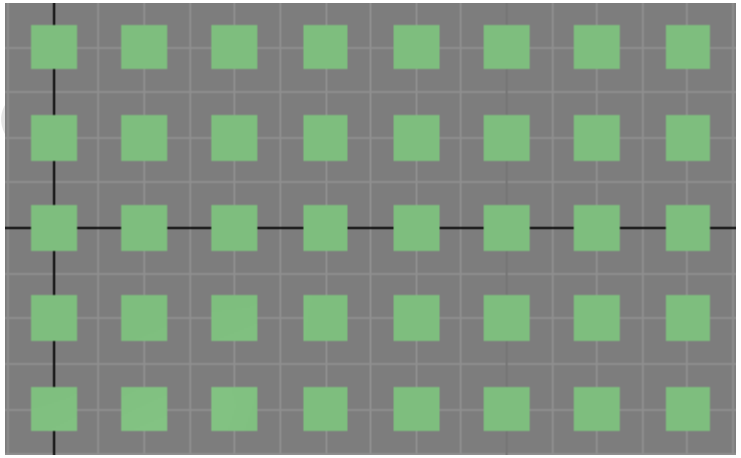


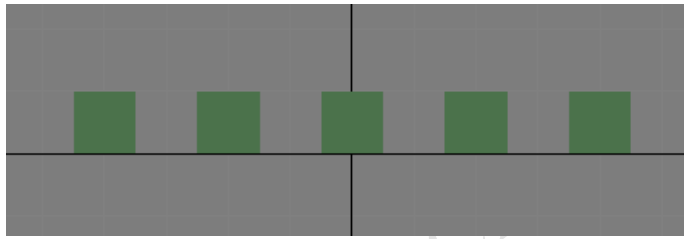
Parallel projection

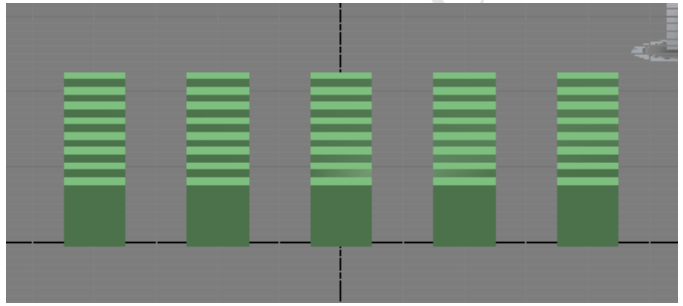


Perspective projection

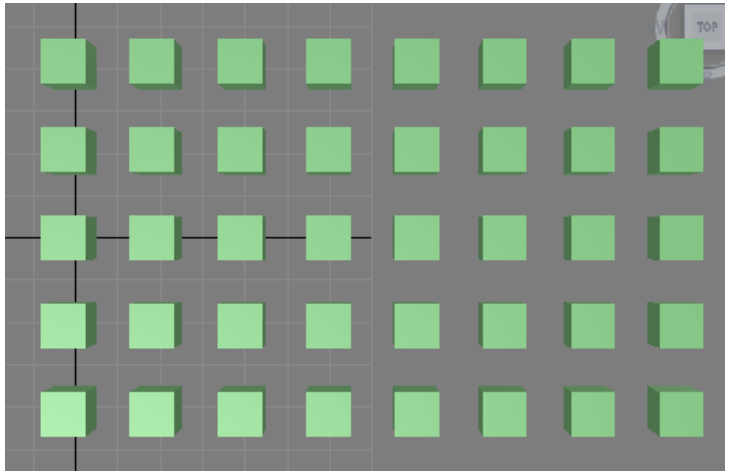
Parallel projection

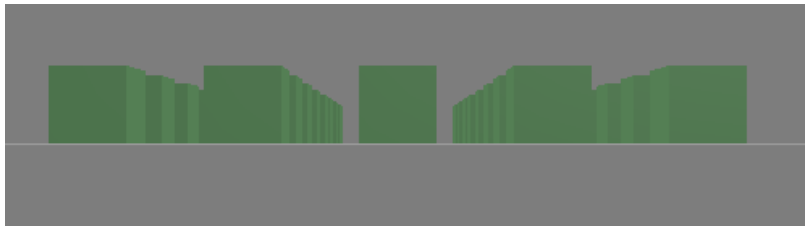


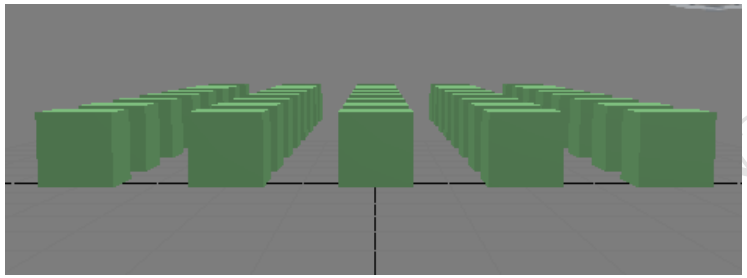




Perspective projection

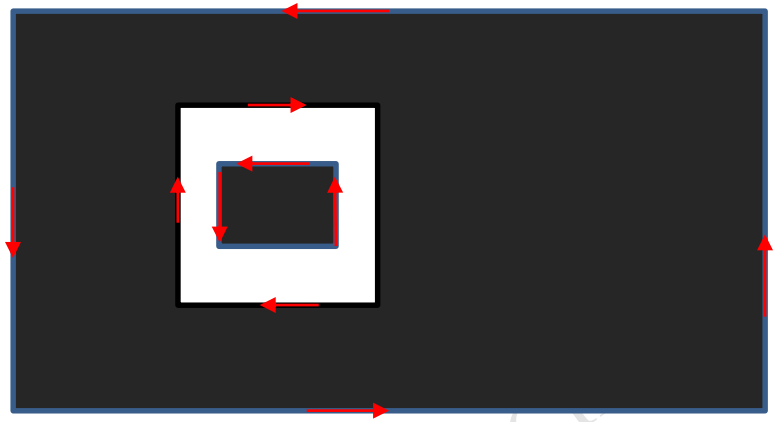


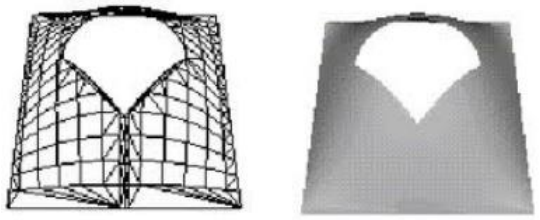




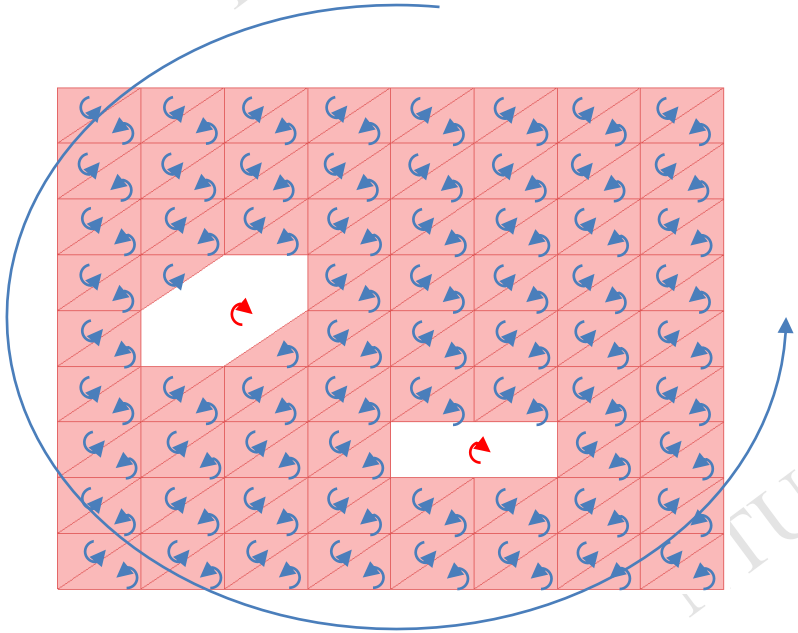
Supplementary material

- Trimmed surface in openGL





Trimmed surface in openGL



Supplementary material

- Half-edge: assist for searching “boundary of surface”

CIT 色彩與照明科技研究所
Graduate Institute of
Color and Illumination Technology

Find us on: facebook