Medical Images
Session 7

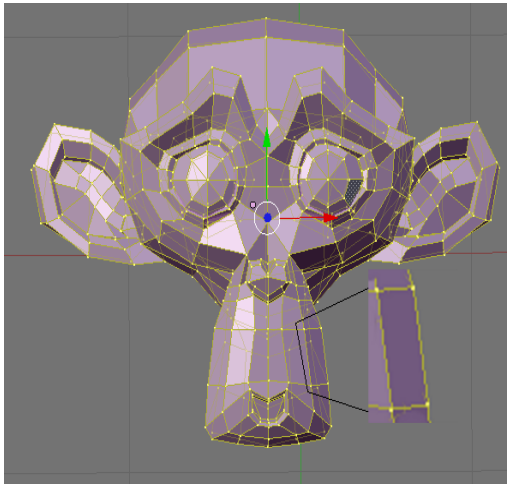
Surface rendering

D. Tost

3D Models

Surface models

Polygonal meshes (vectorial)



Volume models

Voxel models (raster)

Image from Kniss at al., 2002

Geometry and topology

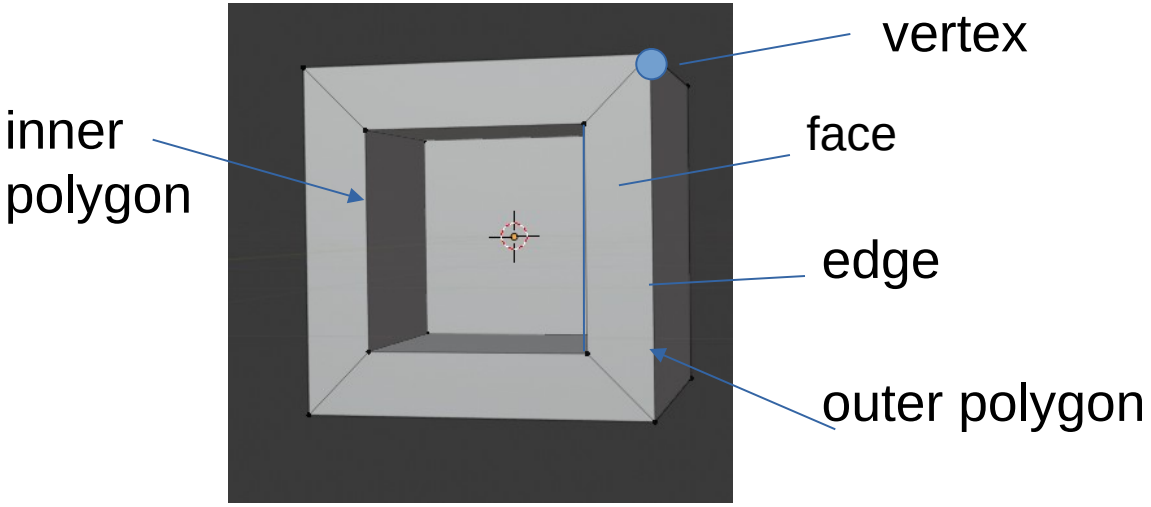
Geometry:

Vertices: 3D points represented by cartesian coordinates (x, y, z)

Edges: line connections between vertices

Faces: closed sequence of edges, generally triangles or quads.

Polygons: set of co-plannar adjacent faces



Geometry and topology

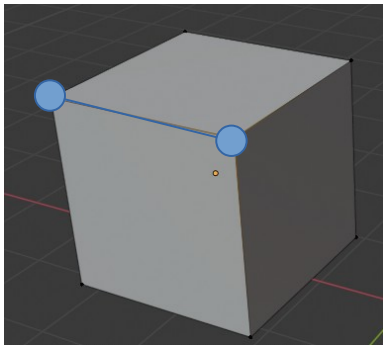
Topology:

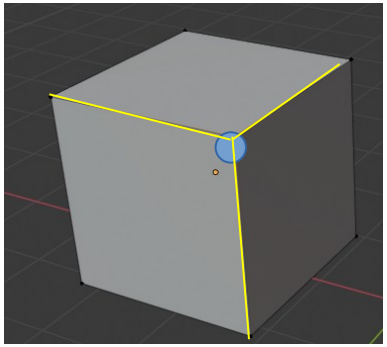
Connection between geometric elements

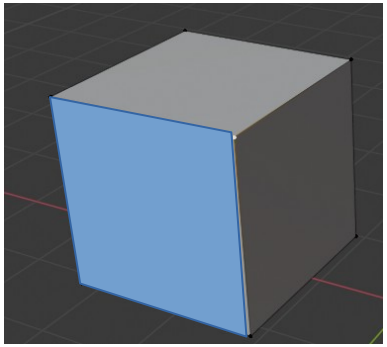
Edge-Vertex: 2 extreme vertices
Vertex- Edge: all edges sharing the vertex

Edge-Face: 2 faces sharing an edge
Face-Edge: edges forming the contour of the face (external and internal edges)

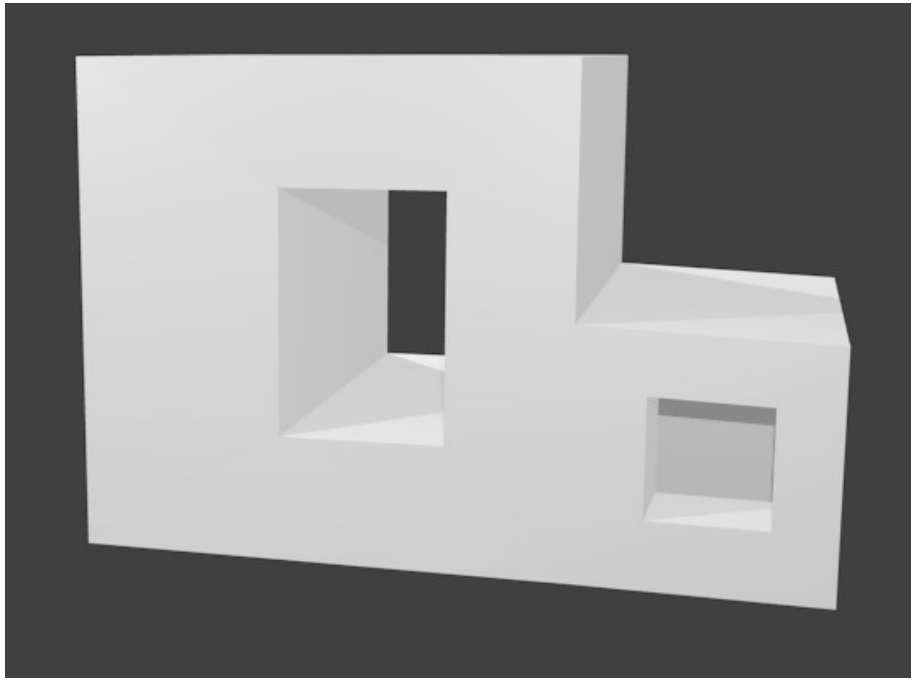
Vertex-Face: all faces sharing the vertex
Face-Vertex: all vertexs of a face





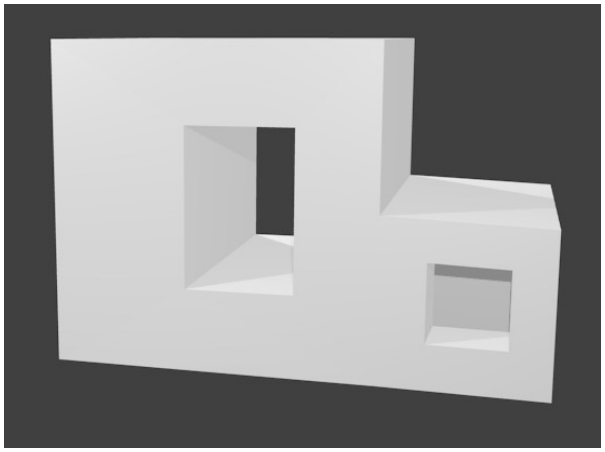


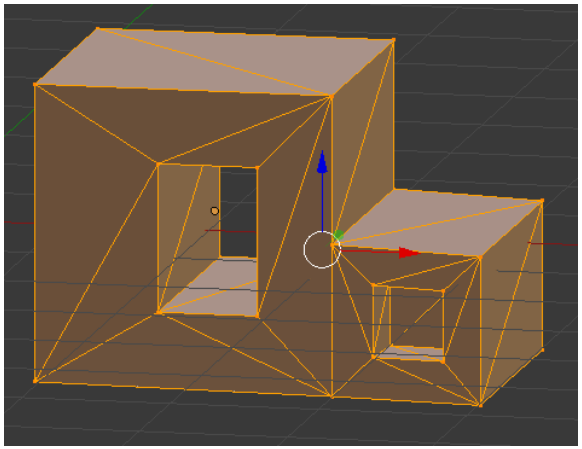
Geometry and topology



2 holes: one passing through

Geometry and topology





Triangular co-plannar faces

Surface Models Data

- Geometry:
  - coordinates of the vertices of the mesh
  - Normal vectors of the faces of the mesh
- Topology: VE + VF or EE+ EV+ VF

e.g:

-faces containers of polygons

- polygons containers of topologically sorted vertices (implicit edge)

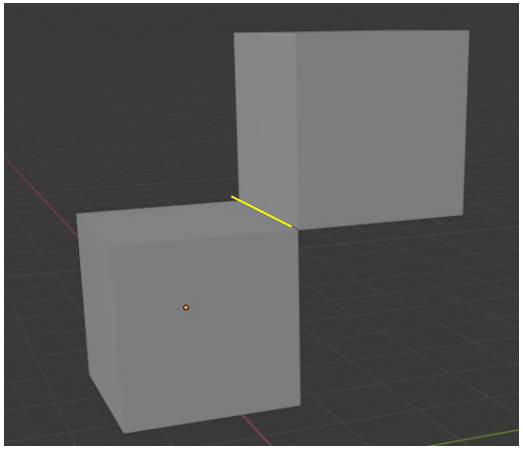
Characteristics

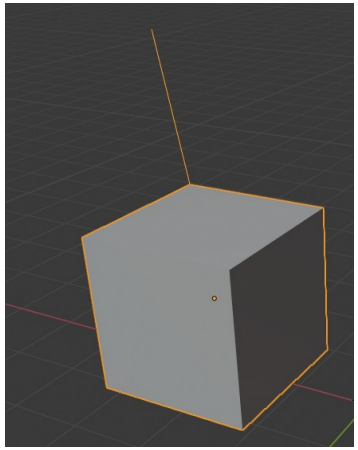
Solid models

Closed surface (continuous, without breaks)

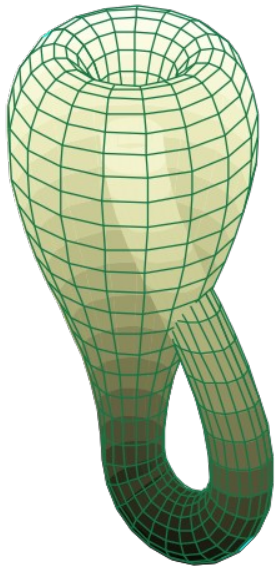
Orientable surface (outside/inside)

2-Manifold: each point is homeomorphic to a disc

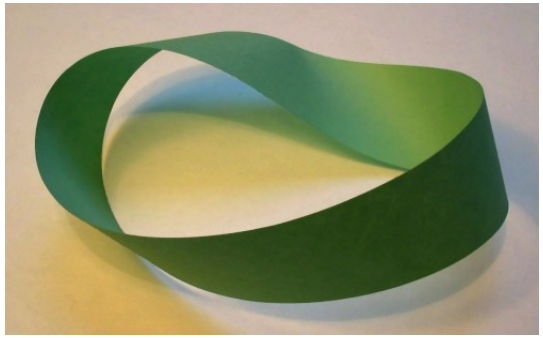




Example of 2 non-2-manifold objects



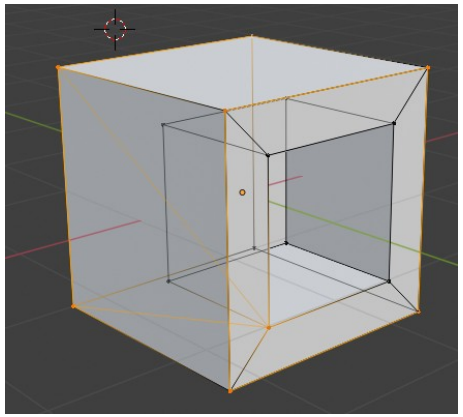
Non-orientable moebius surfaces

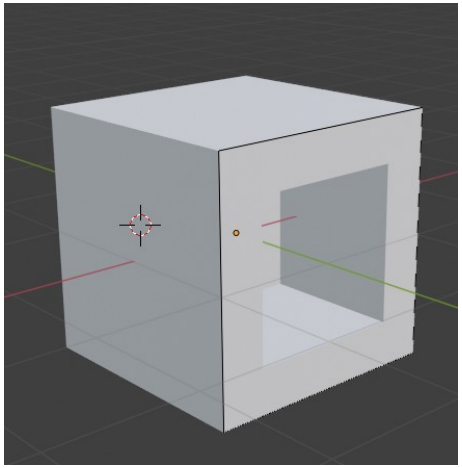


Non-manifold

Solid models

Non-manifold objects and non-orientable objects are not printable

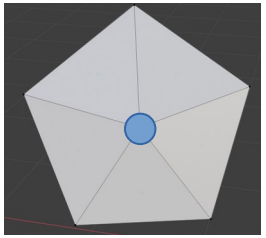


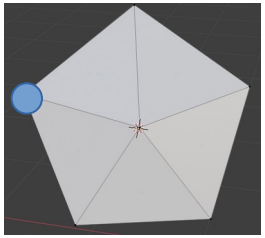


Typical problems:

- Duplicated vertices
- Isolated components of a same solid
- Inner faces
- Isolated edges

- **Edges being shared by more than 2 faces**
- **Faces incident to a vertex not forming a closed or open fan**

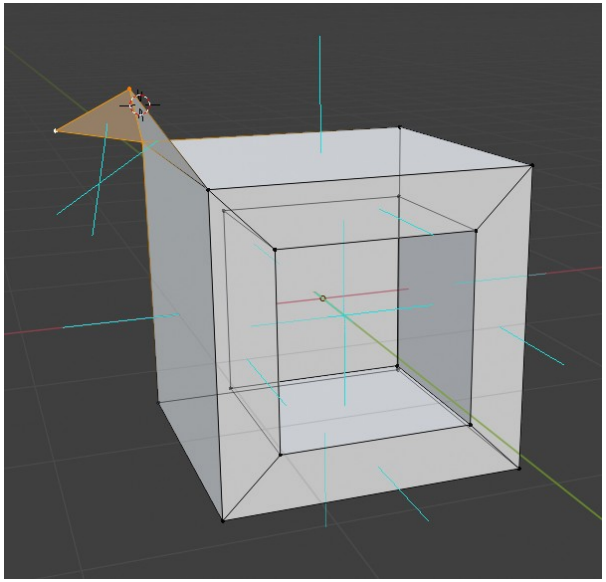




Non-manifold

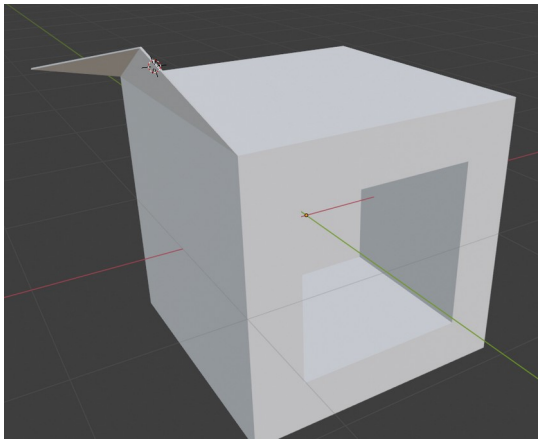
Solid models

Non-manifold objects and non-orientable objects are not printable



Typical problems:

- Duplicated vertices
- Inner faces
- Isolated edges
- Edges being shared by more than 2 faces
- Isolated components of a same solid



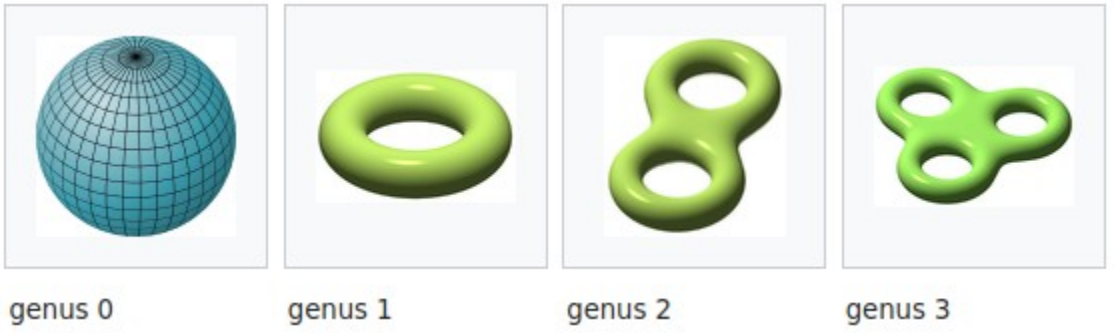
Boundary Representation

Boundary representation (B-rep): the solid is represented through its boundary surface

Euler-Poincaré formula:

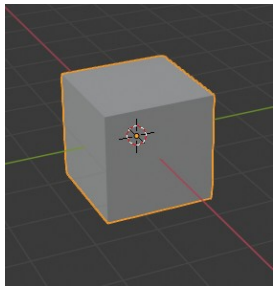
V – E + F - L = 2(B - G)

V = # vertices
E = # edges
F = #faces
L = #inner loops
B = #bodies
G = #genus (number of holes passing through)

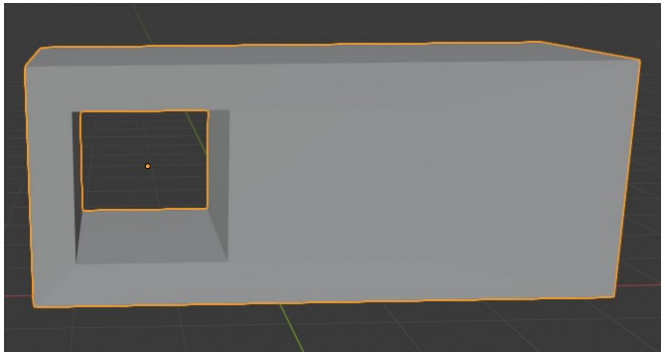


Boundary Representation

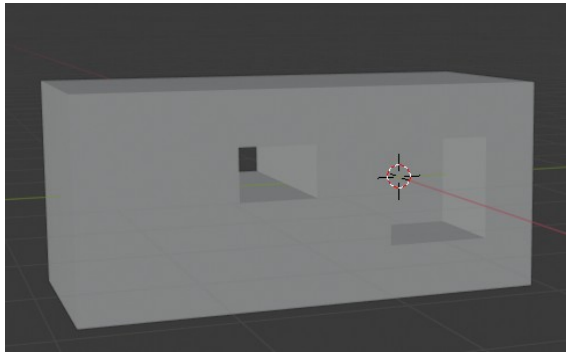
Example 1: 8-12+6-0 = 2(1-0)
V = 8
E= 12
F = 6
L = 0
B = 1
G = 0



Example 2: 16-24+10 -2 = 2(1-1)
V = 16
E= 24
F = 10
L = 2
B = 1
G = 1



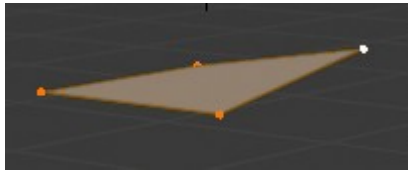
Example 3: 24-36+15 - 3 = 2(1-1)
V = 24
E = 36
F = 15
L = 3
B = 1
G = 1



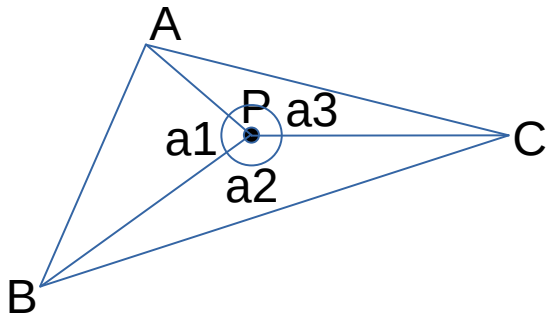
Why triangular faces?

In general, triangular faces are preferred for geometric models. Therefore, often planar faces of a surface are subdivided into co-plannar triangles.

- Triangles are always flat (3 points define a plane)
- They are convex
- They can be depth-sorted
- They can be rendered very efficiently (hardware supported)
- It is easy to determine if a point is inside a triangle



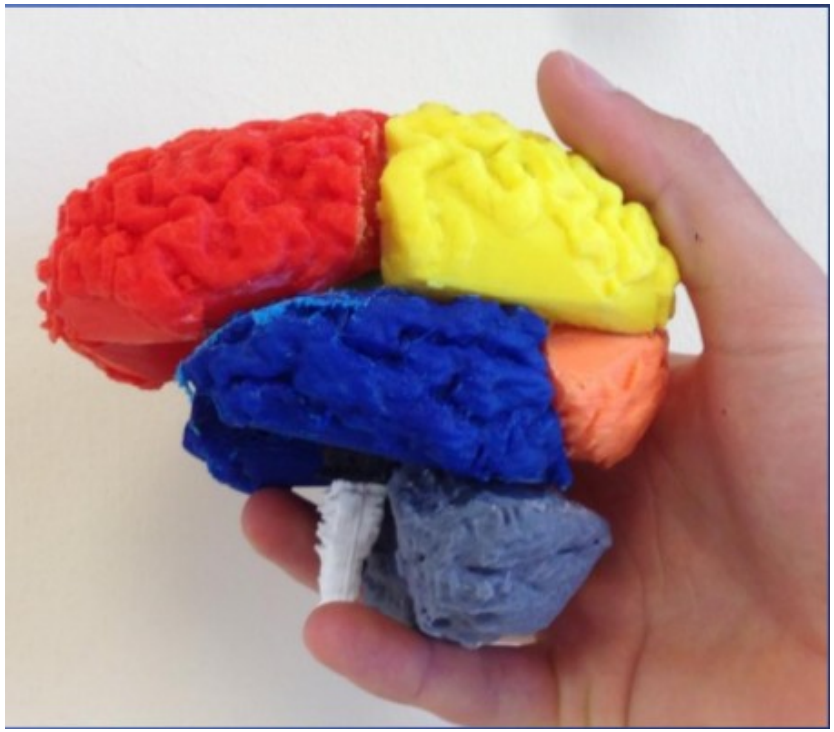
A non-plannar quad



a1+ a2+ a3 = 180º

Surface Models Processing

- Processes
  - Creation
    - Reading/writing and existing file
    - Modeling from scratch
    - 3D scanning
    - Extraction from volume models
  - Render
  - Process
    - Simplification the mesh
    - Splitting the mesh
    - Uniting/splitting the mesh
    - Recomputing the topology of the mesh
    - Smoothing the mesh
    - Removing holes
    - ...
  - 3D printing



Surface models creation

- Mesh editors; Modeling tools
  - CAD purposes: Rhinoceros, SolidWorks
  - «Artistic» purposes: Maya, Blender, 3DMax, Krita
- Mesh processing and refactoring
  - Meshlab
  - Netfabb
  - Slic3r
- Mesh 3D printing

File formats for polygonal meshes

General format:

.ply (http://paulbourke.net/dataformats/ply/)

.stl

And many others

Application dependent format:

.blend (Blender)

.3ds (Autodesk)

.obj (wavefront)

.vtk (paraview, slicer, vtk libs)

And many others

STL format for polygonal meshes

ASCII: made of ASCII characters (bytes), readable and editable with text editors.

Binary: encoding the information at the bit level; more compact; not editable with a text editor.

```
solid name

facet normal ni nj nk
    outer loop
        vertex v1x v1y v1z
        vertex v2x v2y v2z
        vertex v3x v3y v3z
    endloop
Endfacet

endsolid name
```

http://en.wikipedia.org/wiki/STL_%28file_format%29

Practical work
Read an stl model

Edit o look inside liver.stl

>> In a terminal do : `more liver.stl`

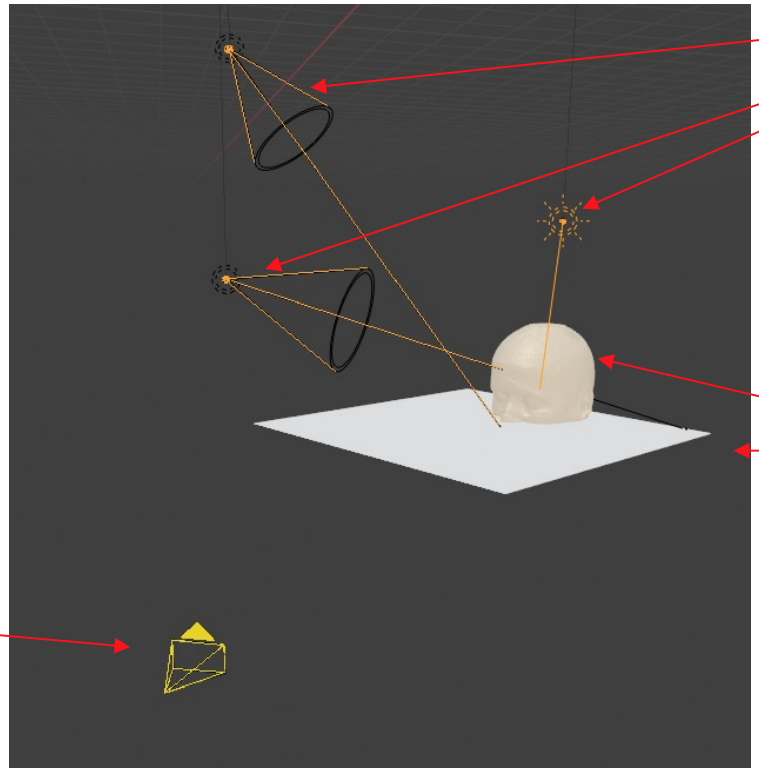
Is it a binary or an ascii file?

Do you understand the structure?

Rendering

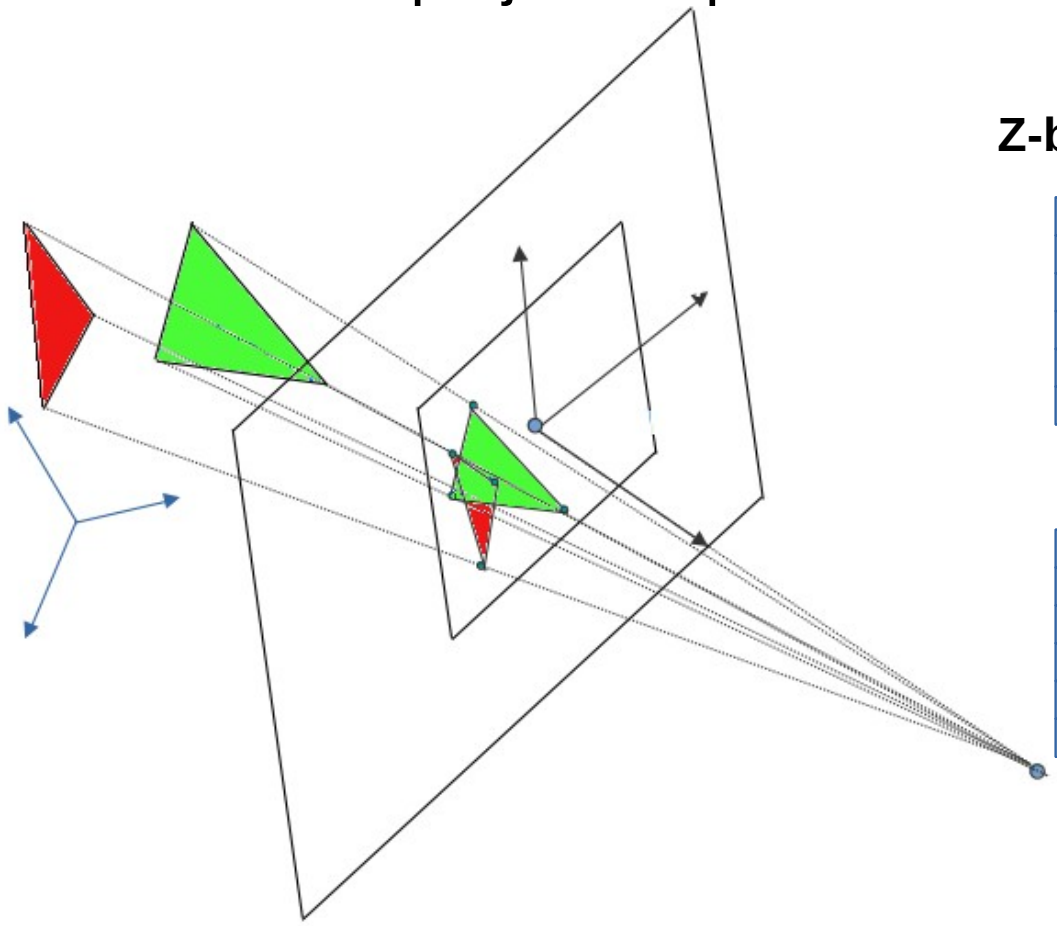
Elements

Rendering consists of obtaining a rasterized image of the 3D model viewed by a virtual observer and ligheçted by virtual lights

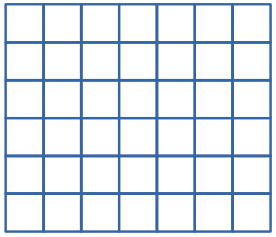


Rendering of polygonal mesh

- Based on the projection of the vertices of polygons of the scene in the projection plane



Z-buffer algorithm



Depth buffer (z-buffer)
Stores the depth (z) of the visible points

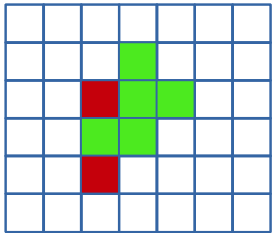
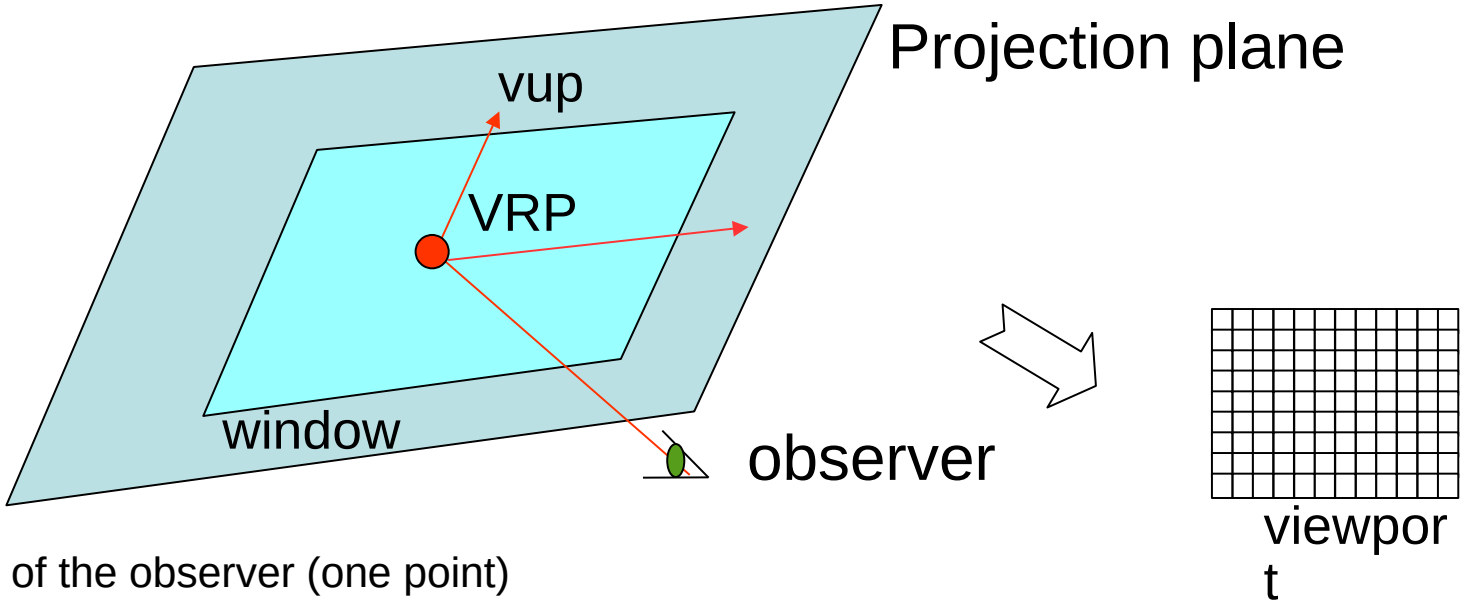


Image buffer
Stores the color of the pixels

Camera

It expresses where is the virtual observer, at which points it focuses, how is the camera oriented and which part of the projection will be rasterized in the final image (window in the projection plane).



OBS: position of the observer (one point)

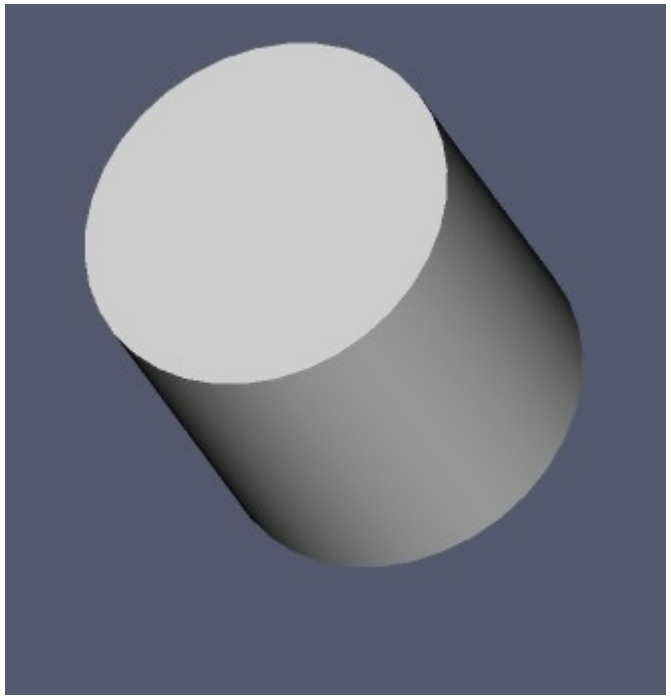
VRP: view reference point, projection center

VUP: vector that indicates the camera verticality

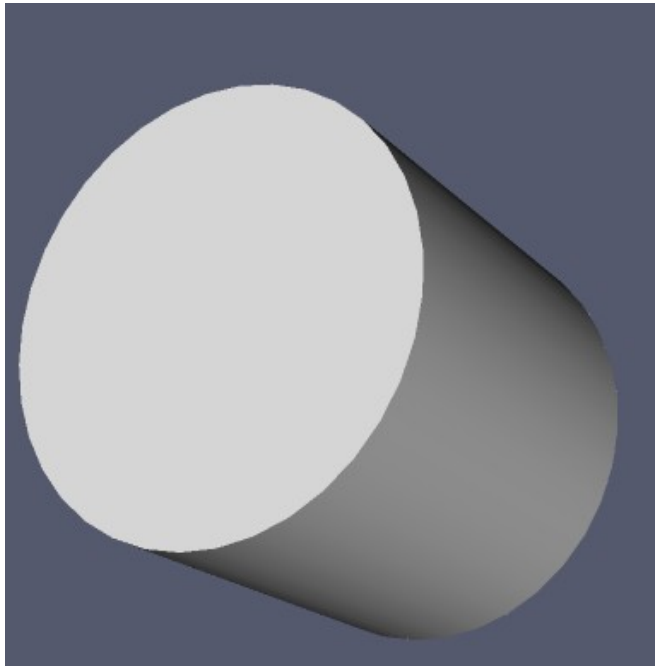
Window: rectangle of the projection plane that will be be rasterized.

Camera

Projection type

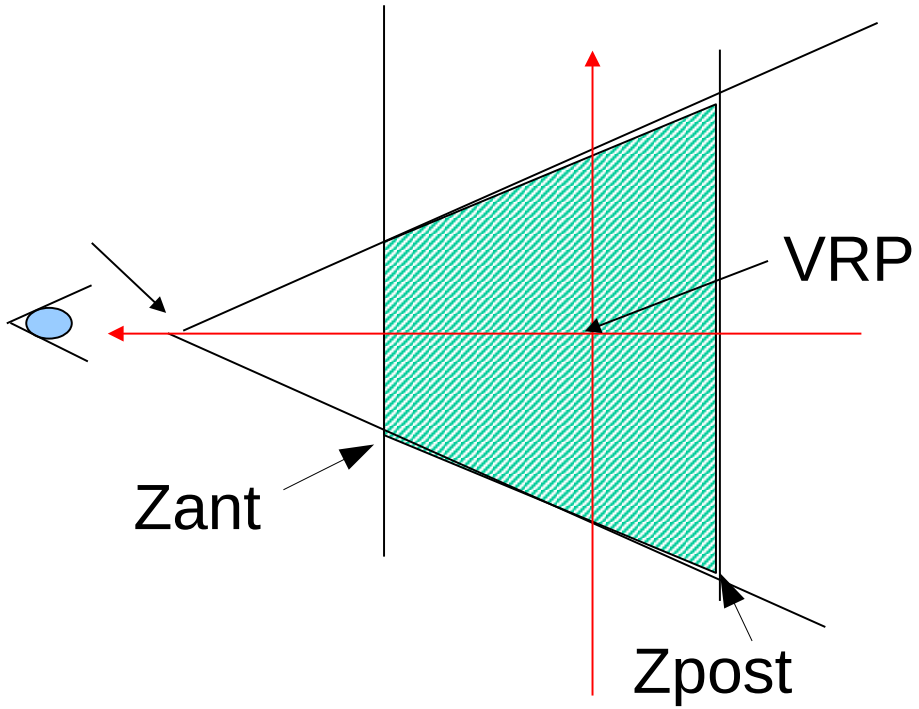


Orthographic



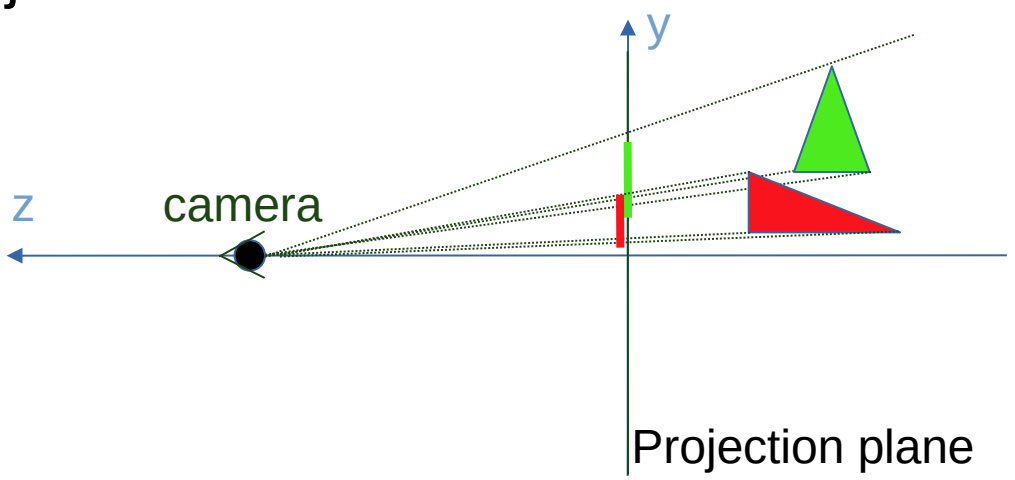
Perspective

Camera

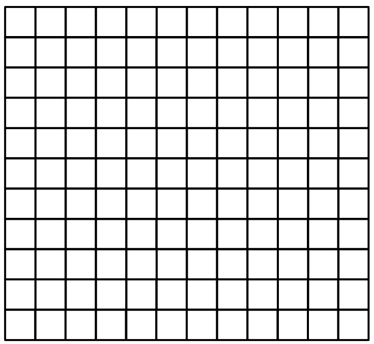


Rendering of polygonal mesh

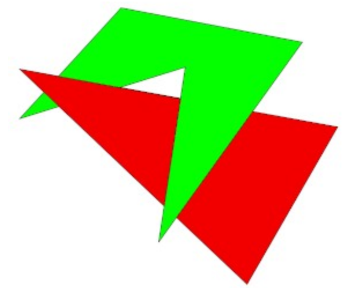
- Based on the projection of the vertices of polygons of the scene in the projection plane
- For each face of each mesh, project it. At every pixel of the projection compare the depth of the projected point to the current stored depth in the Z.Buffer. If it is less (nearer the observer), replace the depth and the color. Otherwise, reject.

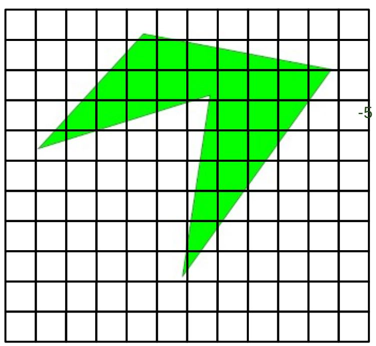


Rendering of polygonal mesh

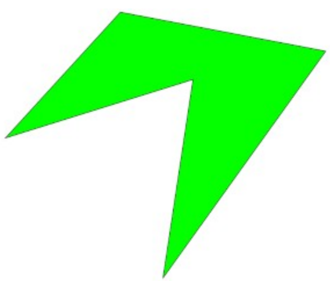


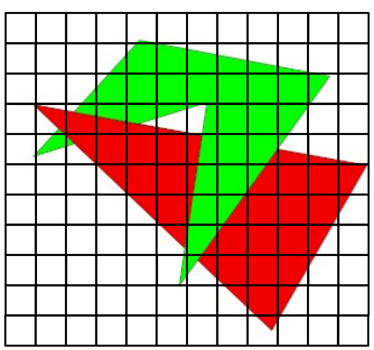
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10



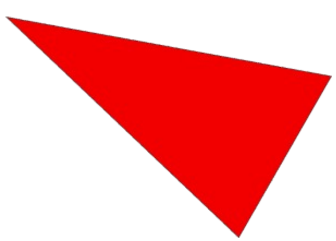


-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	5	6	7	8	-10	-10	-10	-10
-10	-10	-10	4	5	6	7	8	9	10	-10	-10
-10	-10	3	4	5	-10	-10	8	9	10	-10	-10
-10	2	-10	-10	-10	-10	8	9	10	-10	-10	-10
-10	-10	-10	-10	-10	-10	8	9	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	9	10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	9	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10

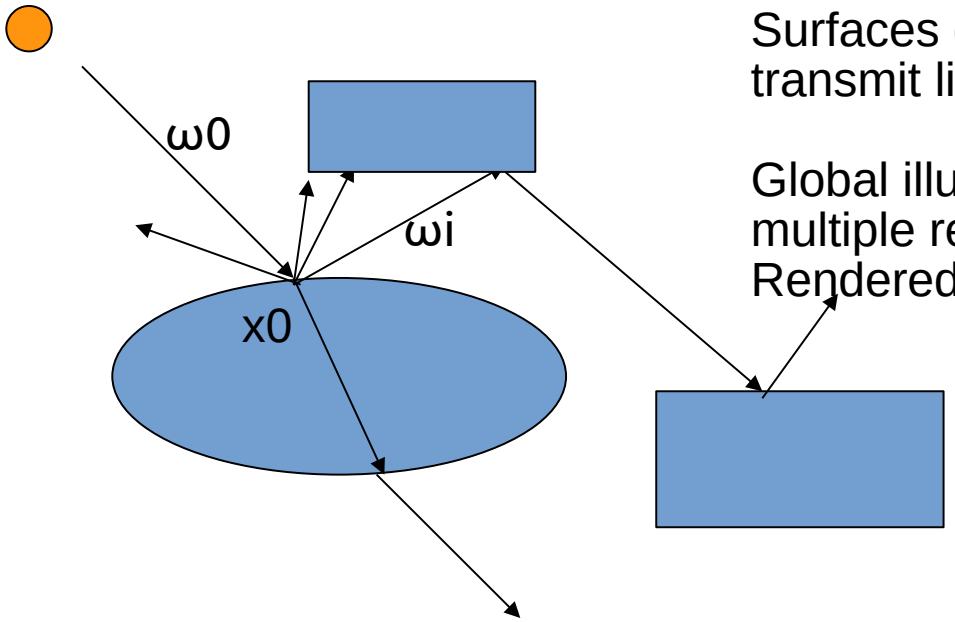




-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
-10	-10	-10	-10	5	6	7	8	-10	-10	-10	-10
-10	-10	-10	4	5	6	7	8	9	10	-10	-10
-10	2	2	3	5	-10	-10	8	9	10	-10	-10
-10	2	4	6	8	10	8	9	10	-10	-10	-10
-10	-10	-10	6	9	11	8	9	10	11	12	13
-10	-10	-10	-10			9	10	11	12	13	-10
-10	-10	-10	-10	-10	-10	9	10	11	12	12	13
-10	-10	-10	-10	-10	-10	-10	10	11	12	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	11	-10	-10	-10
-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10



Surface Shading



Surfaces can emit, absorb, reflect and transmit light.

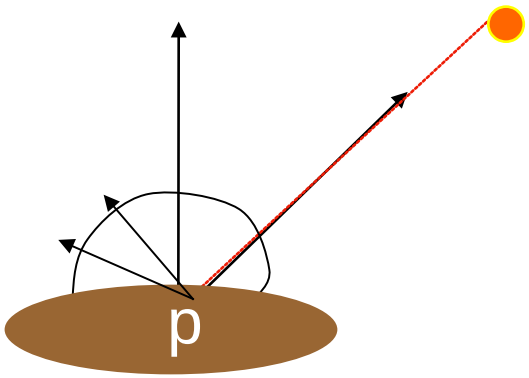
Global illumination models take into account multiple reflections and transmissions. Rendered with Ray-tracing.

λ = wave length, in practice RGB

Local illumination models (as those used in most volume rendering software) take into account only the direct contribution of light sources on surfaces, not the contribution of light bounced or transmitted by other surface. Rendered with Z-Buffer.

Surface shading

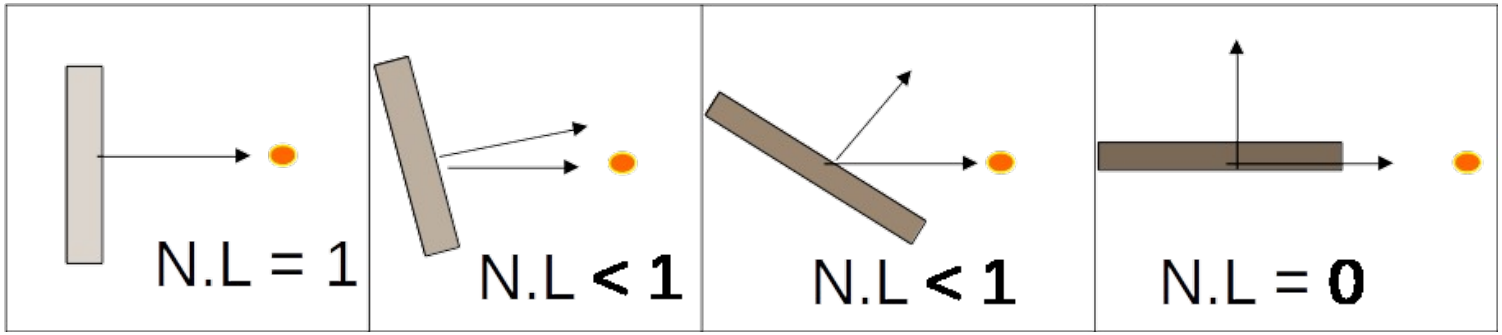
Ideally diffuse reflection. Lambert's law



The amount of light reflected at a point of an ideally diffuse surface is proportional to the cosinus of the angle formed by the normal vector of the surface at the point and the light vector

$I_d = kd.Od.\ I.\ (N \bullet L)$

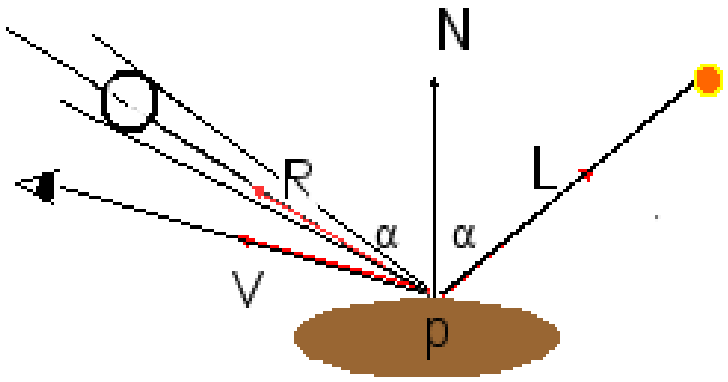
(if N and L and normalized)



Surface shading

Ideally specular reflection

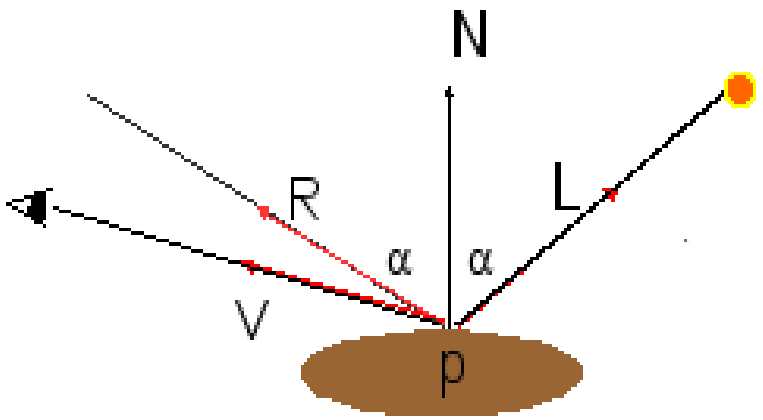
The incident light is reflected in a direction symmetric to the incident direction. The specular reflection is perceptible in a solid angle centred in R.



$$I_s = ks.Os.I.\ (R \bullet V)^n$$

Surface shading

Simple local (not global) illumination model:



$$I_{rgb}(p,\vec{v}) = Ia\,ka\,Od_{rgb} + \sum_i I_i(kd\,Od_{rgb}\,\vec{N}\cdot L_i + ks\,Os_{rgb}(\vec{R}_i\vec{V})^n)$$

For each light source i

Surface shading

Object material modeling

- Diffuse color Od (R, G, B): each channel from 0 to 1
- Diffuse reflection coefficient kd= 0...1
- Opacity alfa = 0 (transparent) .... 1 (opaque)
- Specular color Os(R, R, B): each channel from 0 to 1
- Specular reflection coefficient ks = 0...1
- Specular exponent (shininess) n =1...500

$$I_{rgb}(p,\vec{v}) = I\left(kd\,Od_{rgb}\,\vec{N}\cdot L + ks\,Os_{rgb}\left(\vec{R}\cdot\vec{V}\right)^{n}\right)$$

Surface shading

Object material modeling

- Diffuse color Od (R, G, B): each channel from 0 to 1
- Diffuse reflection coefficient kd= 0...1
- Ambient reflection coefficient ka= 0...1
- Opacity alfa = 0 (transparent) .... 1 (opaque)
- Specular color Os(R, R, B): each channel from 0 to 1
- Specular reflection coefficient ks = 0...1
- Specular exponent (shininess) n =1...500
- Ambient intensity Ia(R, G, B):: each channel from 0 to 1

Exercise 1

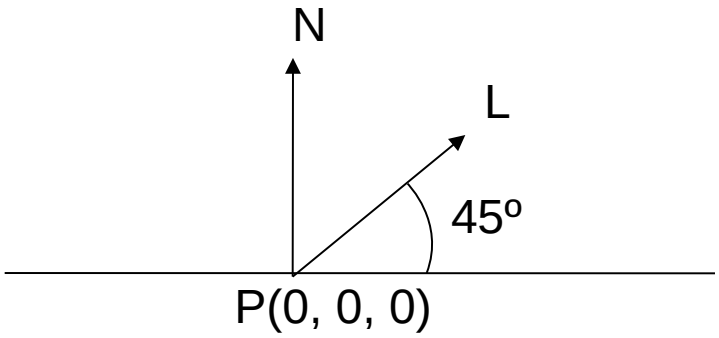
An object is perfectly diffuse, with a diffuse reflection coefficient of 0.6 and a diffuse color of (1, 1, 0). Its is lighted by a point source of pure white intensity(1, 1, 1).

Supposing that the point coordinates are P(0, 0, 0), the normal vector at P is N=(0, 1, 0) and the light vector at P is (0.707, 0.707, 0), compute the intensity of light reflected at P.

Exercise 1

An object is perfectly diffuse, with a diffuse reflection coefficient of 0.6 and a diffuse color of (1, 1, 0). Its is lighted by a point source of pure white intensity(1, 1, 1).

Supposing that the point coordinates are P(0, 0, 0), the normal vector at P is N=(0, 1, 0) and the light vector at P is (0.707, 0.707, 0), compute the intensity of light reflected at P.



kd Od I N.L

$$Id(P) = 0.6 * \begin{bmatrix}1\\1\\0\end{bmatrix} * \begin{bmatrix}1\\1\\1\end{bmatrix} * 0.707$$

$$= \begin{bmatrix}0.42\\0.42\\0\end{bmatrix}$$

Less of an half of the incident light and only the yellow light

We can make a drawing in the (x, y) plane, because Nz=Lz=0

N.L = 0*0.707 + 1*0.707 + 0 = 0.707

Exercise 2

An object is perfectly diffuse, with a diffuse reflection coefficient of 0.6 and a diffuse color of (1, 1, 0). Its is lighted by a point source of pure blue intensity(0, 0, 1).

Supposing that the point coordinates are P(0, 0, 0), the normal vector at P is N=(0, 1, 0) and the light vector at P is (0.707, 0.707, 0), compute the intensity of light reflected at P.

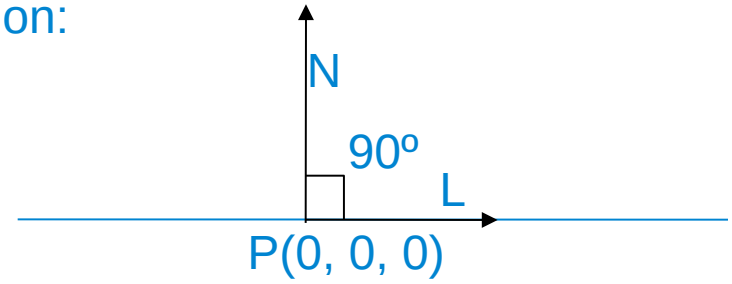
Solution: No need to compute anything: since the diffuse color of the object is yellow, it can reflect only yellow light, but the incident light is blue. Therefore: Id(P) = (0, 0, 0)

Exercise 3

An object is perfectly diffuse, with a diffuse reflection coefficient of 0.6 and a diffuse color of (1, 1, 0). Its is lighted by a point source of pure white intensity(1, 1, 1).

Supposing that the point coordinates are P(0, 0, 0), the normal vector at P is N=(0, 1, 0) and the light vector at P is (1, 0, 0), compute the intensity of light reflected at P.

Solution:



N.L = 0 because their angle is 90º, thus:

Id(P) = (0, 0, 0)

Exercice 4

An object has a diffuse reflection coefficient of 0 and a diffuse color of (1, 1, 0), a specular color (1, 1, 1) a specular reflection coefficient 0.8 and shininess exponent 100. It is lighted by a point source of pure white intensity(1, 1, 1).

Supposing that the point coordinates are P(0, 0, 0), the normal vector at P is N=(0, 1, 0), the light vector at P is (0.707, 0.707, 0), the viewing vector (0.707, 0.707, 0) compute the intensity of light reflected at P.

Exercise 4

An object has a diffuse reflection coefficient of 0 and a diffuse color of (1, 1, 0), a specular color (1, 1, 1) a specular reflection coefficient 0.8 and shininess exponent 100. It is lighted by a point source of pure white intensity(1, 1, 1).

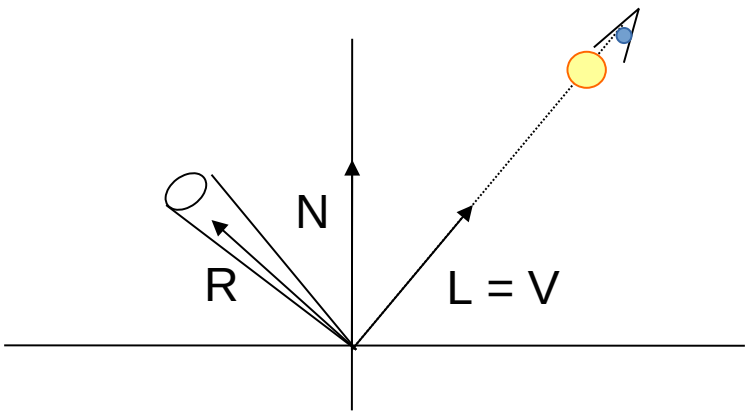
Supposing that the point coordinates are P(0, 0, 0), the normal vector at P is N=(0, 1, 0), the light vector at P is (0.707, 0.707, 0), the viewing vector (0.707, 0.707, 0) compute the intensity of light reflected at P.

Answer:

Because the object is purely specular, we only need to take care of the specular reflection. The viewing vector and the light vector have the same direction, so the viewer is in the opposite direction of the reflection. Thus IP= (0, 0, 0).

Let's check it mathematically:

The reflected vector R is (-0.707, 0.707, 0), thus R.V = -0.707*0.707 + 0.707*0.707 + 0.0 = 0

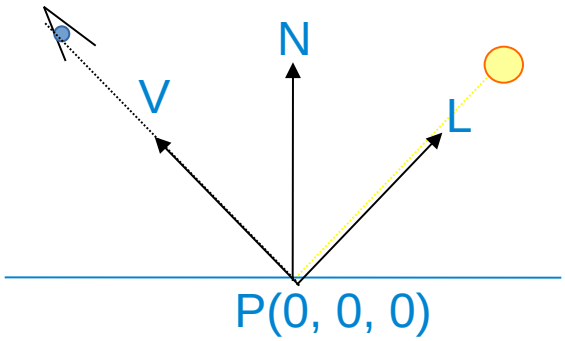


Exercice 5

An object has a diffuse reflection coefficient of 0.3 and a diffuse color of (1, 1, 0), a specular color (1, 1, 1) a specular reflection coefficient 0.5 and shininess exponent 100. It is lighted by a point source of pure white intensity(1, 1, 1).

Supposing that the point coordinates are P(0, 0, 0), the normal vector at P is N=(0, 1, 0), the light vector at P is (0.707, 0.707, 0), the viewing vector (-0.707, 0.707, 0) compute the intensity of light reflected at P.

Solution:



N.L = 0.707 and Id(P) = (0.212, 0.212, 0.0)

R==V, thus R.V = 1 because V is the symmetric vector of V in relation to N, thus Is(P) = (0.5, 0.5, 0.5)

Thus, I(P) = (0.712, 0.712, 0.5)

3D Slicer

http://
www.slicer.org/

Sample data:
http://www.slicer.org/slicerWiki/index.php/SampleData

Slicer 3D

Free, open source software package for visualization and image analysis. 3D Slicer is natively designed to be available on multiple platforms, including Windows, Linux and Mac Os X.

3D Slicer as an Image Computing Platform for the Quantitative Imaging Network. Fedorov A., Beichel R., Kalpathy-Cramer J., Finet J., Fillion-Robin J-C., Pujol S., Bauer C., Jennings D., Fennessy F., Sonka M., Buatti J., Aylward S.R., Miller J.V., Pieper S., Kikinis R. Magn Reson Imaging. 2012 Nov;30(9):1323-41.

Slicer 3D

Built on an open source software infrastructure based on the NA-MIC Kit

- The Visualization Toolkit (VTK http://www.vtk.org/): 3D rendering
- The Insight Toolkit (ITK http://www.itk.org/): image processing
- Qt (http://qt-project.org) user interface
- Common Toolkit (CTK http://www.commontk.org) : biomedical tools (DICOM)
- Teem(http://teem.sourceforge.net/) libraries for raster data processing (nrrd)

Download and installation

http://download.slicer.org/

Get Slicer 4.

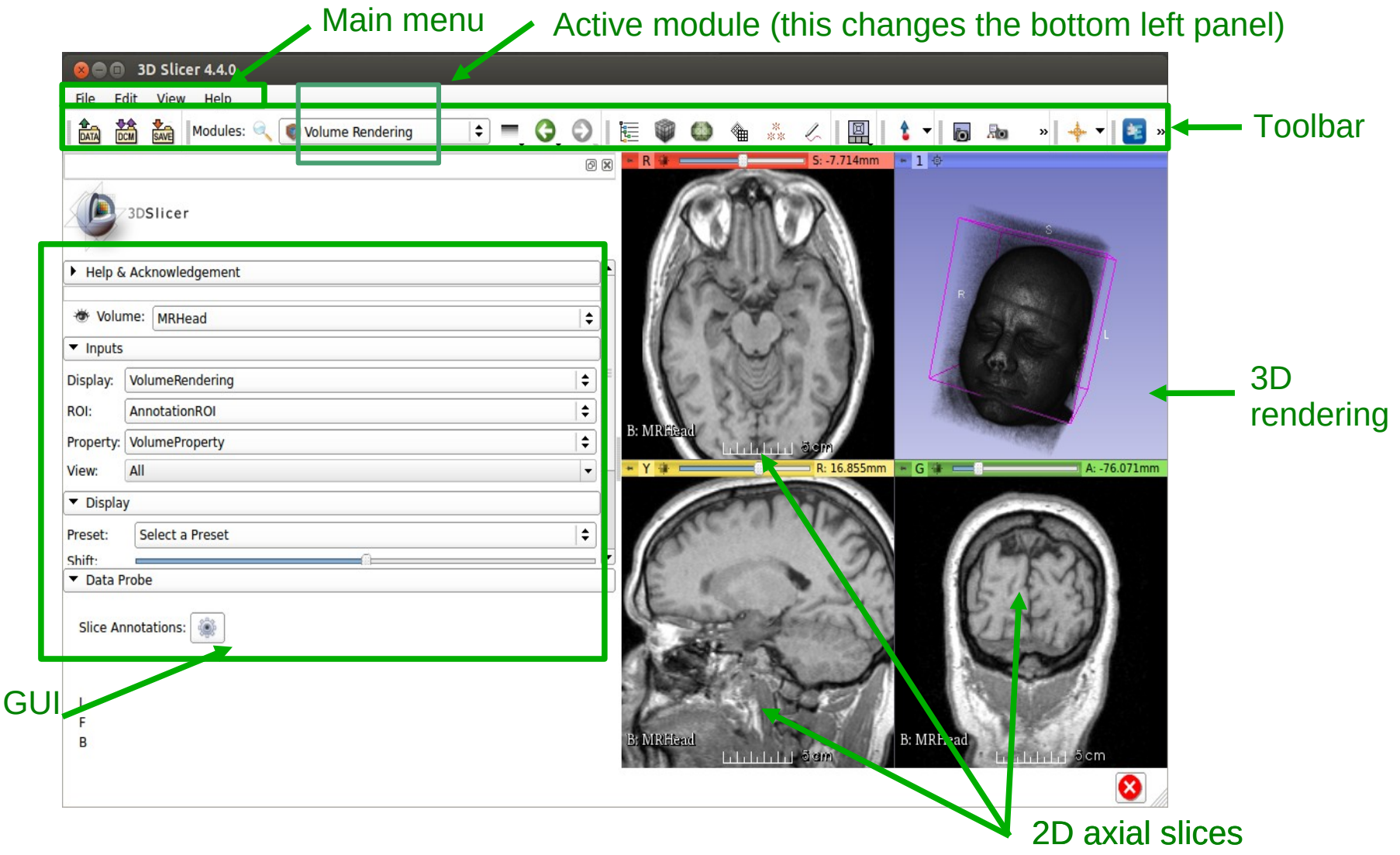
Slicer 4 is the latest version of 3D Slicer, a free, comprehensive software platform for medical image analysis and visualization developed with NIH support.

3D Slicer is distributed under a permissive BSD-style open source license. It has a thriving user and developer community.

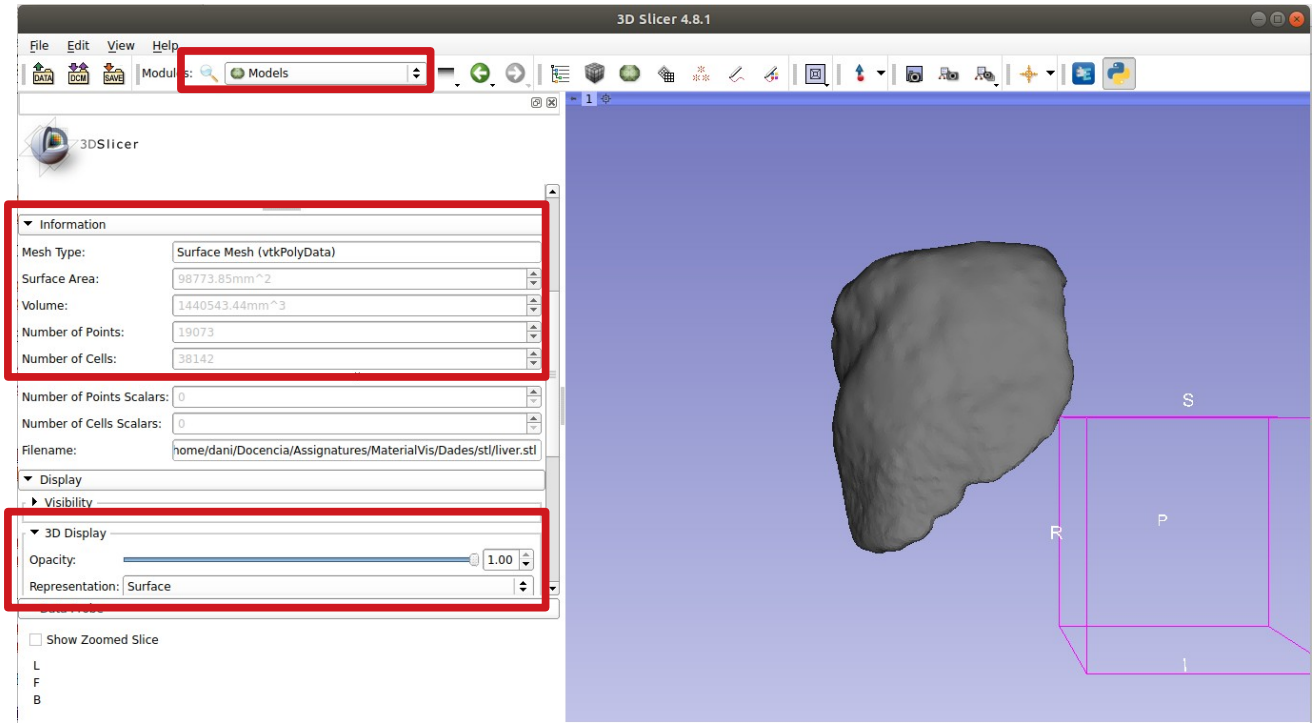
Pre-compiled binaries

		Windows	Mac OS X	Linux
stable release	64 bit	4.4.0 64 bit installer 2014-11-02 r23774 (164.2MB)	4.4.0 64 bit installer 2014-11-02 r23774 (224.0MB)	4.4.0 64 bit archive 2014-11-02 r23774 (244.3MB)
	32 bit	4.3.0 32 bit installer 2013-09-06 r22408 (157.2MB)		
nightly build	64 bit	nightly 64 bit installer 2014-11-07 r23781 (164.2MB)	nightly 64 bit installer 2014-11-07 r23781 (224.0MB)	nightly 64 bit archive 2014-11-07 r23781 (244.3MB)
	32 bit	nightly 32 bit installer 2013-11-24 r22717 (159.0MB)		

Interface

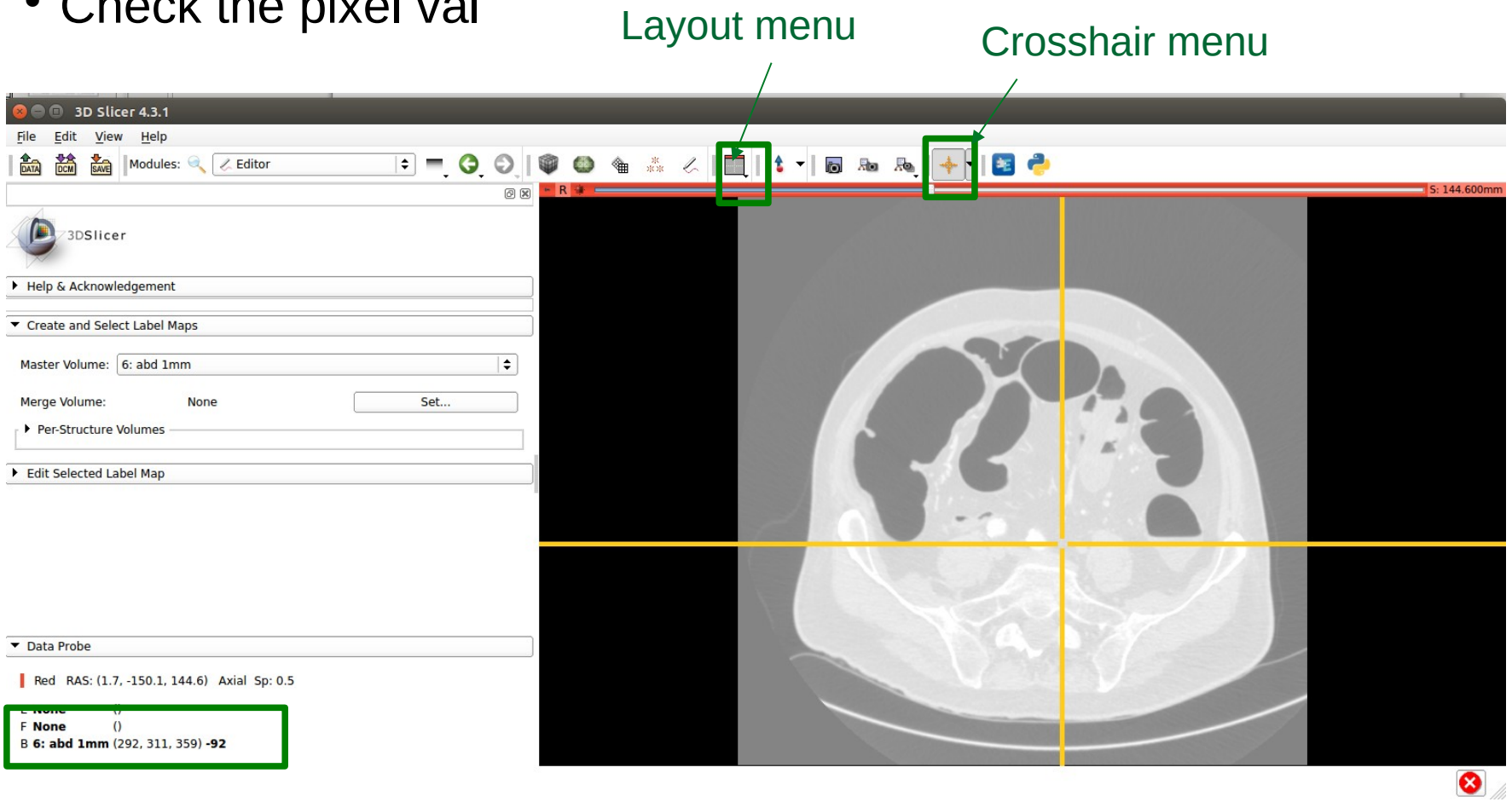

Main menu
Active module (this changes the bottom left panel)
3D Slicer 4.4.0
File Edit View Help
Modules: Volume Rendering
Toolbar
3DSlicer
Help & Acknowledgement
Volume: MRHead
Inputs
Display: VolumeRendering
ROI: AnnotationROI
Property: VolumeProperty
View: All
Display
Preset: Select a Preset
Shift:
Data Probe
Slice Annotations:
GUI
3D rendering
2D axial slices

Practical work
Read an stl model

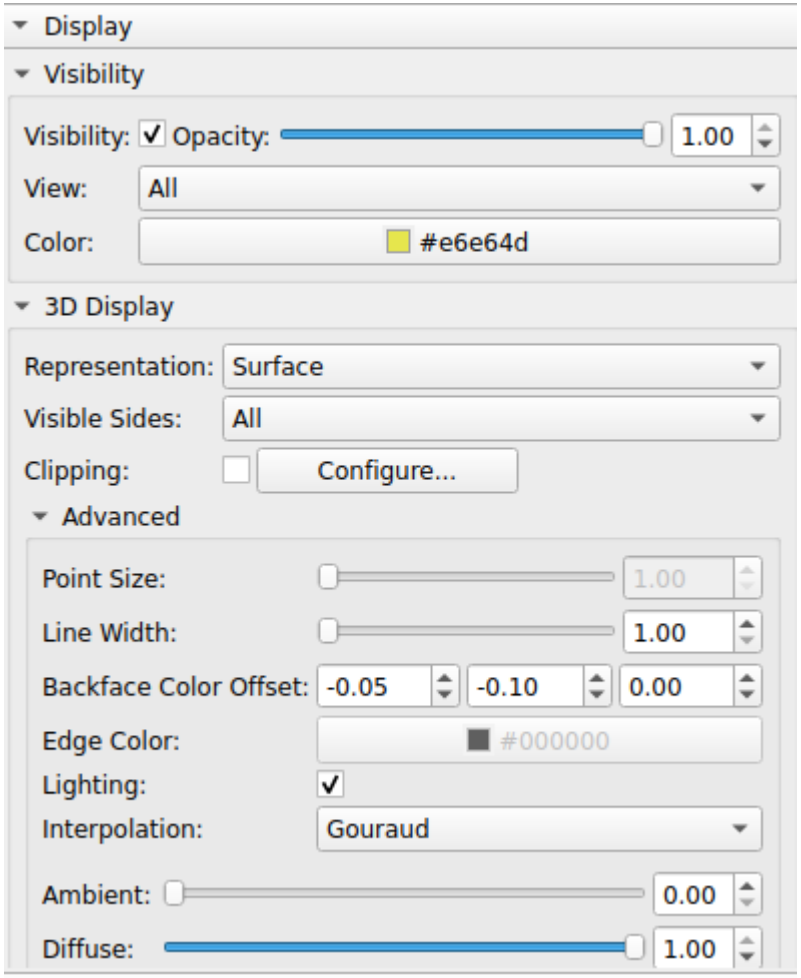


Practical work: navigation

- Enable the crosshair
- Move through the image
- Check the pixel val



Practice
surface shading



Play with the optical properties in the **Model** panel

Use the Surface Toolbox