VTK (Visualization Toolkit)

2017 하계 CDE학회 튜토리얼

2017.08.18

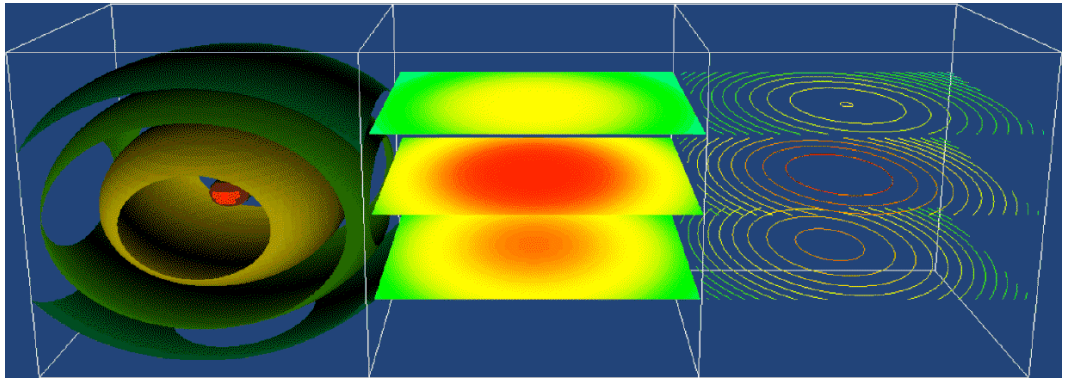
한국과학기술연구원

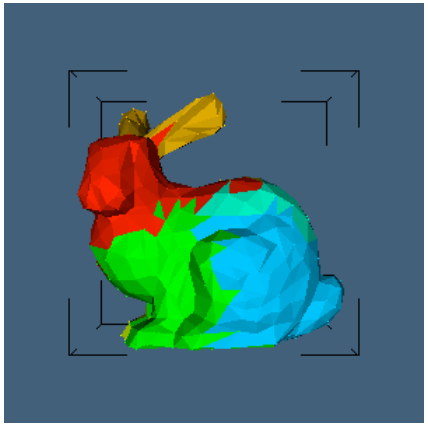
김 영 준

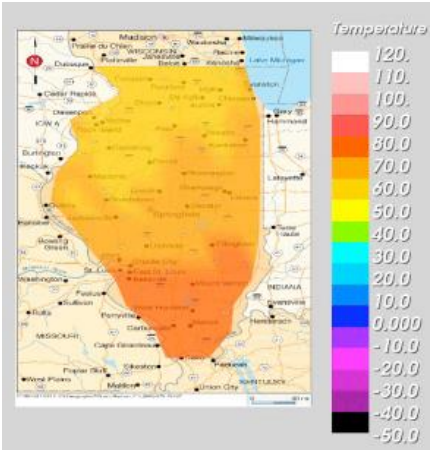
VTK for Scientific Visualization

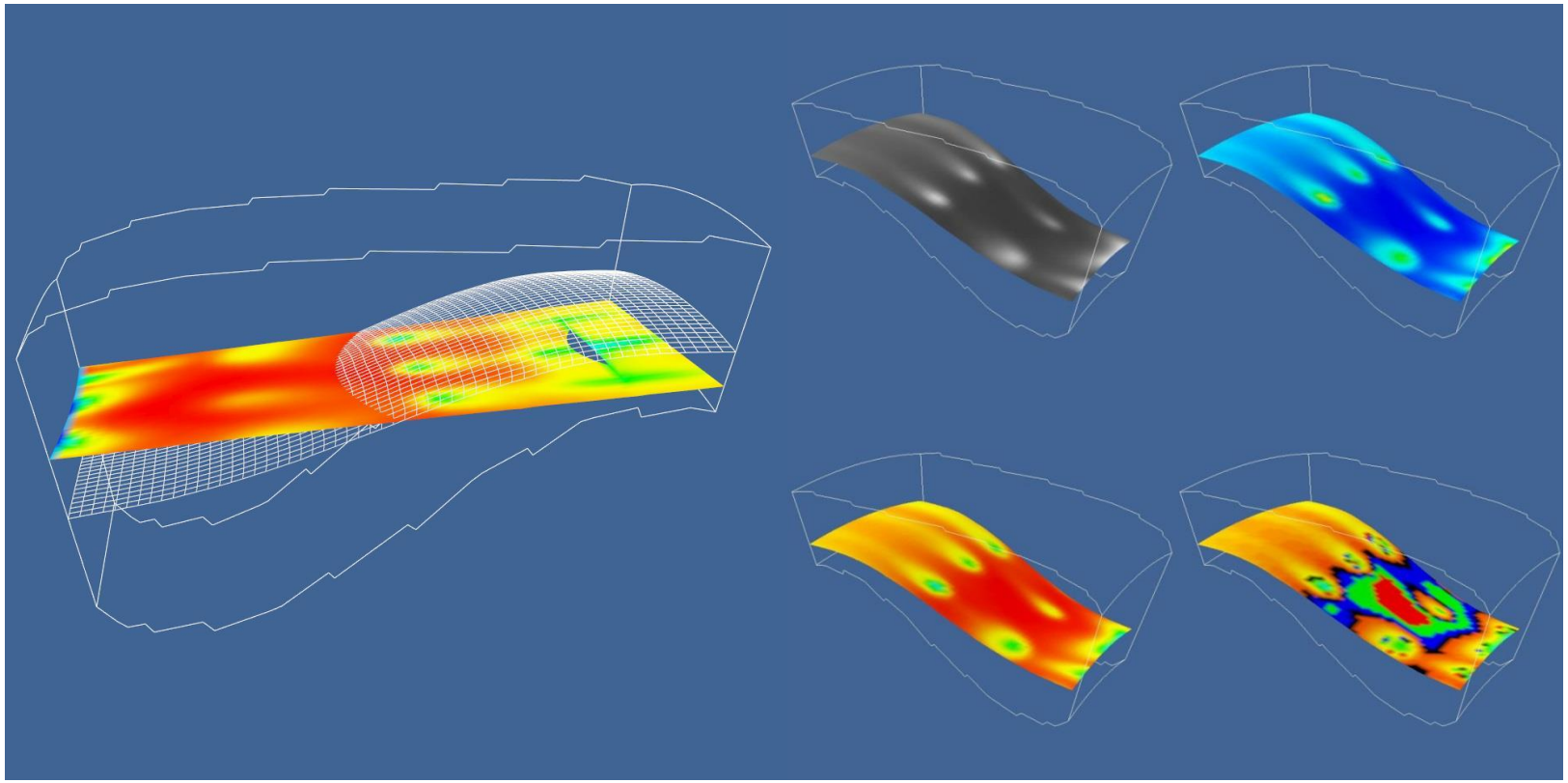
INTRODUCTION

What can VTK do?

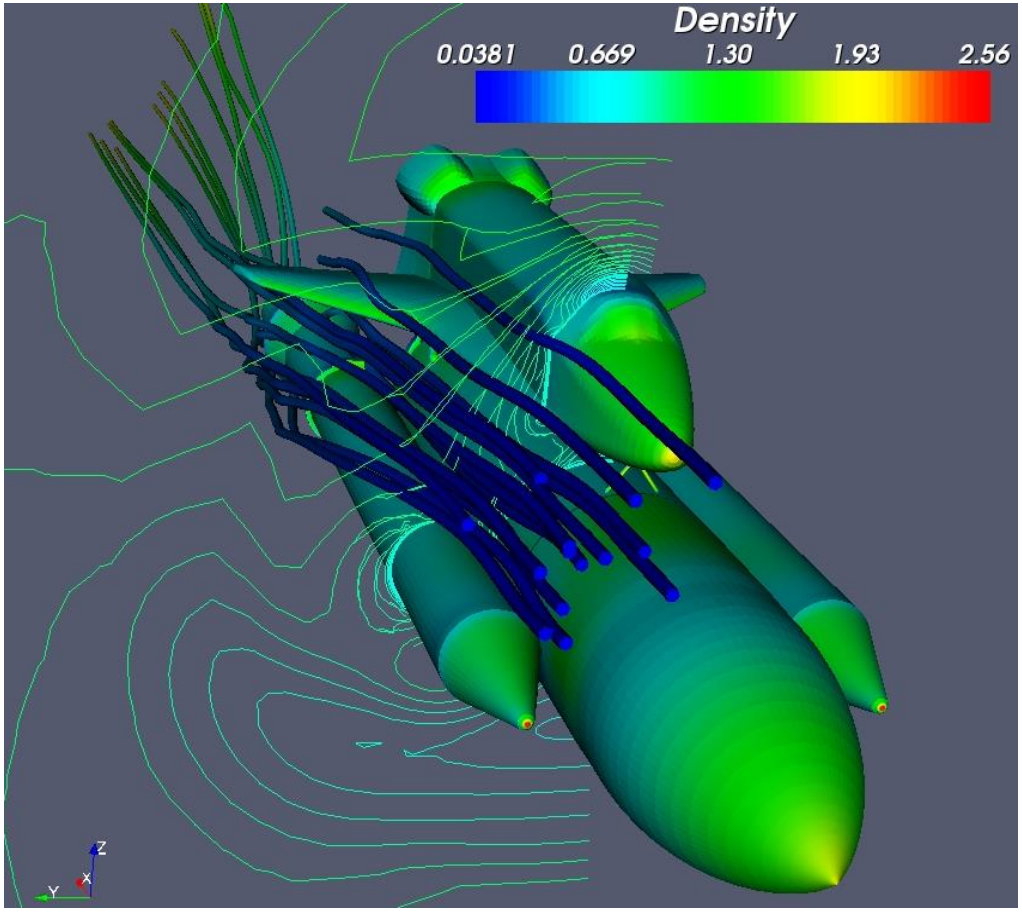




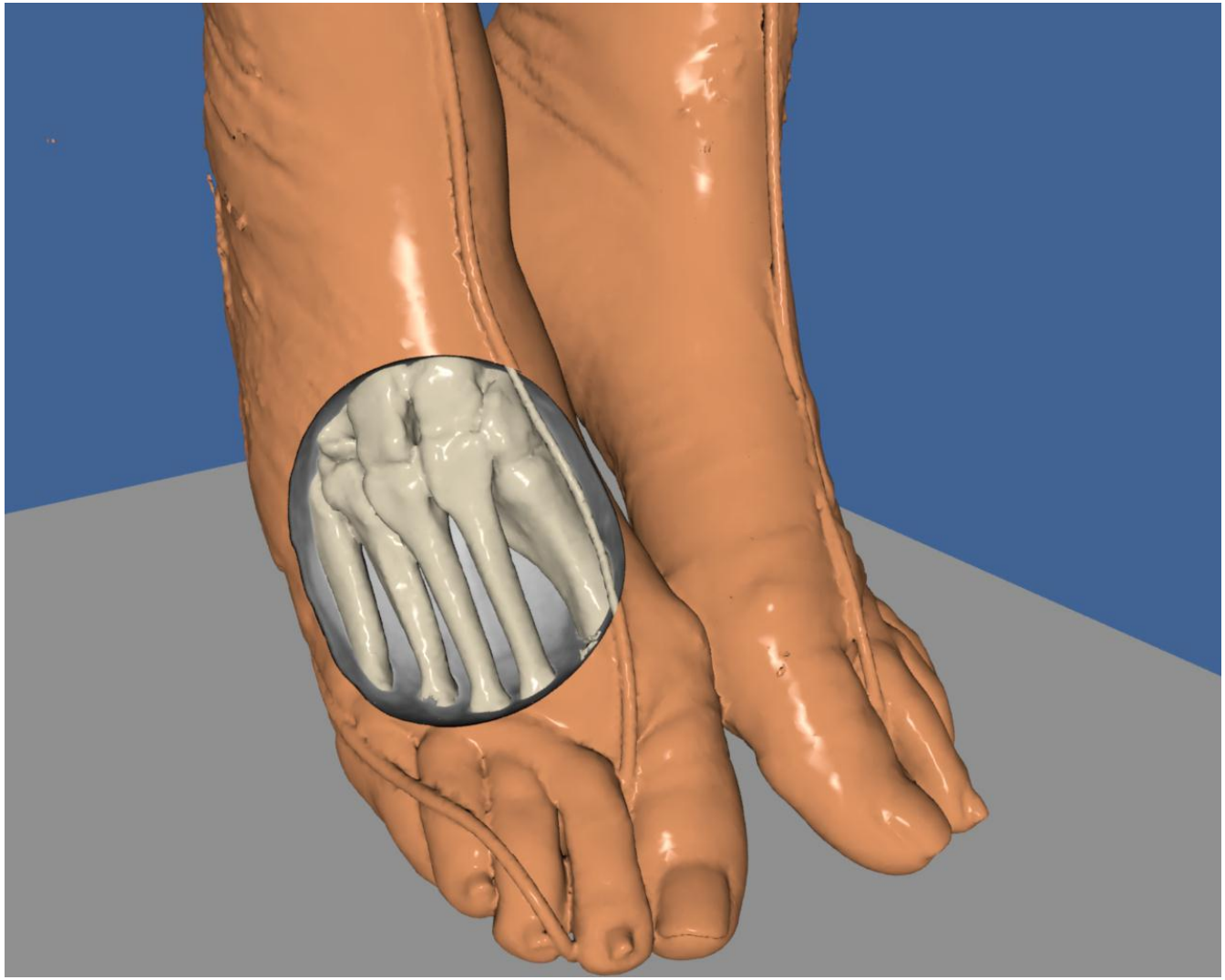




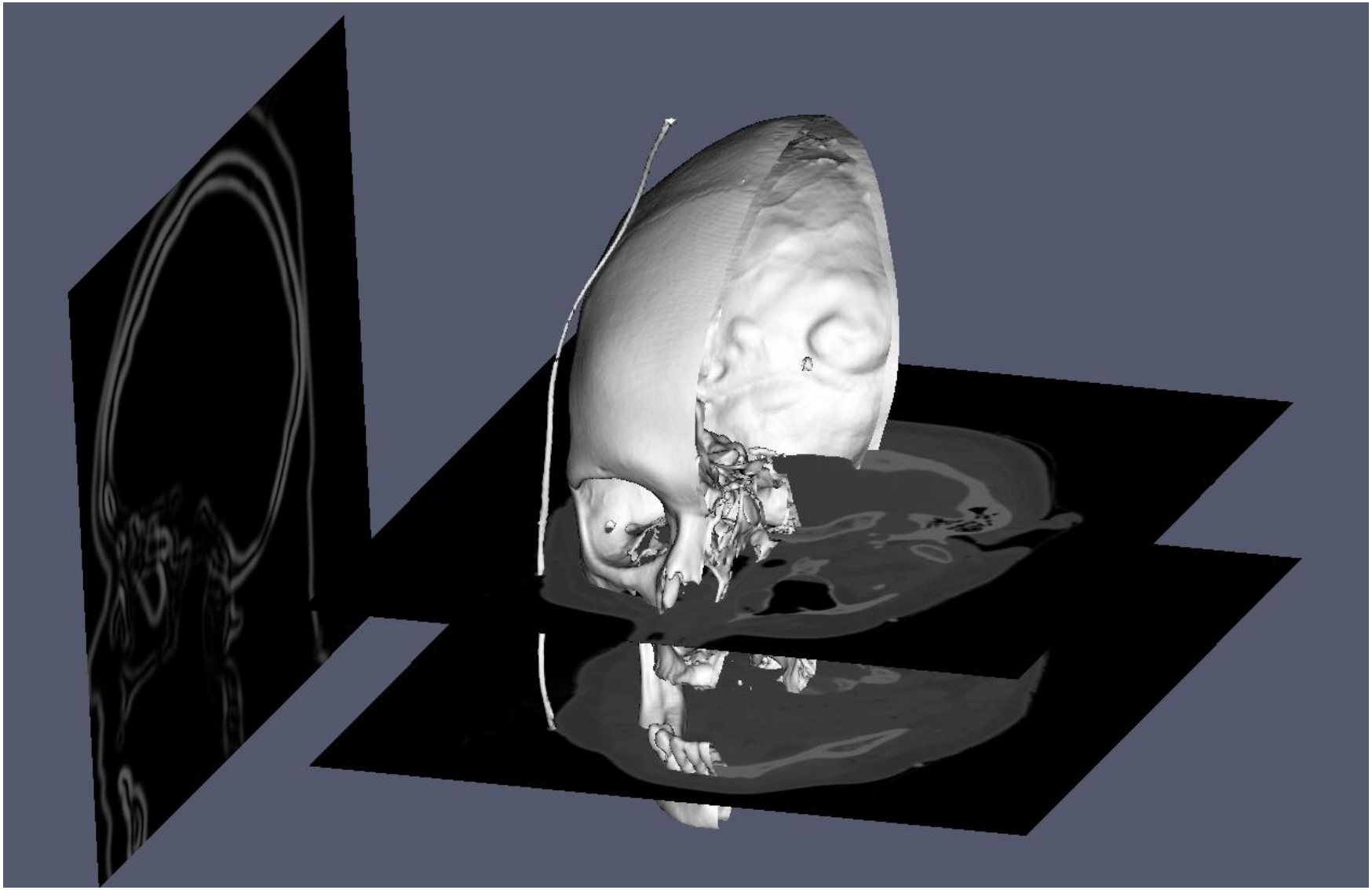
가스터빈 연소실(combustor) 내의 연소과정의 3차원 가시화. Lookup table에 따라 다양한 가시화 가능



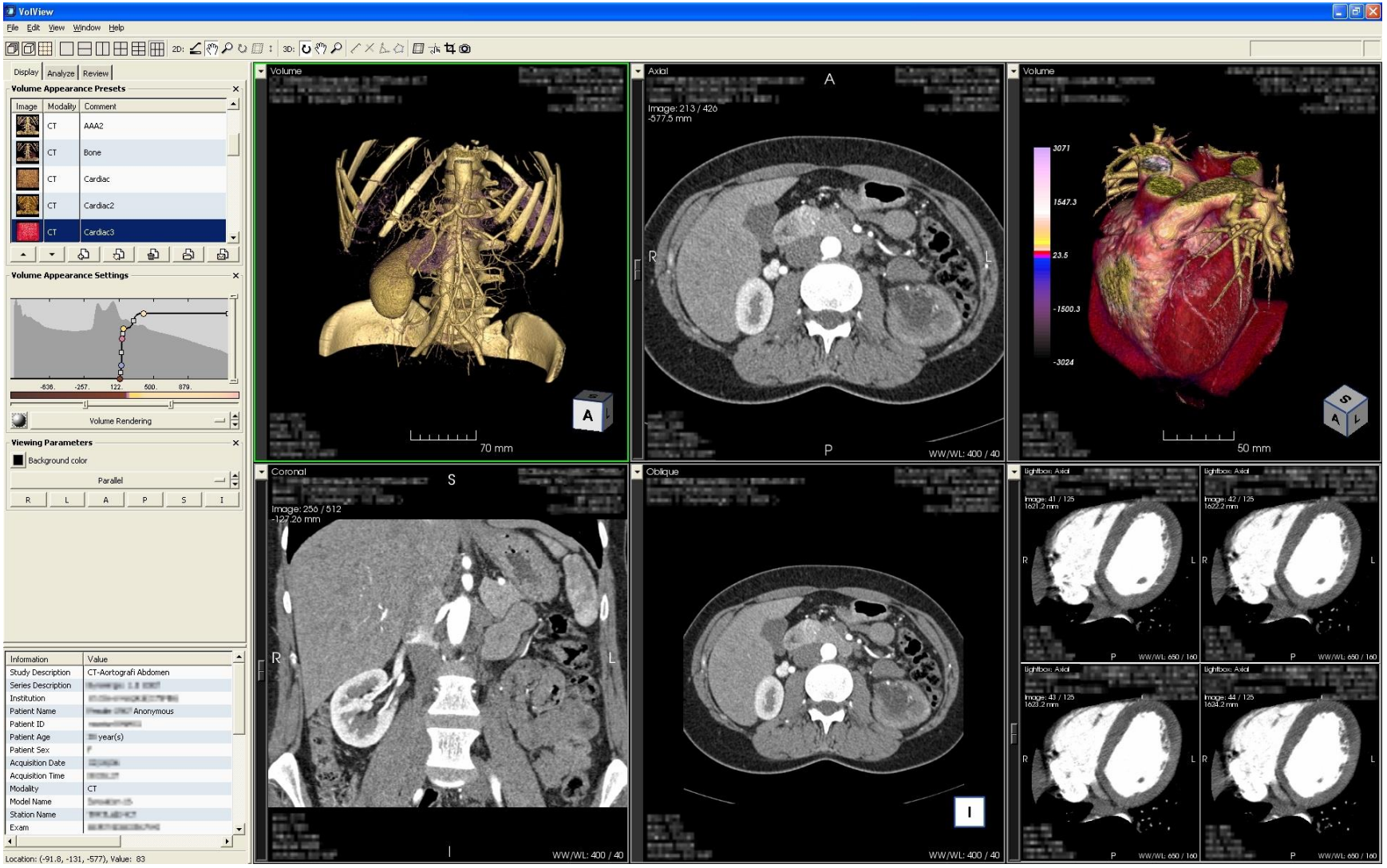
스페이스 셔틀 주위의 유동(fluid flow) 가시화 (컬러맵으로 표현된 색상은 해당 지점에서의 flow density를 표현)



CT scan from the visible woman dataset. An isosurface of the skin is clipped with a sphere to reveal the underlying bone structure.
Author: Original visualization author Bill Lorensen

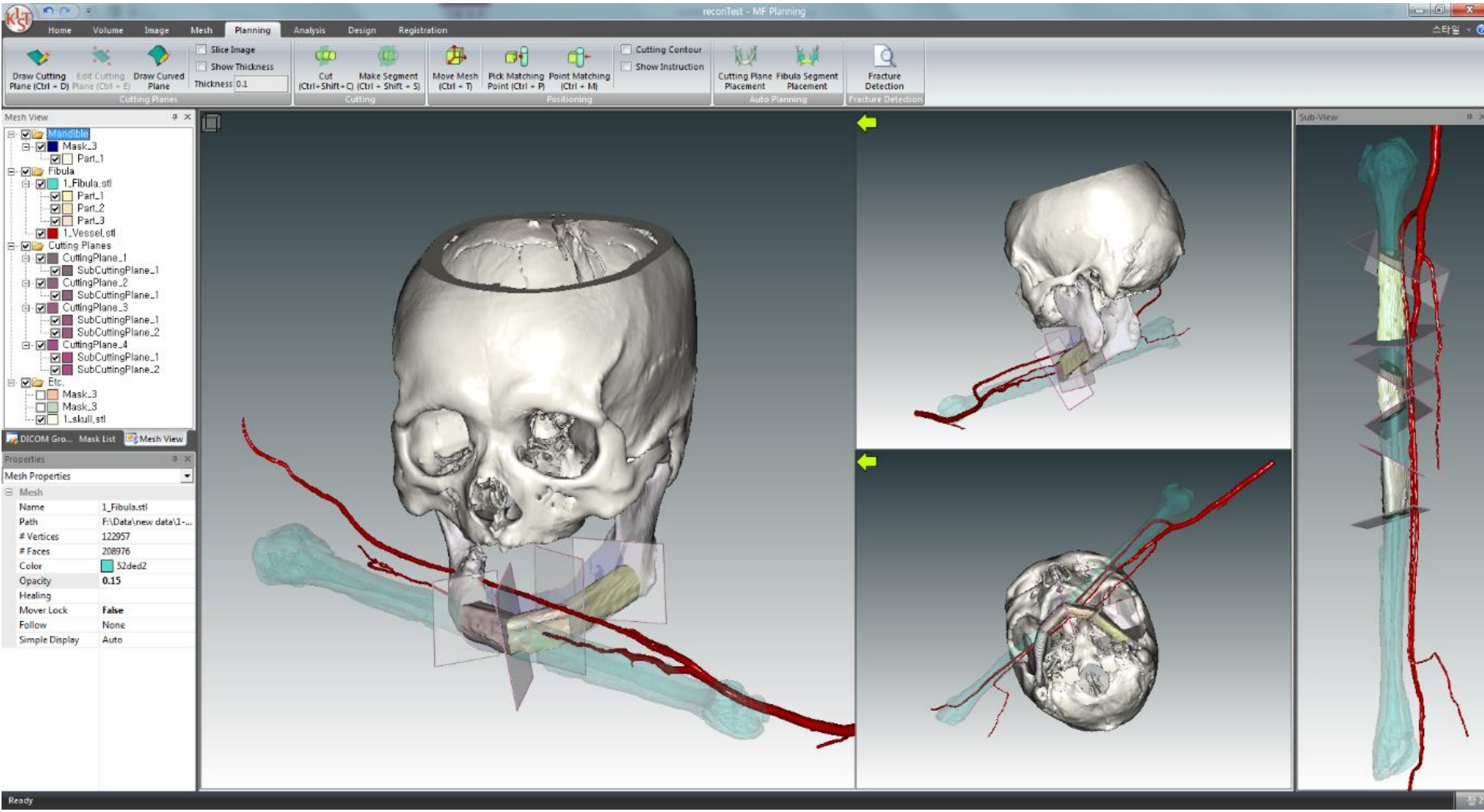


Volume rendering and image display from the visible woman dataset.
Author: Kitware, Inc.

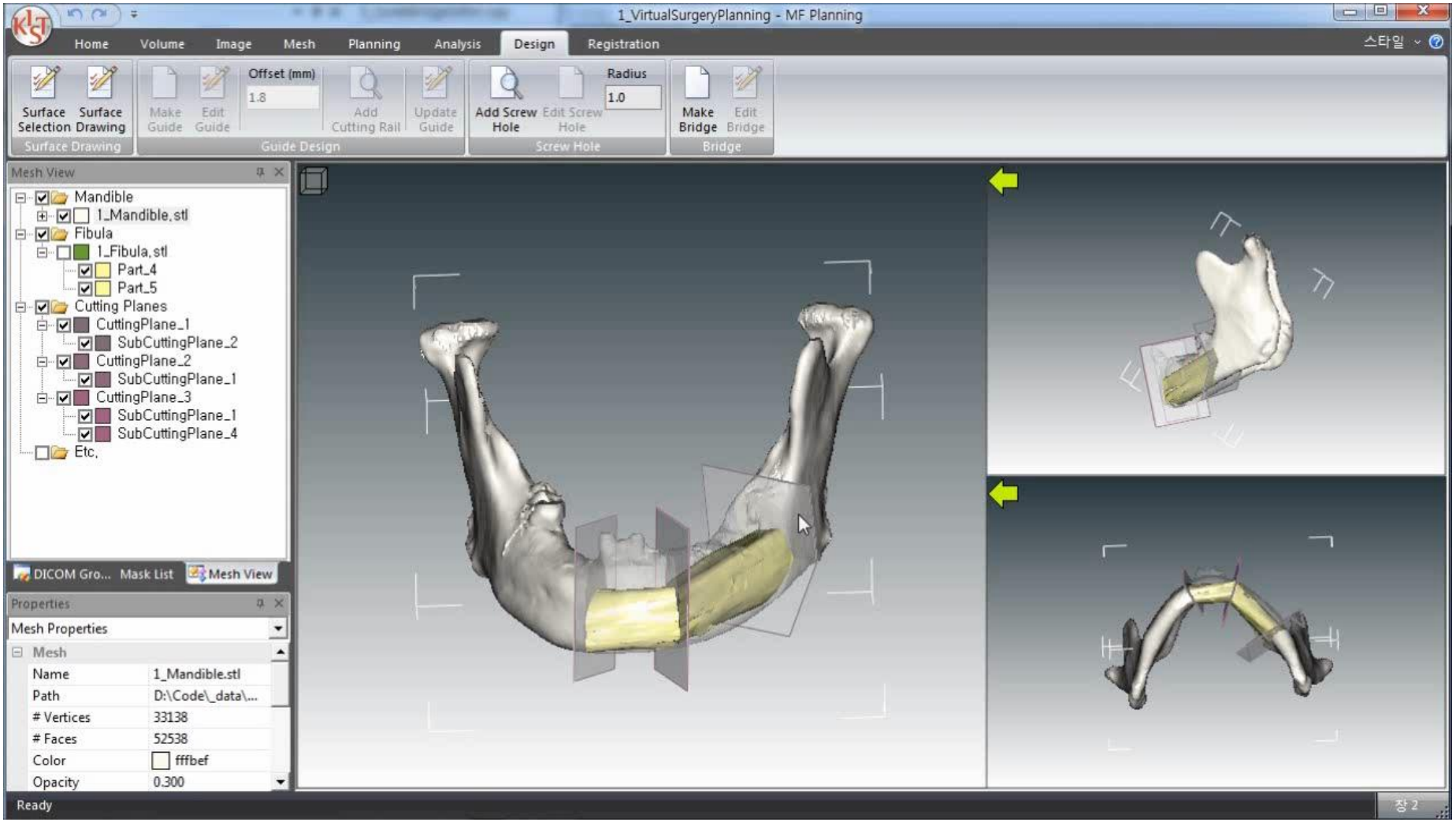


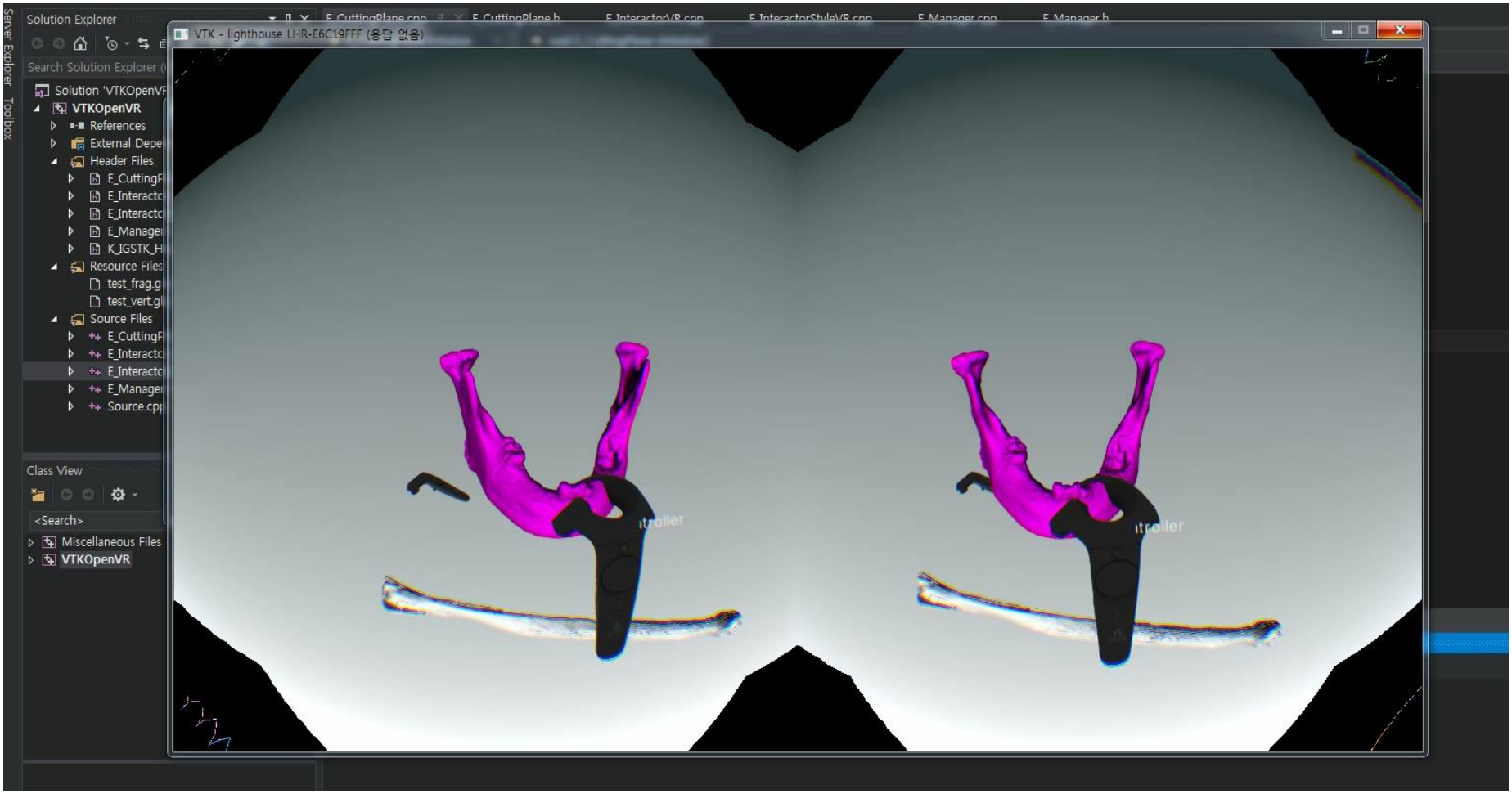
Volume rendering and CT display of a human torso, with emphasis on the kidney
Author: Kitware, Inc.

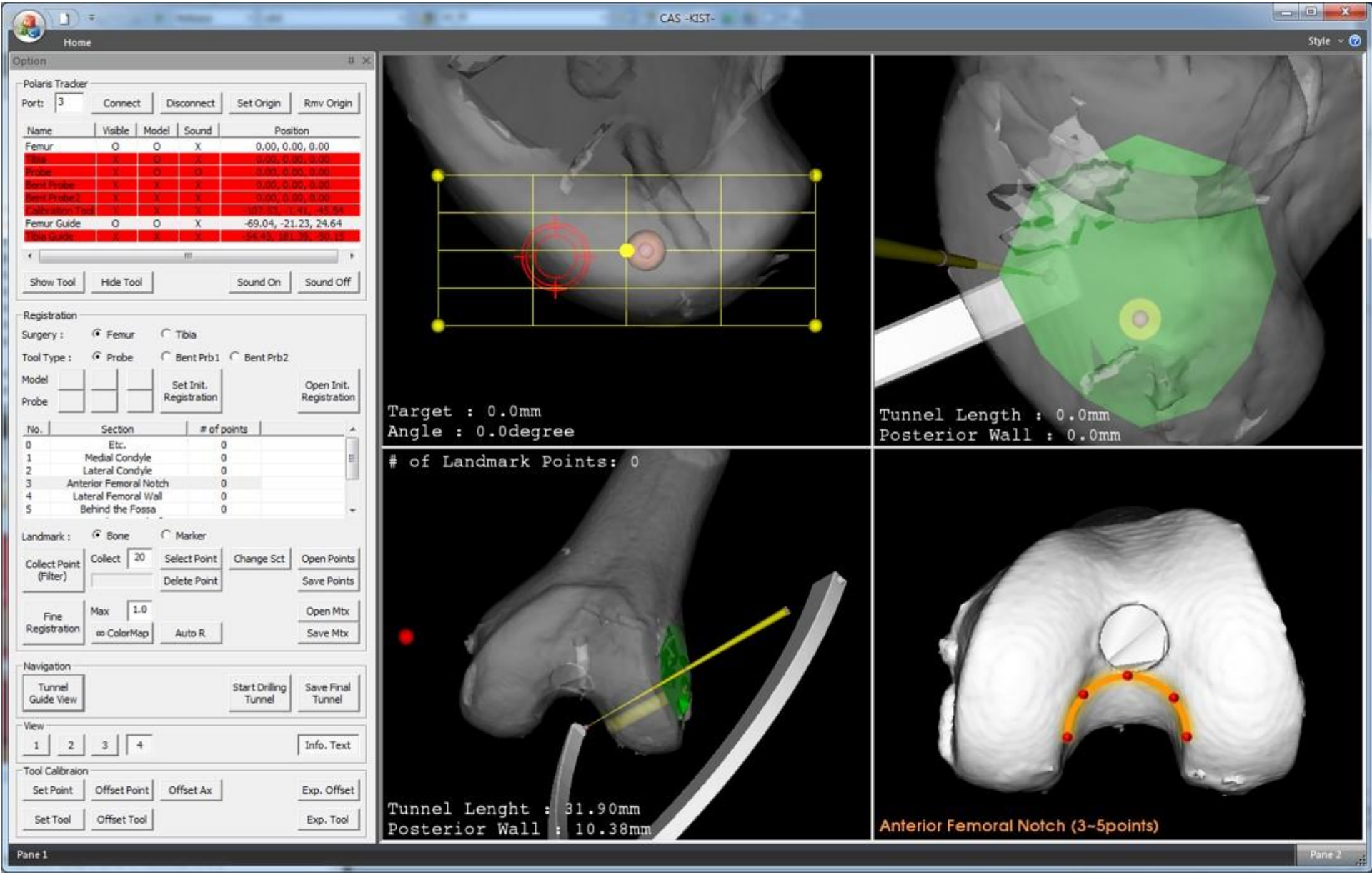
3D Surgical Planning



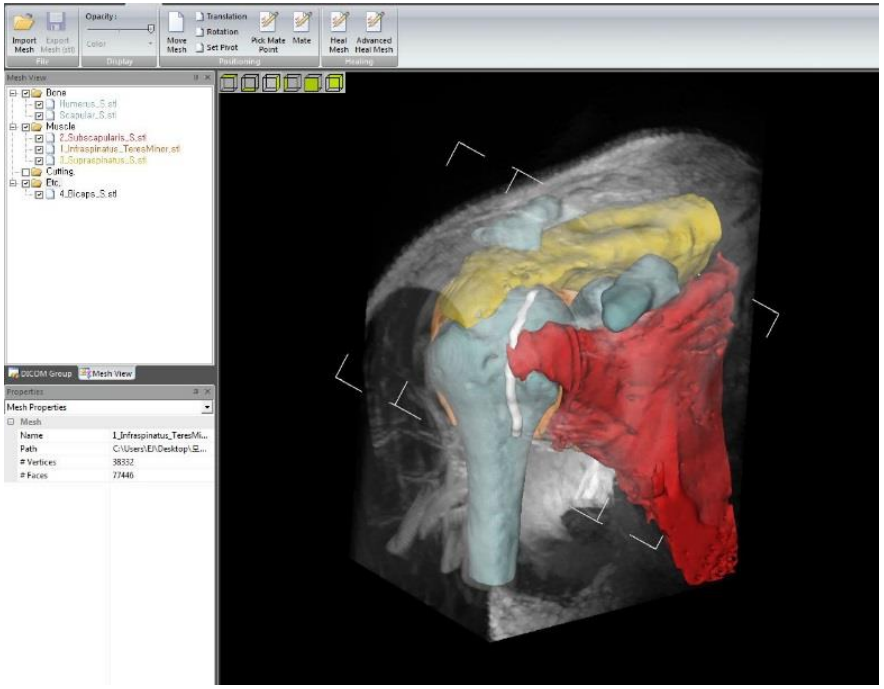
Modeling of Surgical Resection Guide



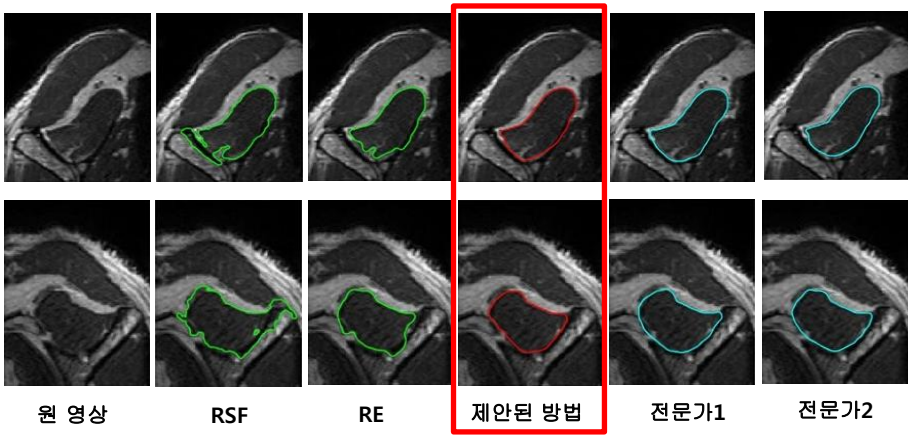

Solution Explorer
VTK - lighthouse LHR-E6C19FFF (응답 없음)
Class View
VTKOpenVR



- 삼성서울병원 정형외과 왕준호 교수님 공동연구



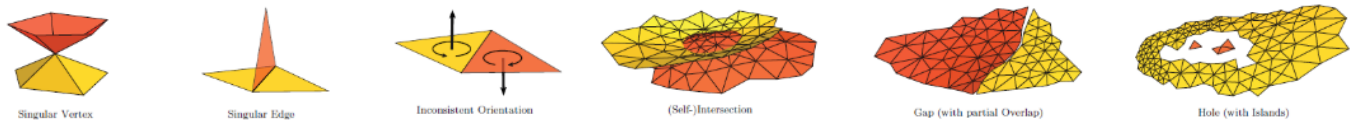
- **2차원 영상간의 비교**



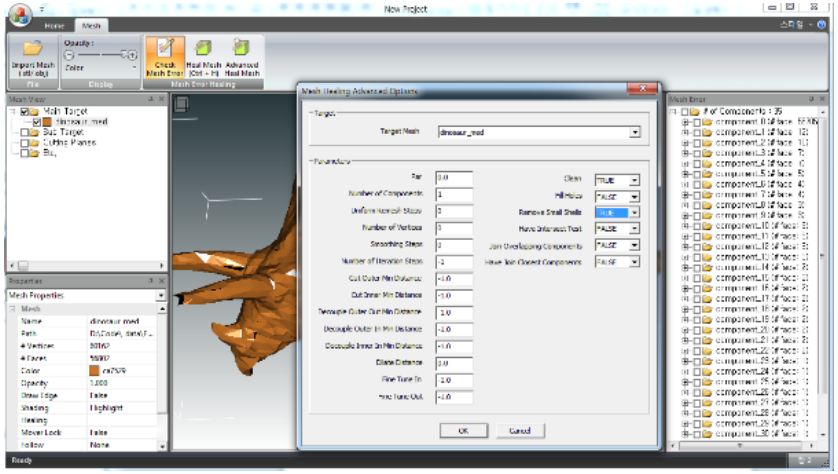
- 건국대학교 정형외과 정석원 교수님 공동연구

S.W. Chung, ..., Y. Kim*, "Serial Changes in 3-Dimensional Supraspinatus Muscle Volume following Rotator Cuff Repair", *The American Journal of Sports Medicine*, E-pub, Jun. 2017 (I.F. = 5.673)

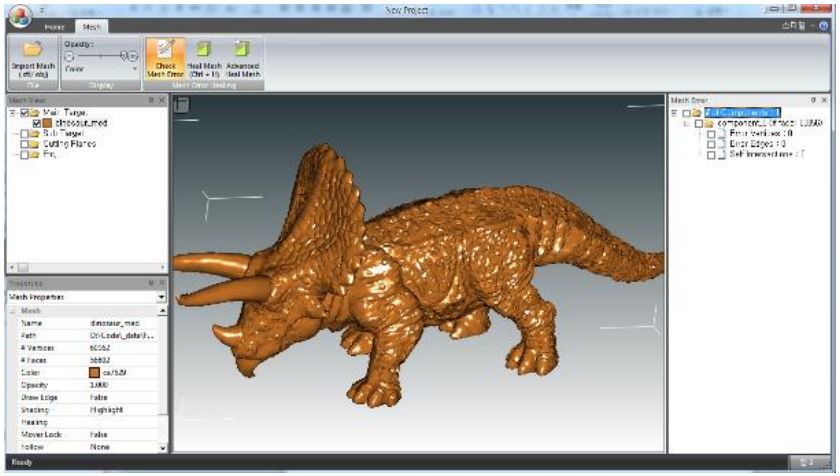
S. Kim, D. Lee, S. Park, K. Oh, S.W. Chung*, Y. Kim*, "Automatic segmentation of supraspinatus from MRI by internal shape fitting and autocorrection", *Computer Methods and Programs in Biomedicine*, Vol. 140, pp. 165-174, Mar. 2017 (I.F. = 2.503)



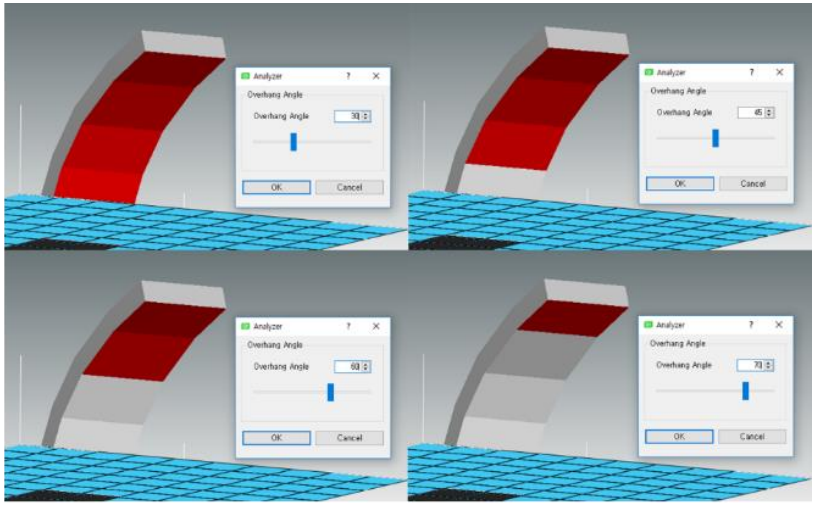
다양한 mesh error 타입



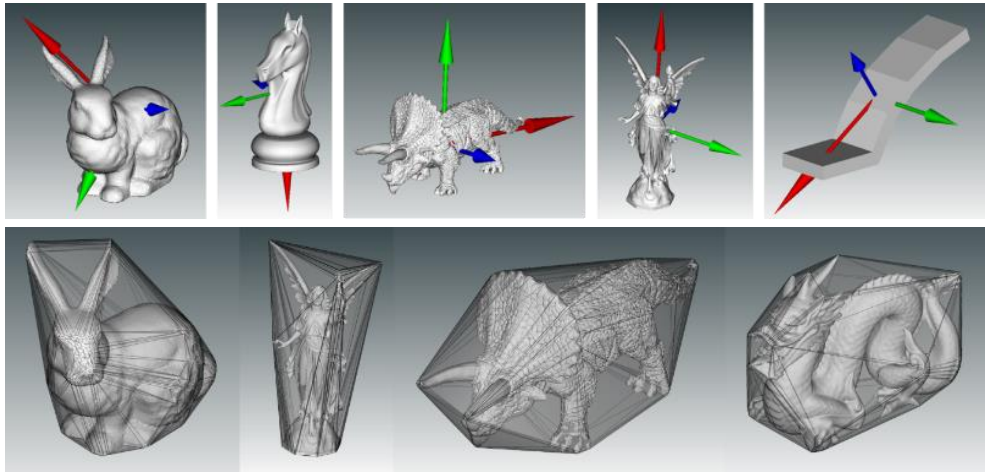
Mesh error 분석 및 healing 옵션



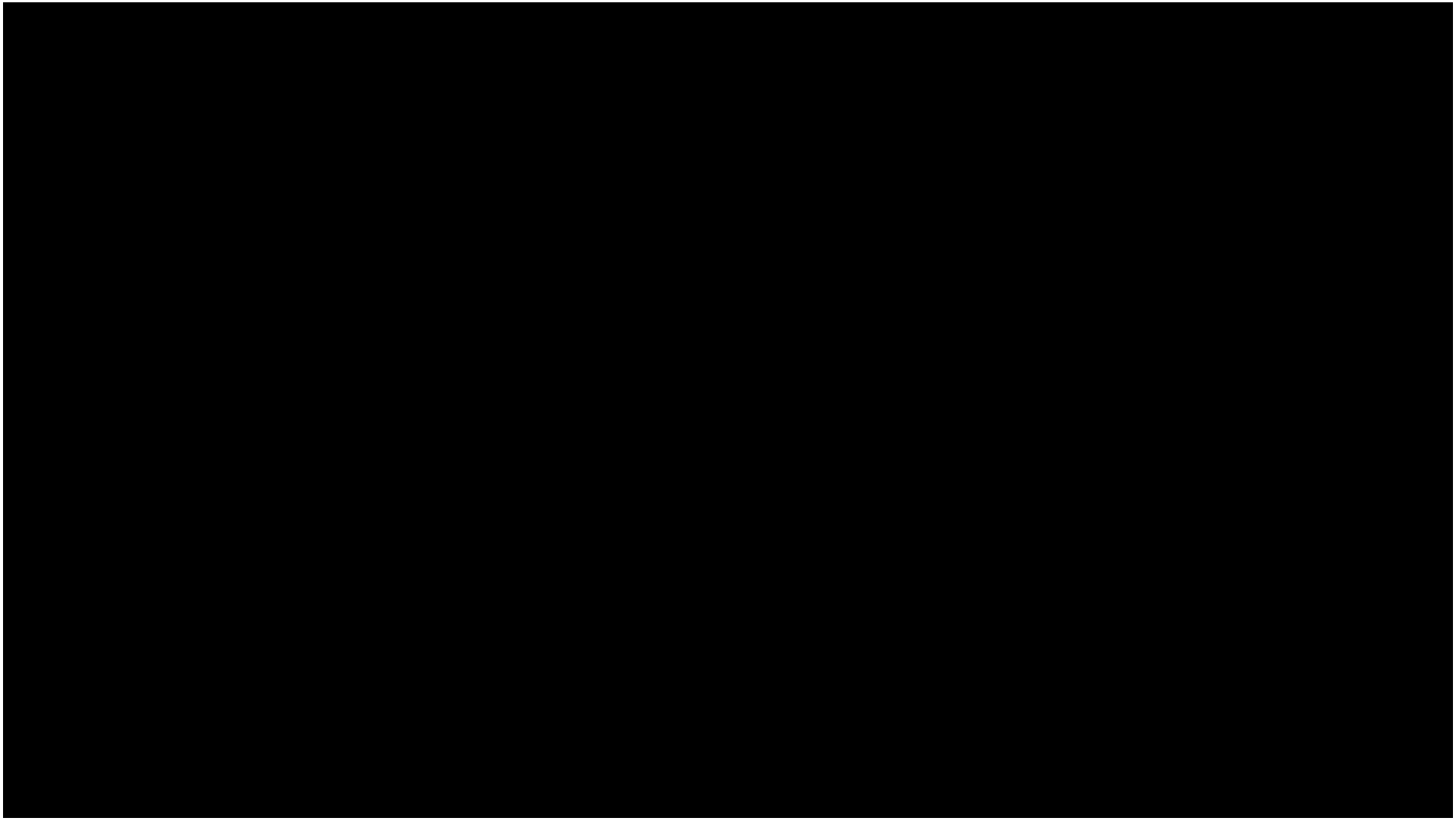
Mesh healing 결과



Overhang 각도에 따른 support 생성 영역의 변화



다양한 형상에 대한 PCA 분석 및 Convex hull 구현 결과



VTK (Visualization toolkit)

- Visualization ToolKit
  - Scientific visualization
  - Information visualization
  - Written in C++
  - C++, Java, Python, Tcl API's
- Much more than that:
  - Excellent framework for scientific programming

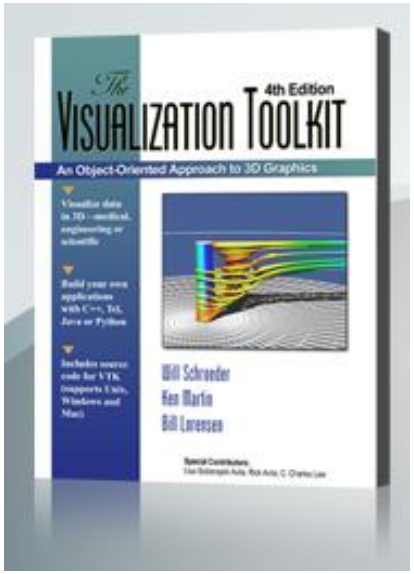
VTK (Visualization toolkit)

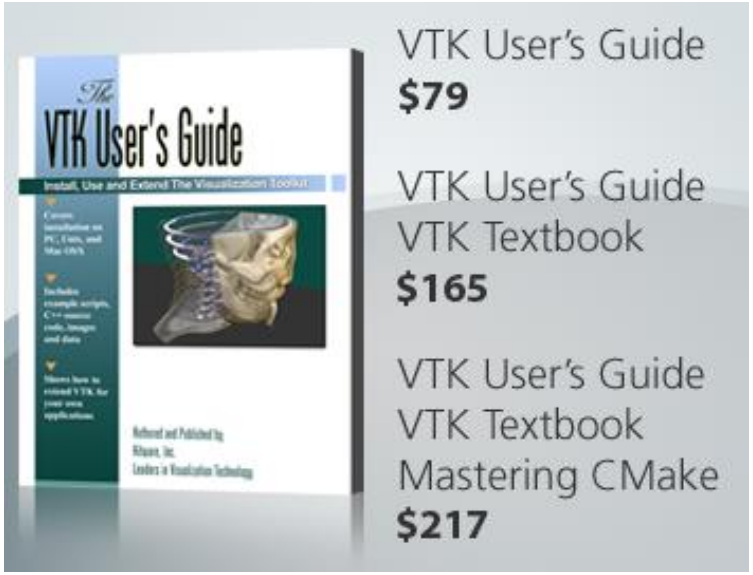
- Open-source, freely available software system for 3D computer graphics, image processing and visualization
- Supports a wide variety of visualization algorithms including: scalar, vector, tensor, texture, and volumetric methods; and advanced modeling techniques such as: implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation.
- Has an extensive information visualization framework, has a suite of 3D interaction widgets, supports parallel processing, and integrates with various databases on GUI toolkits such as Qt and Tk.
- Cross-platform and runs on Linux, Windows, Mac and Unix platforms.

Links

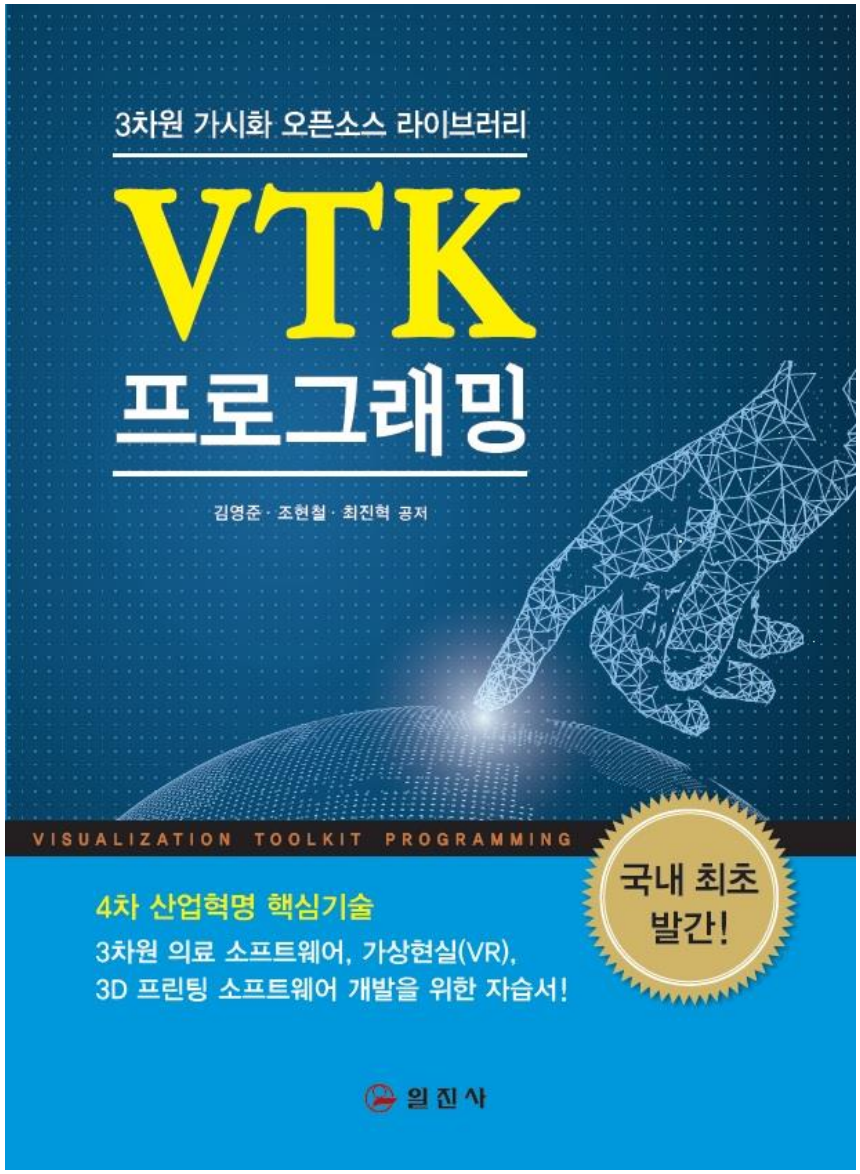
- http://www.vtk.org/
- http://www.vtk.org/Wiki/VTK/Tutorials
- http://www.vtk.org/Wiki/VTK/Tutorials/External_Tutorials

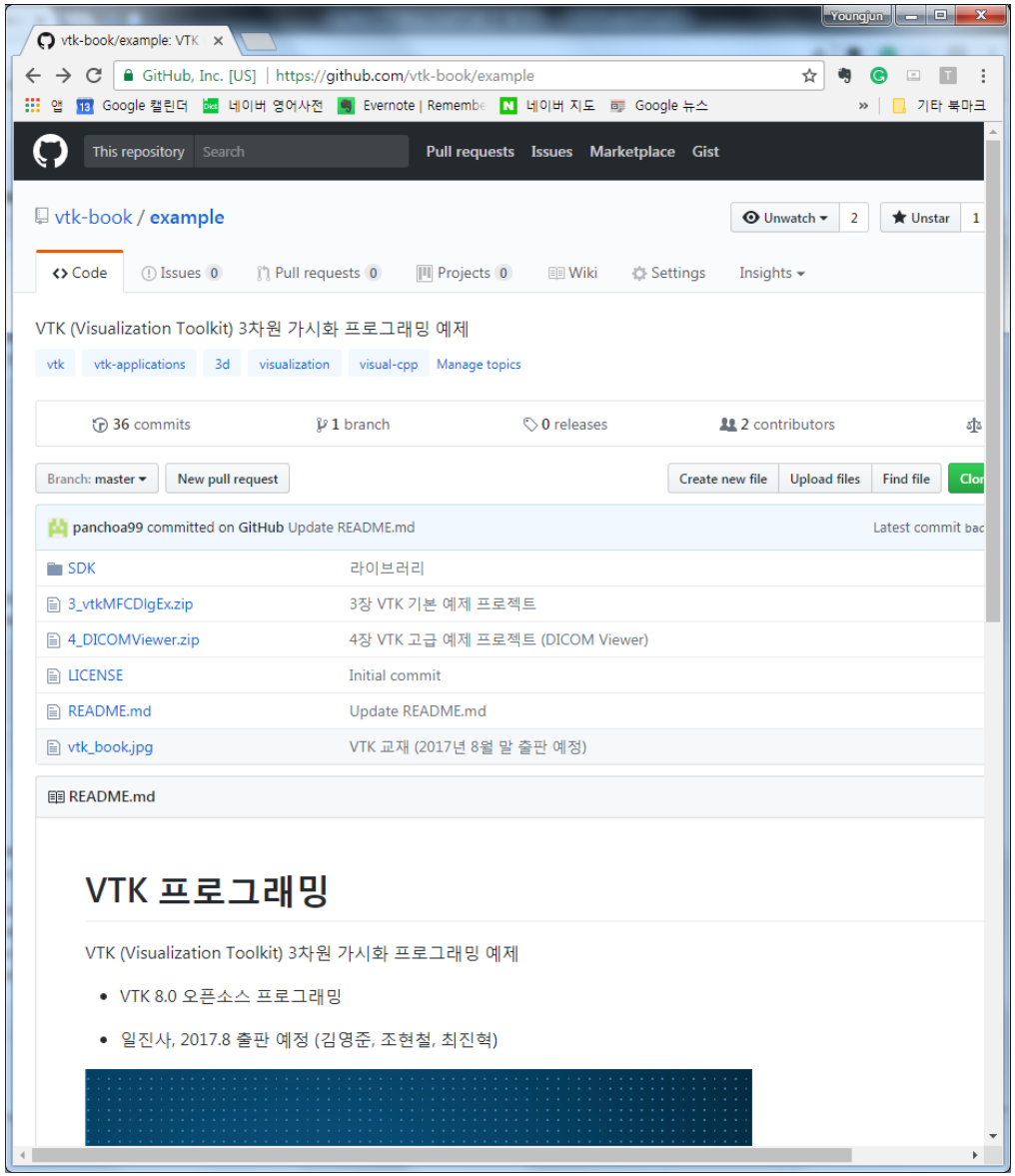
Books





https://github.com/vtk-book/example





Installation

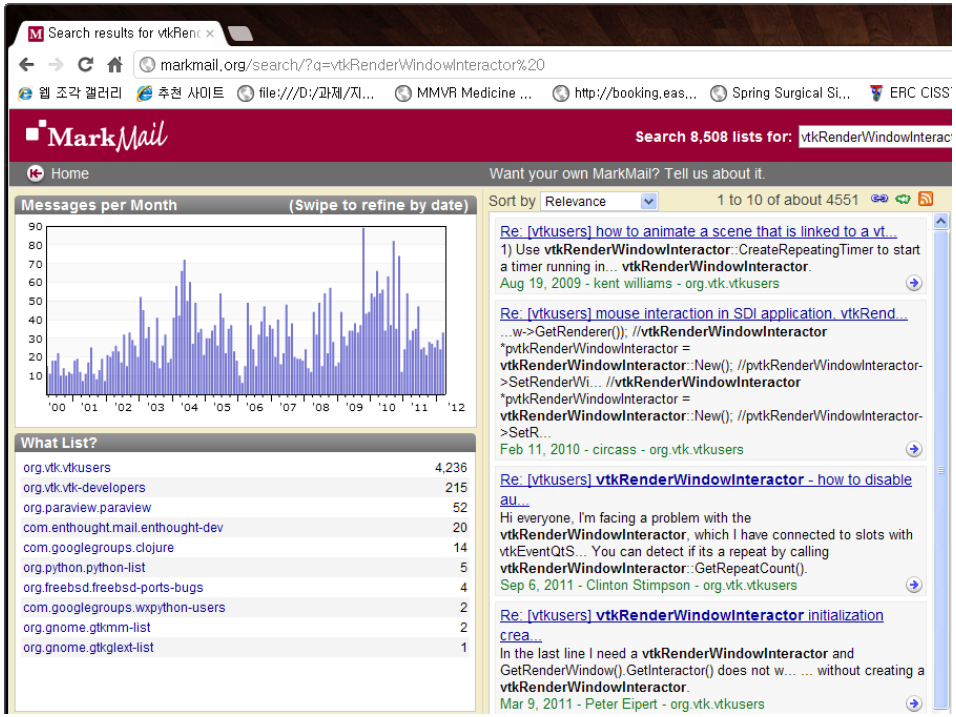
- https://github.com/vtk-book/example
  → VTK8.0_installation_guide.doc
- http://www.vtk.org/VTK/resources/software.html
- Latest release ver. 8.0.0
- Installer / Source installation
- Data
- Documentation

VTK8.0_installation_guide.doc

https://github.com/vtk-book/example

Tips

- Googling!!
- http://www.vtk.org/Wiki/VTK
- MarkMail (http://markmail.org/)



VTK – The Visualization ToolKit

- Open source, freely available software for 3D computer graphics, image processing, and visualization
- Managed by Kitware Inc.
- Strictly object-oriented design (C++)
- C++, Tcl/Tk, Python, Java

Features

- Visualization techniques for visualizing
  - Scalar fields
  - Vector fields
  - Tensor fields
- Mesh processing
- Image processing
- Volume rendering
- Your own algorithms

Additional features

- Parallel support (message passing, multithreading)
- Stereo support
- Integrates easily with Motif, Qt, Tcl/Tk, Python/Tk, X11, Windows, …
- Event handling
- 3D widgets

3D graphics

- Surface rendering
- Volume rendering
  - Ray casting
  - Texture mapping (2D)
  - Volume pro support
- Lights and cameras
- Textures
- Save render window to .png, .jpg, …

Visualization

- Data types
  - Polygonal data
    - points, lines, polygons, triangle strips
  - Images and volumes
  - Structured grids
  - Unstructured grids

- Scalar algorithms
  - Iso-contouring
  - Color mapping
- Vector algorithms
  - Streamlines / streamtubes

Pros and cons

(+)

- Free and open source
- Create graphics/visualization applications fairly fast
- Object oriented - easy to derive new classes
- Build applications using "interpretive" languages Tcl, Python, and Java
- Many (state of the art) algorithms
- Heavily tested in real-world applications
- Large user base provides decent support
- Commercial support and consulting available

Pros and cons

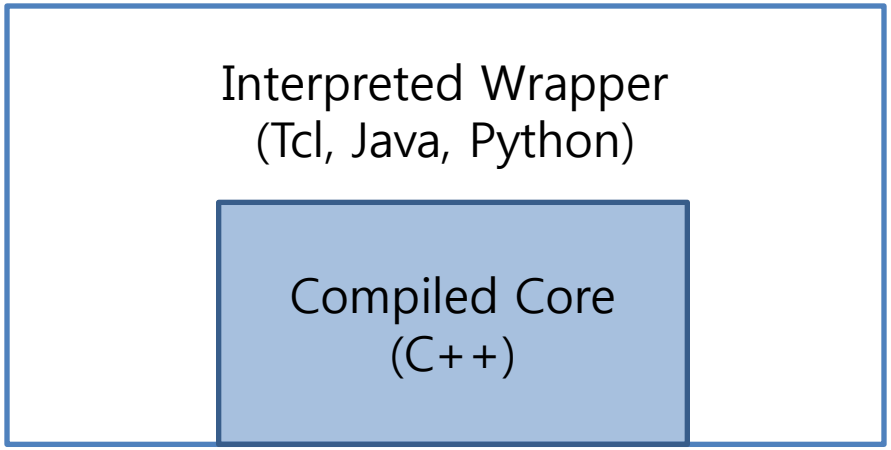
(-)

- Not a super-fast graphics engine due to portability and C++ dynamic binding – you need a decent workstation

- Very large class hierarchy => learning threshold might be steep

System architecture

- 2 basic subsystems
  - Compiled C++ class library
  - Interpreted wrapper layer



Note: Applications can be built entirely in C++.

Object-oriented system

- As VTK is an object-oriented system, the access of class and instance data members is carefully controlled in VTK.
- Example:

  vtkSetMacro(Tolerance,double);

  vtkGetMacro(Tolerance,double);

  →

  virtual void SetTolerance(double);

  virtual double GetTolerance();

Reference counting

vtkObjectBase* obj = vtkExampleClass::New(); → ref. cnt: 1

otherObject->SetExample(obj); → ref. cnt: 2

obj->Delete(); → ref. cnt: 0

- Register() and UnRegister() methods:

  increases and decreases ref. counting

  → this is handled automatically by the various "set" method

- **"Smart pointer"** automatically manages ref. counting!

```
vtkSmartPointer<vtkObjectBase> obj =
    vtkSmartPointer<vtkExampleClass>::New();
otherObjtect->SetExample( obj );
```

Rendering engine (1)

- vtkProp
  - vtkActor (for geometric data)
  - vtkVolume (for volumetric data)
  - vtkActor2D (for 2D data)

  → position, orientation, scale…

  * vtkLODActor / vtkLODProp3D: automatically changes its geometric representation / # of mappers to maintain interactivity

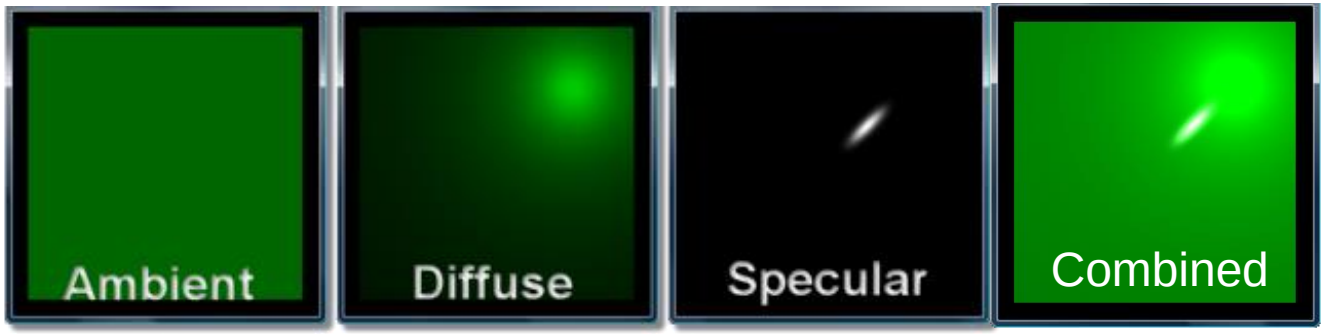
Rendering engine (2)

- vtkAbstractMapper
- vtkProperty / vtkVolumeProperty
  - Color, opacity, ambient,diffuse, specular coefficient of material
  - Transfer functions (map the scalar value to color & opacity)
- vtkCamera
- vtkLight
- vtkRenderer
  - Managing the rendering process for the scene

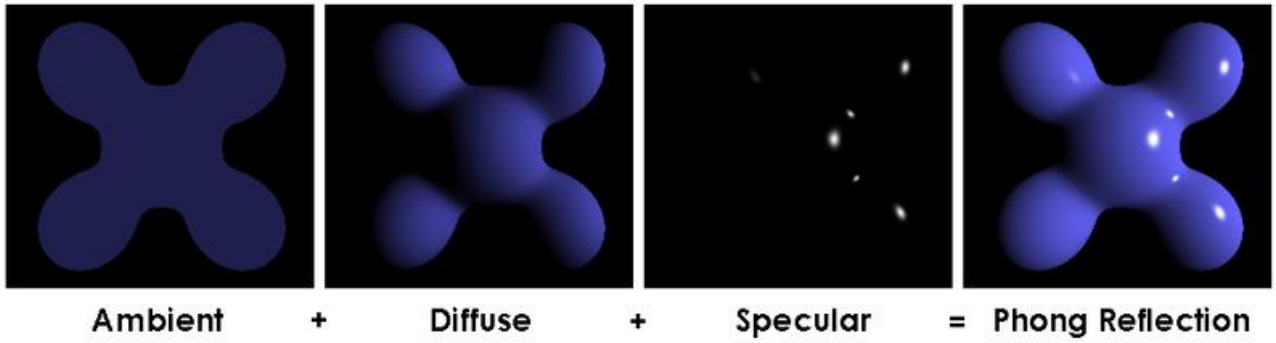
Lighting in Computer Graphics

Ambient component: constant lighting, radiant effect in lighting
Diffuse component: simulates re-emission from a non-shiny(irregular) surface
Specular component: bright highlights on a surface look shiny, mirror-like surface

Lighting



Reflection



http://xoax.net/cpp/crs/opengl/lessons/Lesson4/
http://www.tomdalling.com/blog/modern-opengl/07-more-lighting-ambient-specular-attenuation-gamma/

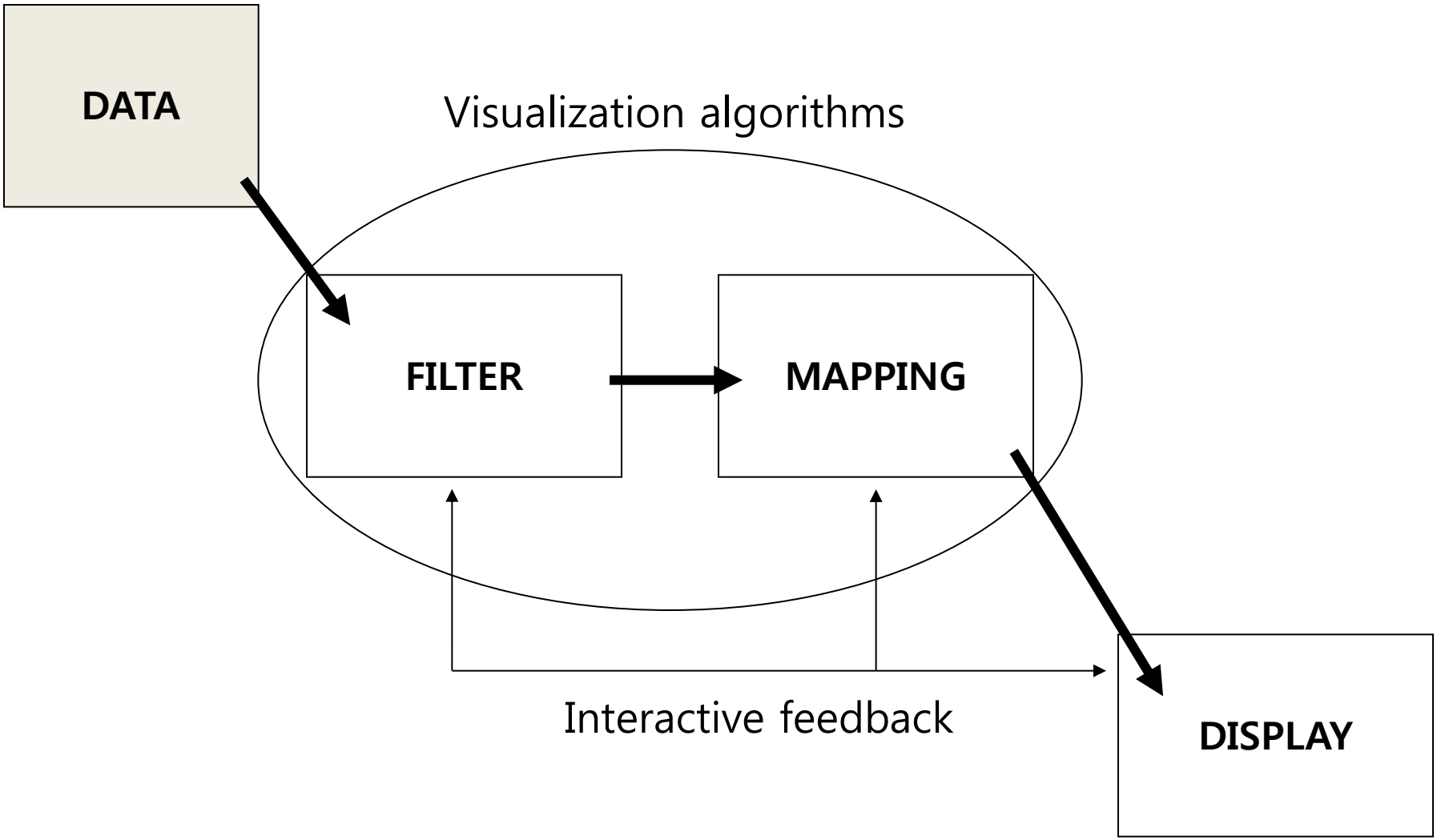
Rendering engine (3)

- vtkRenderWindow
  - Provides a connection btw. OS and VTK rendering engine
  - Contains a collection of vtkRenderers,
    parameters to control the features (e.g. stereo, anti-alising,..)
- vtkRenderWindowInteractor
  - Processing mouse, key, time events
- vtkTransform
  - Set position & orientation for objects (props, lights, camears..)
- vtkLookupTable, vtkColorTransferFunction, vtkPiecewiseFunction

Rendering engine example

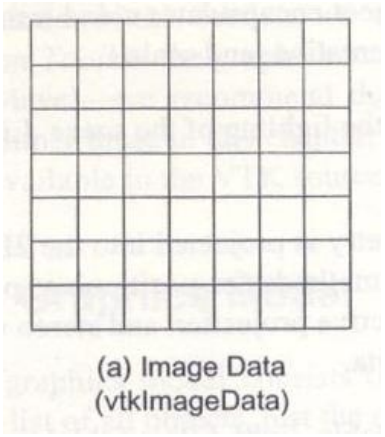
```
vtkCylinderSource* cylinder = vtkCylinderSource::New();
vtkPolyDataMapper* cylinderMapper = vtkPolyDataMapper::New();
cylinderMapper->SetInputConnection( cylinder->GetOutputPort() );

vtkActor* cylinderActor = vtkActor::New();
cylinderActor->SetMapper( cylinderMapper );

vtkRender* ren1 = vtkRenderer::New();
ren1->AddActor( cylinderActor );

vtkRenderWindow* renWin = vtkRenderWindow::New();
renWin->AddRenderer( ren1 );

vtkRenderWindowInteractor* iren = tkRenderWindowInteractor::New();
iren->SetRenderWindow( renWin );

renWin->Render();    iren->Start();
```

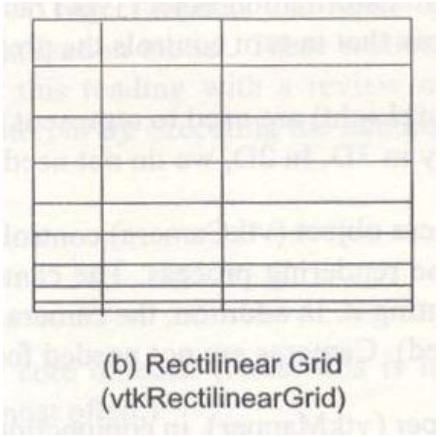
The visualization pipeline (1)

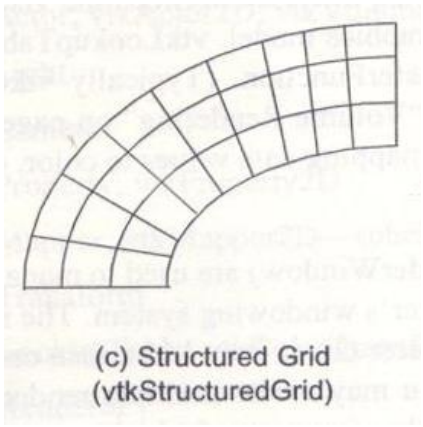


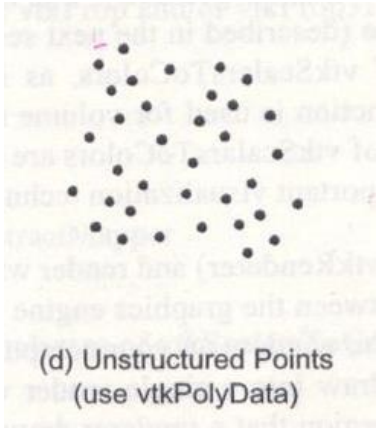
The visualization pipeline (2)

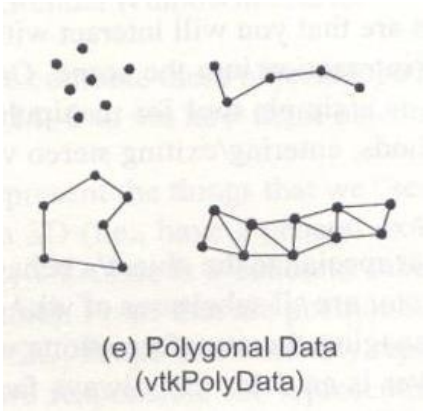
- Dataset types in VTK

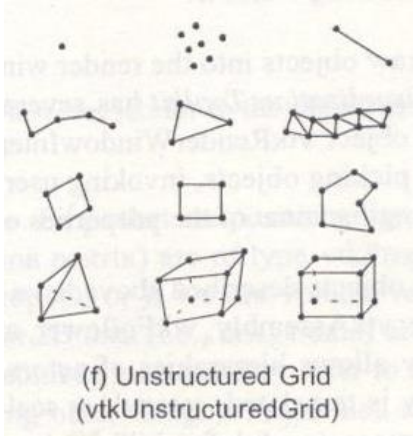

(a) Image Data
(vtkImageData)


(b) Rectilinear Grid
(vtkRectilinearGrid)


(c) Structured Grid
(vtkStructuredGrid)

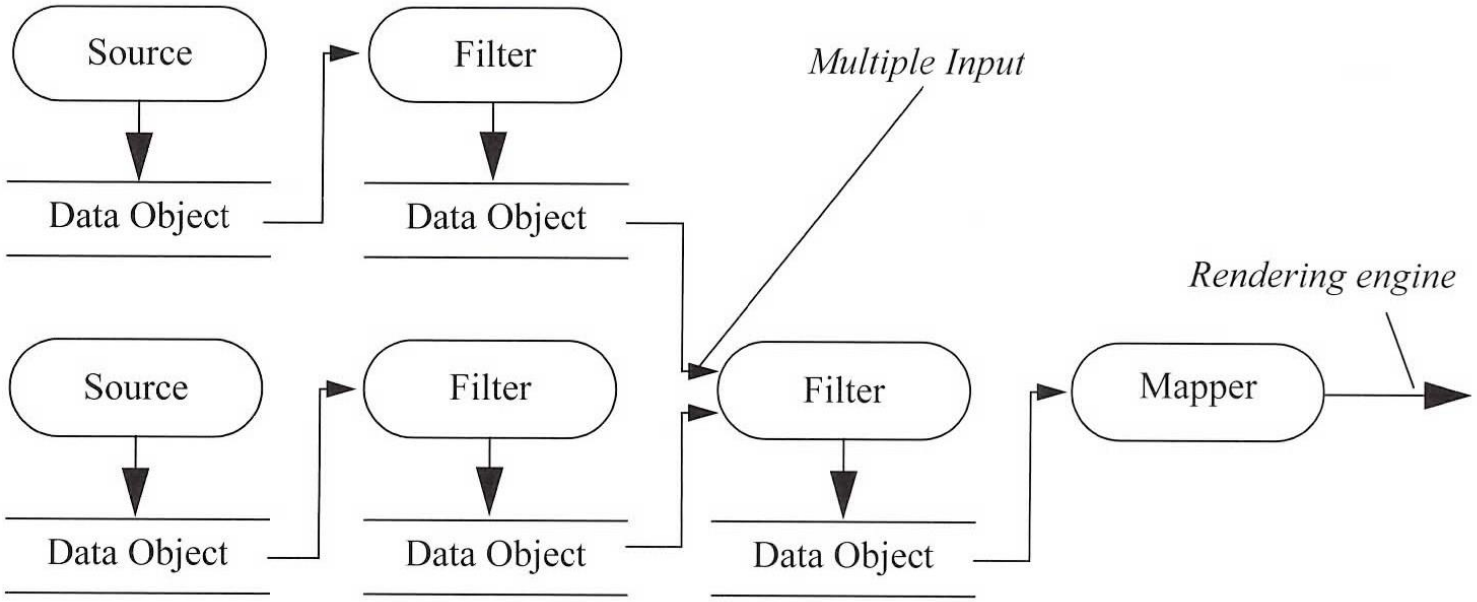

(d) Unstructured Points
(use vtkPolyData)


(e) Polygonal Data
(vtkPolyData)


(f) Unstructured Grid
(vtkUnstructuredGrid)

The visualization pipeline (3)

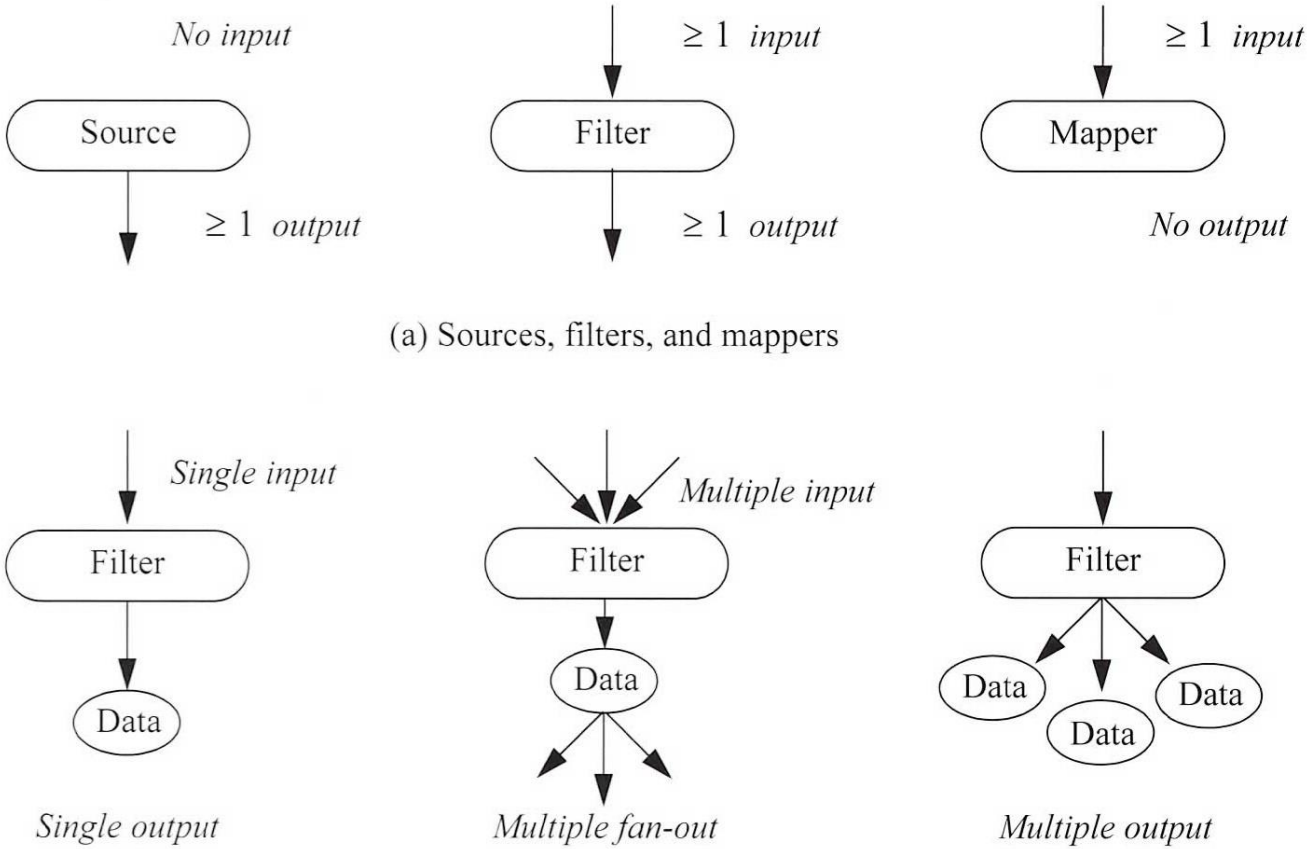
- vtkAlgorithm



Data object are connected with algorithms (filters) to create the visualization pipeline

The visualization pipeline (4)

- vtkAlgorithm



The visualization pipeline (5)

- Pipeline topology construction:

  aFilter->SetInputConnection(anotherFilter->GetOutputPort());

  → Sets the input to the filter "aFilter"

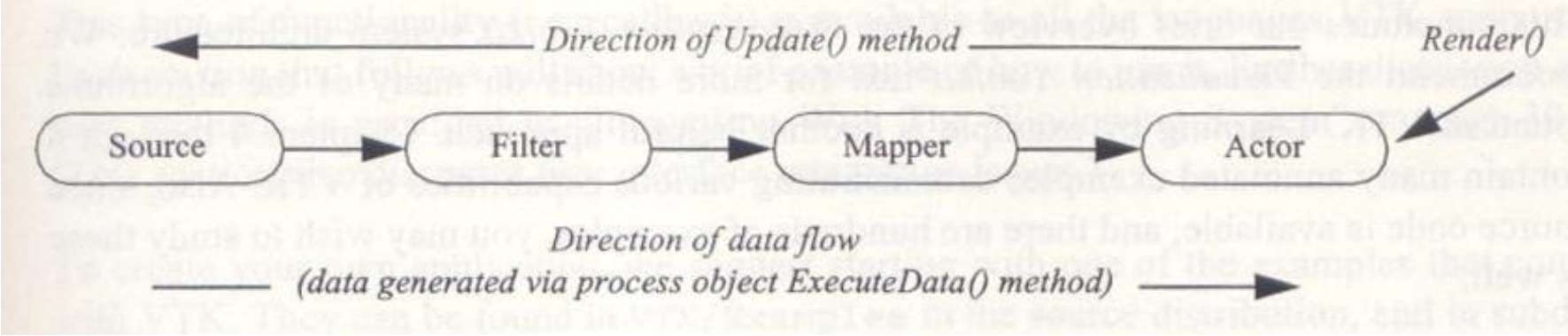
  to the output of the filter "anotherFilter"

  * VTK produces errors at run-time if the data object types are incompatible

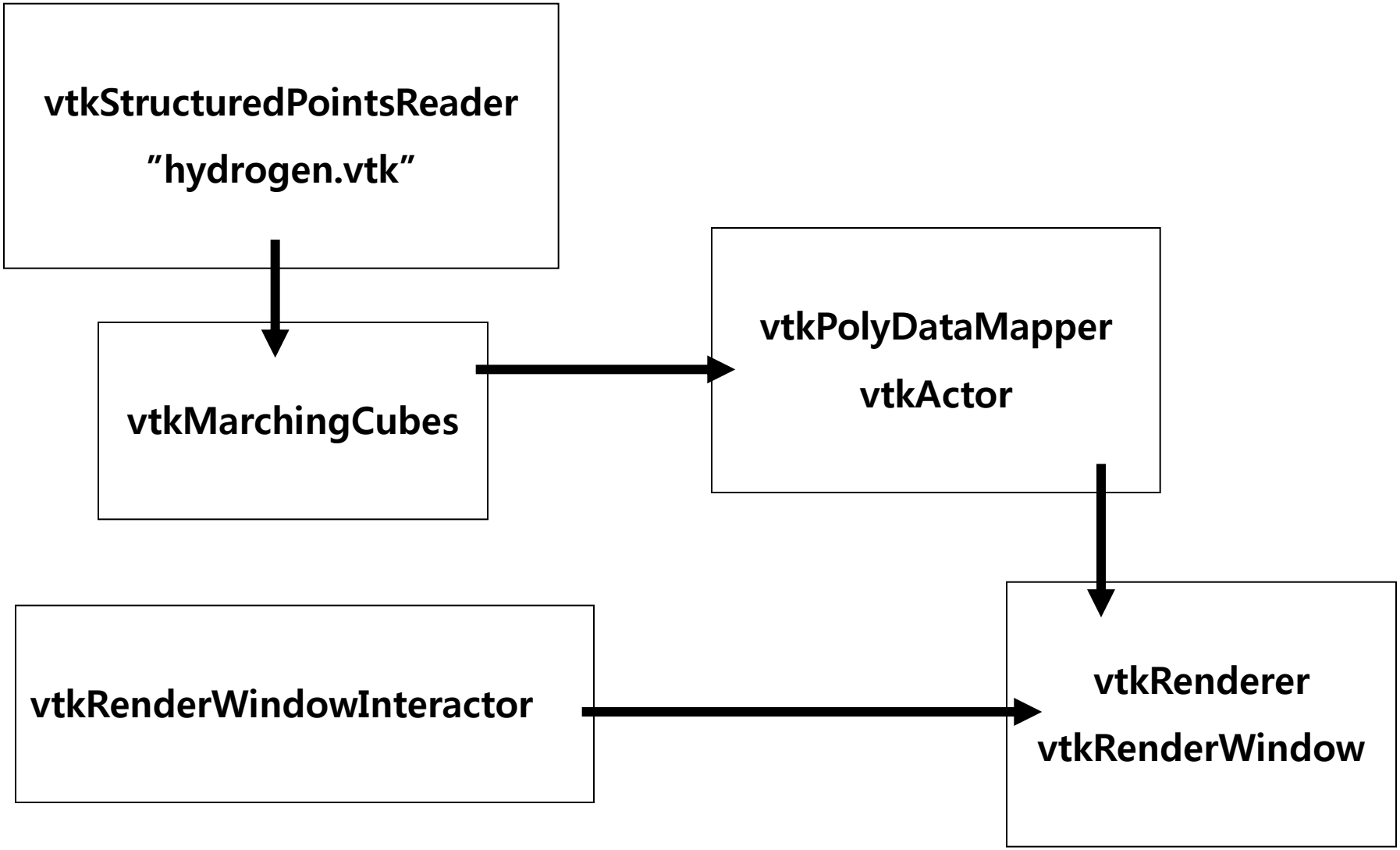
- Pipeline execution:
  - Update() method forces execution of the pipeline (not necessary)

  c.f.) Render() method often initiates the request for data

Conceptual overview of pipeline execution



The visualization pipeline - example



Objects

- Data objects
  - vtkPolyData
  - vtkImageData
- Process objects
  - Source objects (vtkReader, vtkSphereSource)
  - Filter objects (vtkContourFilter)
  - Mapper objects (vtkPolyDataMapper)

Conversion between languages

For example, the C++ statement

```
anActor->GetProperty()->SetColor(red, green, blue);
```

in Tcl becomes

```
[anActor GetProperty] SetColor $red $green $blue
```

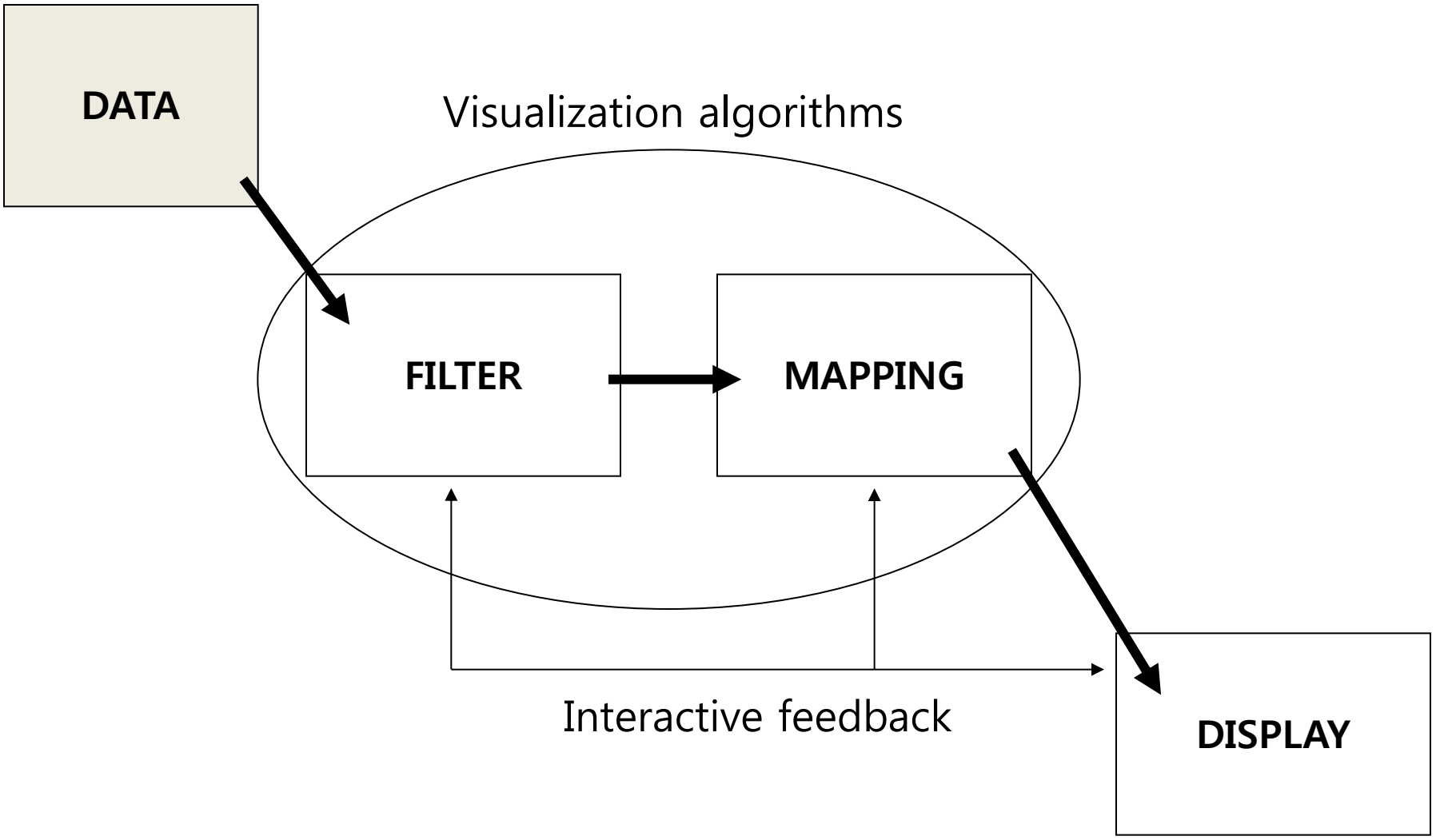
in Java becomes

```
anActor.GetProperty().SetColor(red, green, blue);
```

and in Python becomes

```
anActor.GetProperty().SetColor(red, green, blue)
```

The visualization pipeline



Rendering engine example

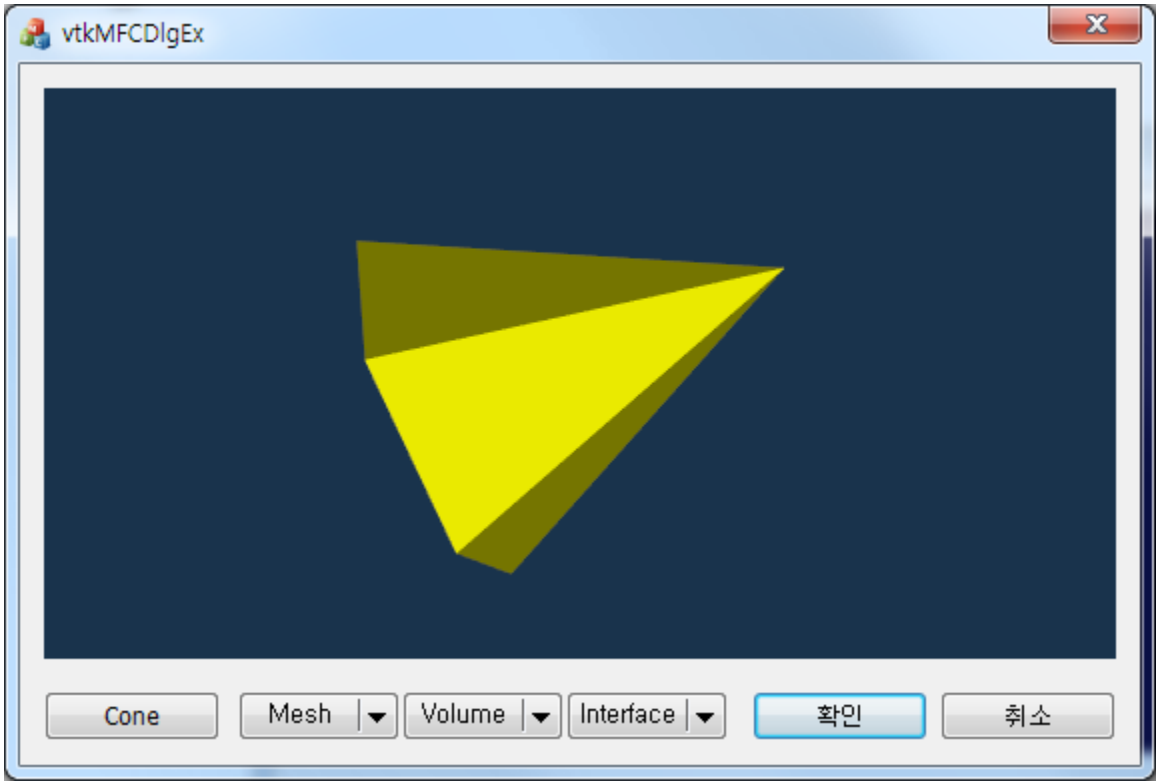
```
vtkCylinderSource* cylinder = vtkCylinderSource::New();
vtkPolyDataMapper* cylinderMapper = vtkPolyDataMapper::New();
cylinderMapper->SetInputConnection( cylinder->GetOutputPort() );

vtkActor* cylinderActor = vtkActor::New();
cylinderActor->SetMapper( cylinderMapper );

vtkRender* ren1 = vtkRenderer::New();
ren1->AddActor( cylinderActor );

vtkRenderWindow* renWin = vtkRenderWindow::New();
renWin->AddRenderer( ren1 );

vtkRenderWindowInteractor* iren = tkRenderWindowInteractor::New();
iren->SetRenderWindow( renWin );

renWin->Render();    iren->Start();
```

VTK for Scientific Visualization

MESH DATA PROCESSING

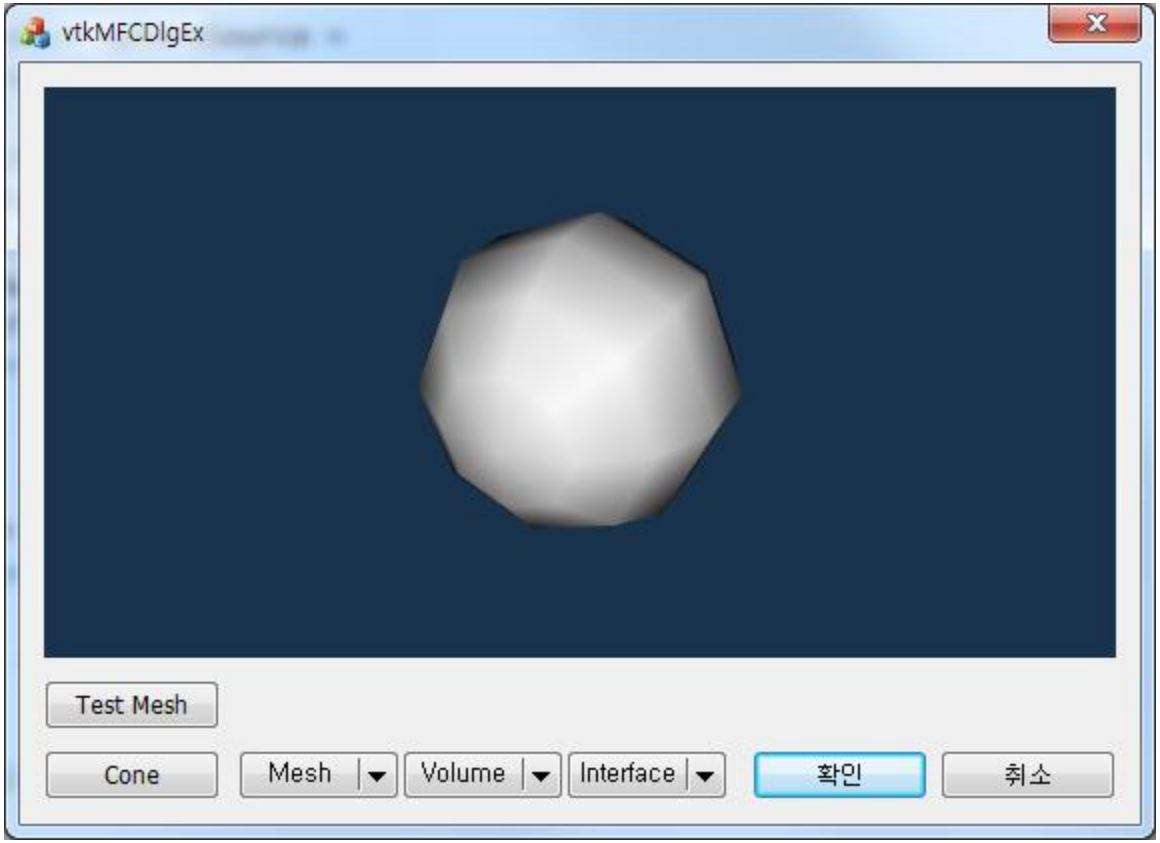
MFC application

- https://github.com/vtk-book/example
  → **3_vtkMFCDlgEx.zip**



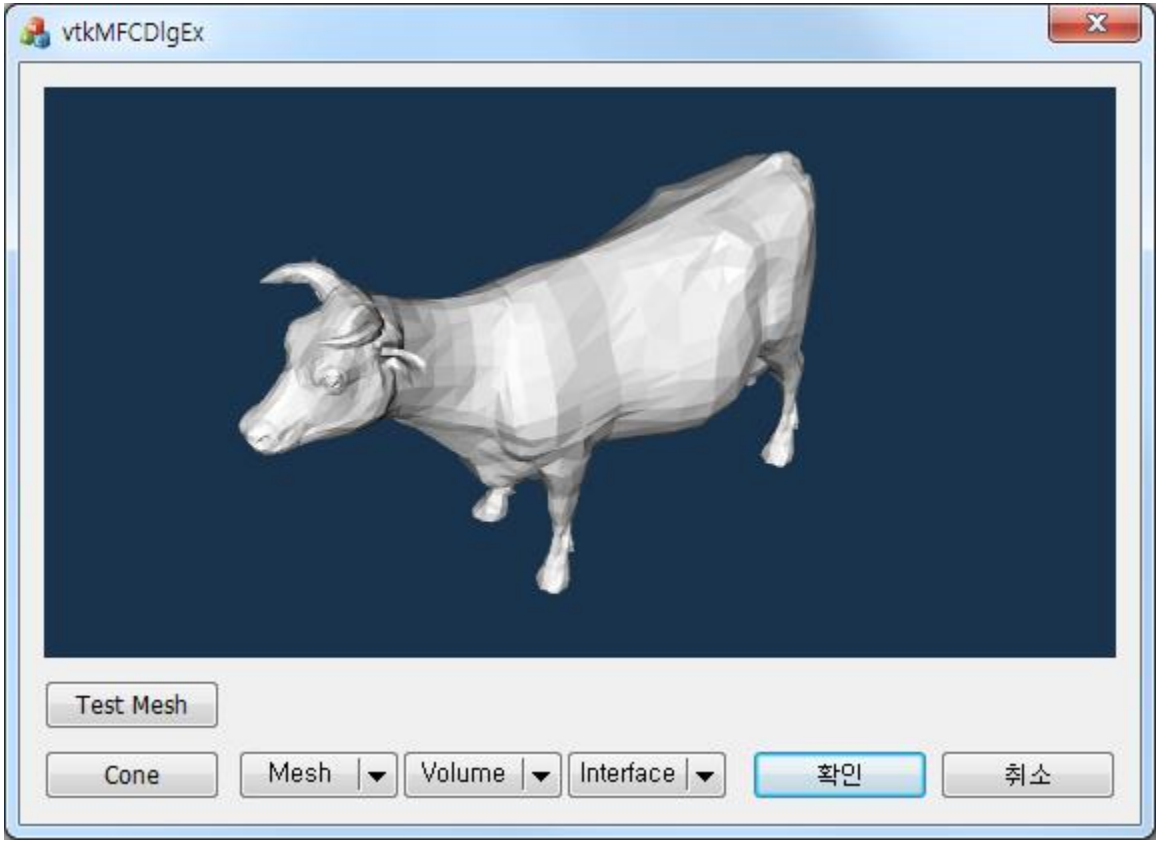
- VTK 교재
  - 제3장 VTK 실습 > 3.1절 VTK 프레임워크 프로젝트 생성하기

Sphere



```
void CvtkMFCDlgExDlg::OnBnClickedButtonTestMesh()
{
    vtkSmartPointer<vtkRenderer> prevRenderer = m_vtkWindow->GetRenderers()->GetFirstRenderer();
    if (prevRenderer != NULL) m_vtkWindow->RemoveRenderer(prevRenderer);

    // Create a sphere source
    vtkSmartPointer<vtkSphereSource> sphereSource =
        vtkSmartPointer<vtkSphereSource>::New();

    // Create a mapper and an actor
    vtkSmartPointer<vtkPolyDataMapper> mapper =
        vtkSmartPointer<vtkPolyDataMapper>::New();
    mapper->SetInputConnection(sphereSource->GetOutputPort());
    vtkSmartPointer<vtkActor> actor =
        vtkSmartPointer<vtkActor>::New();
    actor->SetMapper(mapper);

    // Visualize
    vtkSmartPointer<vtkRenderer> renderer =
        vtkSmartPointer<vtkRenderer>::New();
    renderer->AddActor(actor);
    renderer->SetBackground(.1, .2, .3); // Background color dark blue
    renderer->ResetCamera();

    ////////////////////////////////////////////////////////////////////////
    //rendering
    m_vtkWindow->AddRenderer(renderer);
    m_vtkWindow->Render();
}
```

STL file reader



vtk stl file open

전체 이미지 동영상 뉴스 지도 더보기 설정 도구

검색결과 약 35,600개 (0.44초)

VTK/Examples/Cxx/IO/ReadSTL - KitwarePublic

www.vtk.org/Wiki/VTK/Examples/Cxx/IO/ReadSTL ▾ 이 페이지 번역하기

2012. 12. 9. - Click here to download ReadSTL. and its CMakeLists.txt **file**. Once the tarball ReadSTL.tar has been downloaded and extracted, cd ReadSTL/ ...

VTK/Examples/Python/STLReader - KitwarePublic

www.vtk.org/Wiki/VTK/Examples/Python/STLReader ▾ 이 페이지 번역하기

2014. 1. 18. - This code snippet reads an **STL file** and creates a PolyData output ... #!/usr/bin/env python import **vtk** filename = "myfile.stl" reader = **vtk**.

VTK/Examples/Cxx/IO/WriteSTL - KitwarePublic

www.vtk.org/Wiki/VTK/Examples/Cxx/IO/WriteSTL ▾ 이 페이지 번역하기

2016. 5. 18. - An **STL file** describes a triangulated three-dimensional surface by the unit normal and vertices (ordered by the right-hand rule) of the triangles.

VTK - Users - Display a STL file with vtk and Java _ vtkSTLReader

vtk.1045678.n5.nabble.com/Display-a-STL-file-with-vtk-and-Jav... ▾ 이 페이지 번역하기

2014. 5. 26. - 댓글 3 - 작성자 2

Hi, I begin to use Java and **VTK**, I search how display a **STL file** (porsche.stl) in java with **VTK**. ... Visit other Kitware **open**-source projects at ...

VTK/Examples/Cxx/IO/R

www.vtk.org/Wiki/VTK/Examples/Cxx/IO/ReadSTL

앱 Google 캘린더 네이버 영어사전 Evernote | Remembe 네이버 지도 Google 뉴스 Online LaTeX Equati 기타 북마크

page | discussion | view source | history

Public Wiki

navigation
- Main page
- Recent changes
- Random page
- Help

search
Search
Go Search

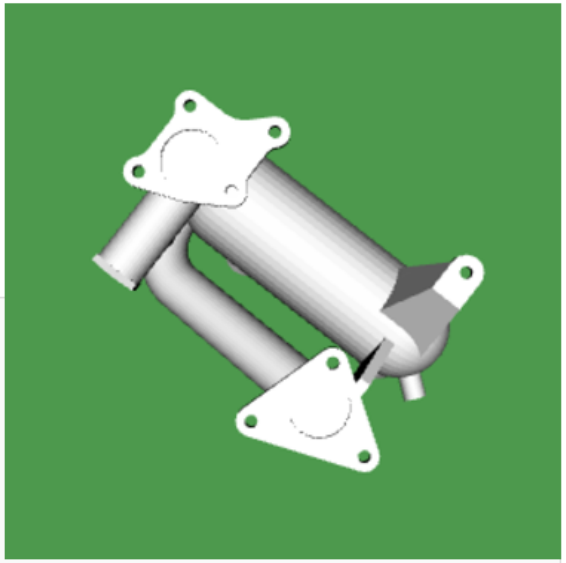
tools
- What links here
- Related changes
- Special pages
- Permanent link
- Page information

print/export
- Create a book
- Download as PDF
- Printable version

VTK/Examples/Cxx/IO/ReadSTL

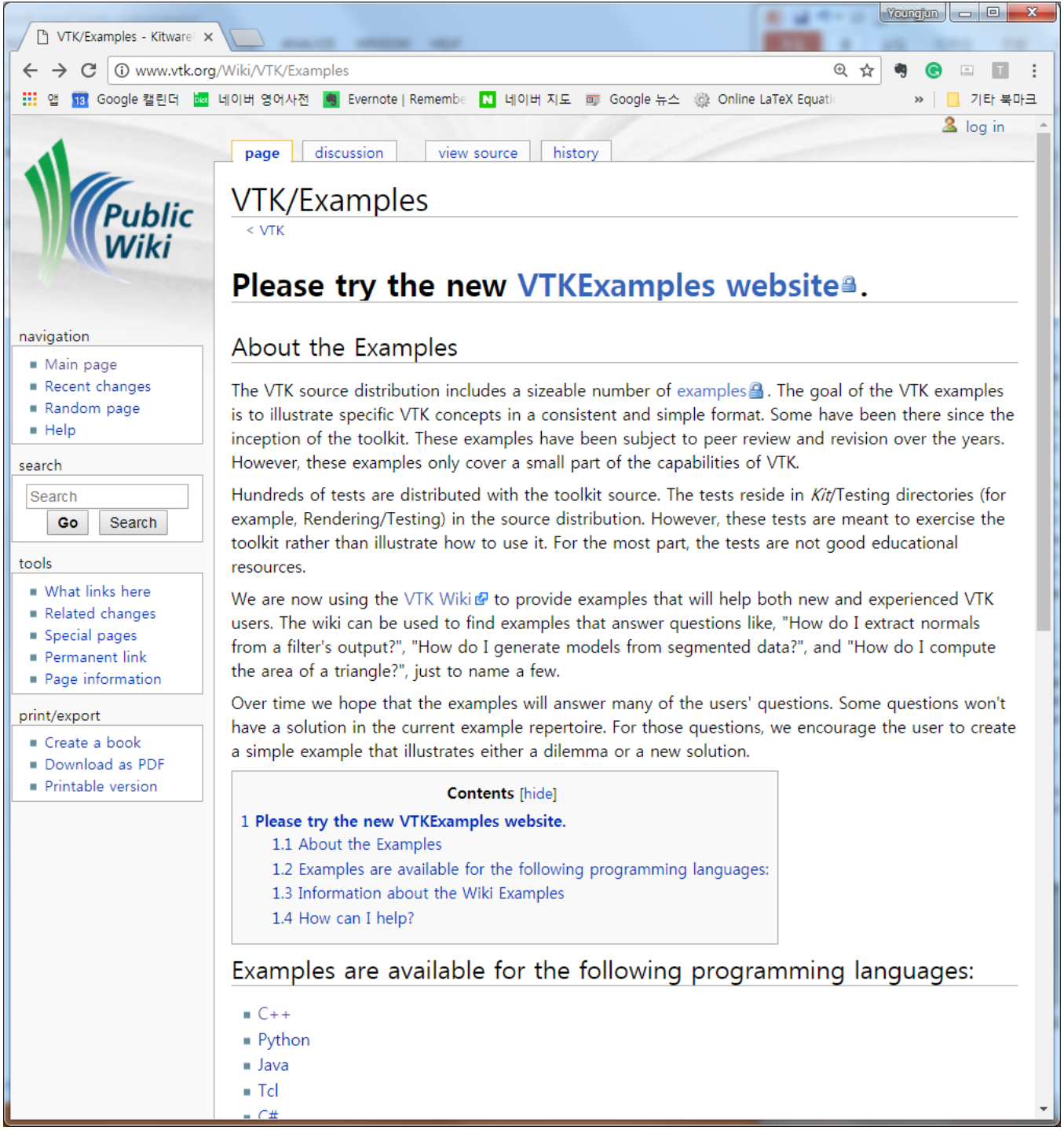
< VTK | Examples | Cxx

Contents [hide]



ReadSTL.cxx

```
#include <vtkPolyData.h>
#include <vtkSTLReader.h>
#include <vtkSmartPointer.h>
#include <vtkPolyDataMapper.h>
#include <vtkActor.h>
#include <vtkRenderWindow.h>
#include <vtkRenderer.h>
#include <vtkRenderWindowInteractor.h>

int main ( int argc, char *argv[] )
{
  if ( argc != 2 )
    {
    cout << "Required parameters: Filename" << endl;
    return EXIT_FAILURE;
    }

  std::string inputFilename = argv[1];

  vtkSmartPointer<vtkSTLReader> reader =
    vtkSmartPointer<vtkSTLReader>::New();
  reader->SetFileName(inputFilename.c_str());
  reader->Update();

  // Visualize
  vtkSmartPointer<vtkPolyDataMapper> mapper =
    vtkSmartPointer<vtkPolyDataMapper>::New();
  mapper->SetInputConnection(reader->GetOutputPort());
```


VTK/Examples - Kitware
www.vtk.org/Wiki/VTK/Examples
log in
page
discussion
view source
history
Public Wiki
VTK/Examples
< VTK
Please try the new VTKExamples website.
About the Examples
The VTK source distribution includes a sizeable number of examples. The goal of the VTK examples is to illustrate specific VTK concepts in a consistent and simple format. Some have been there since the inception of the toolkit. These examples have been subject to peer review and revision over the years. However, these examples only cover a small part of the capabilities of VTK.
Hundreds of tests are distributed with the toolkit source. The tests reside in Kit/Testing directories (for example, Rendering/Testing) in the source distribution. However, these tests are meant to exercise the toolkit rather than illustrate how to use it. For the most part, the tests are not good educational resources.
We are now using the VTK Wiki to provide examples that will help both new and experienced VTK users. The wiki can be used to find examples that answer questions like, "How do I extract normals from a filter's output?", "How do I generate models from segmented data?", and "How do I compute the area of a triangle?", just to name a few.
Over time we hope that the examples will answer many of the users' questions. Some questions won't have a solution in the current example repertoire. For those questions, we encourage the user to create a simple example that illustrates either a dilemma or a new solution.
Contents [hide]
1 Please try the new VTKExamples website.
1.1 About the Examples
1.2 Examples are available for the following programming languages:
1.3 Information about the Wiki Examples
1.4 How can I help?
Examples are available for the following programming languages:
C++
Python
Java
Tcl
C#
navigation
Main page
Recent changes
Random page
Help
search
Search
Go
Search
tools
What links here
Related changes
Special pages
Permanent link
Page information
print/export
Create a book
Download as PDF
Printable version

#1

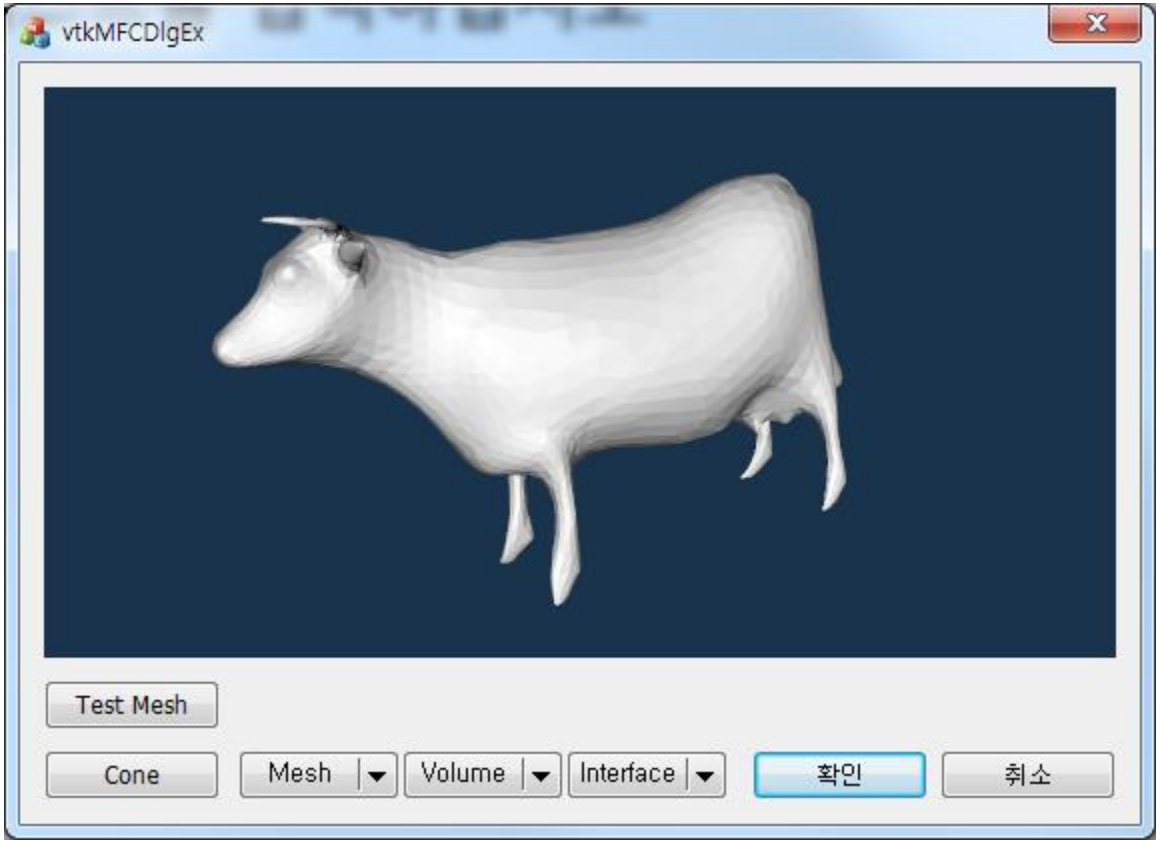
```
// STL file reader
vtkSmartPointer<vtkSTLReader> pSTLReader =
    vtkSmartPointer<vtkSTLReader>::New();
pSTLReader->SetFileName("../data/example.stl"); // 읽을 파일 지정
pSTLReader->Update();

// Mapper
vtkSmartPointer<vtkPolyDataMapper> mapper =
    vtkSmartPointer<vtkPolyDataMapper>::New();
mapper->SetInputConnection(pSTLReader->GetOutputPort());

// Actor
vtkSmartPointer<vtkActor> actor =
    vtkSmartPointer<vtkActor>::New();
actor->SetMapper(mapper);

////////////////////////////////////////////////////////////////////////
// Visualize
vtkSmartPointer<vtkRenderer> renderer =
    vtkSmartPointer<vtkRenderer>::New();
renderer->AddActor(actor);
renderer->SetBackground(.1, .2, .3); // Background color dark blue
renderer->ResetCamera();

//rendering
m_vtkWindow->AddRenderer(renderer);
m_vtkWindow->Render();
```

#2

```
// STL file reader
vtkSmartPointer<vtkSTLReader> pSTLReader =
    vtkSmartPointer<vtkSTLReader>::New();
pSTLReader->SetFileName("../data/example.stl"); // 읽을 파일 지정
pSTLReader->Update();

vtkSmartPointer<vtkPolyData> pPolyData =
    pSTLReader->GetOutput();                // vtkPolyData 형식으로 받아오기

// Create a mapper and actor
vtkSmartPointer<vtkPolyDataMapper> mapper =
    vtkSmartPointer<vtkPolyDataMapper>::New();
mapper->SetInputData(pPolyData);
vtkSmartPointer<vtkActor> actor =
    vtkSmartPointer<vtkActor>::New();
actor->SetMapper(mapper);

// Visualize
vtkSmartPointer<vtkRenderer> renderer =
    vtkSmartPointer<vtkRenderer>::New();
renderer->AddActor(actor);
renderer->SetBackground(.1, .2, .3); // Background color dark blue
renderer->ResetCamera();

////////////////////////////////////////////////////////////////////////
//rendering
m_vtkWindow->AddRenderer(renderer);
m_vtkWindow->Render();
```

Filtering Data

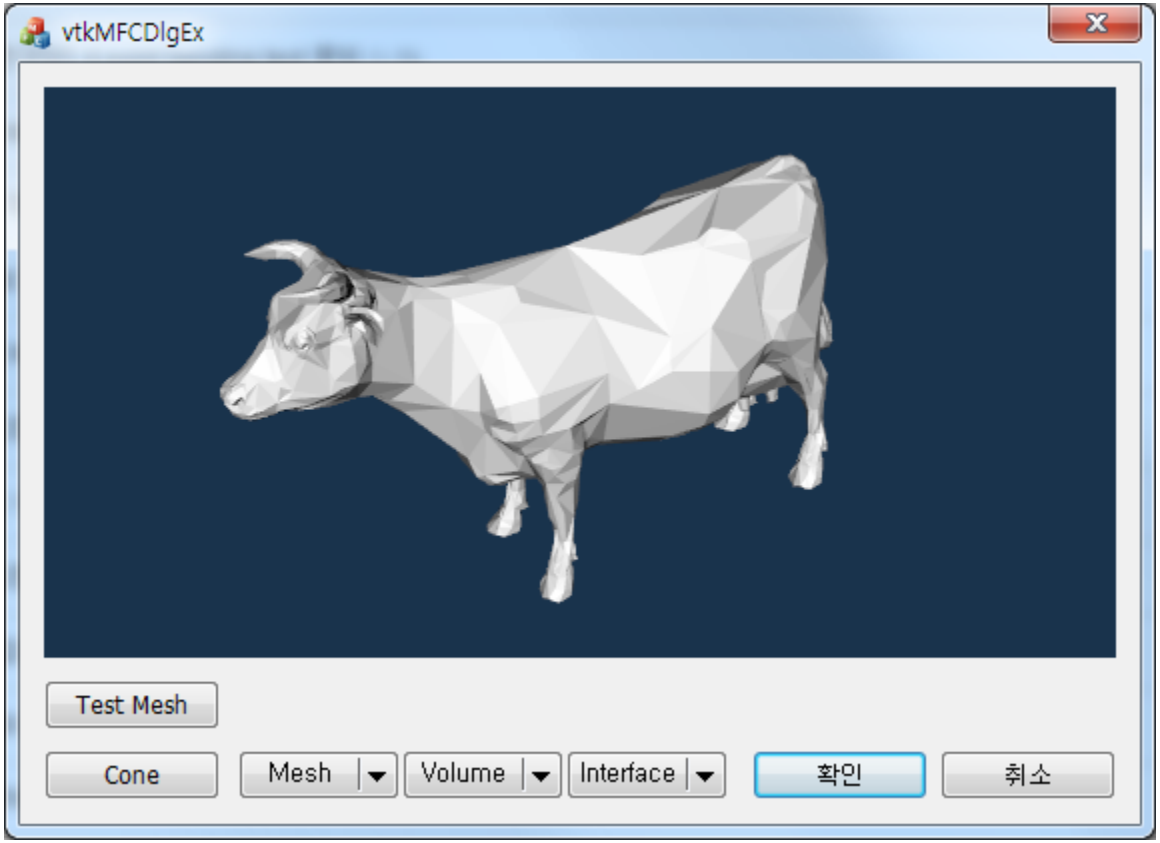
- Smoothing



```
// STL 파일 읽어오기
vtkSmartPointer<vtkSTLReader> stlReader =
    vtkSmartPointer<vtkSTLReader>::New();
stlReader->SetFileName( "../data/example.stl" );
stlReader->Update();

////////////////////////////////////////////////////////////
// Filter 처리 vtkPolyData(dental.stl) →
// Smoothing
vtkSmartPointer<vtkWindowedSincPolyDataFilter> smoothFilter =
    vtkSmartPointer<vtkWindowedSincPolyDataFilter>::New();
smoothFilter->SetInputConnection( stlReader->GetOutputPort() );
smoothFilter->SetNumberOfIterations( 100 );     //반복연산 횟수
smoothFilter->Update();

// Create a mapper and actor
vtkSmartPointer<vtkPolyDataMapper> mapper =
    vtkSmartPointer<vtkPolyDataMapper>::New();
mapper->SetInputConnection( smoothFilter->GetOutputPort() );

vtkSmartPointer<vtkActor> actor =
    vtkSmartPointer<vtkActor>::New();
actor->SetMapper( mapper );
```

Filtering Data

- Decimation

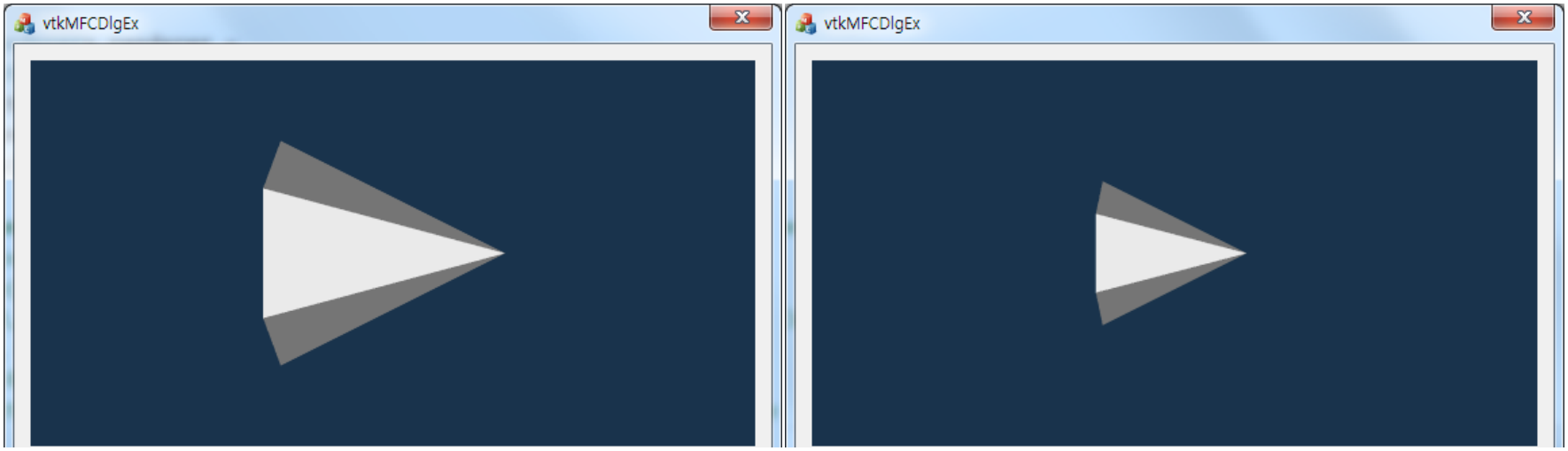


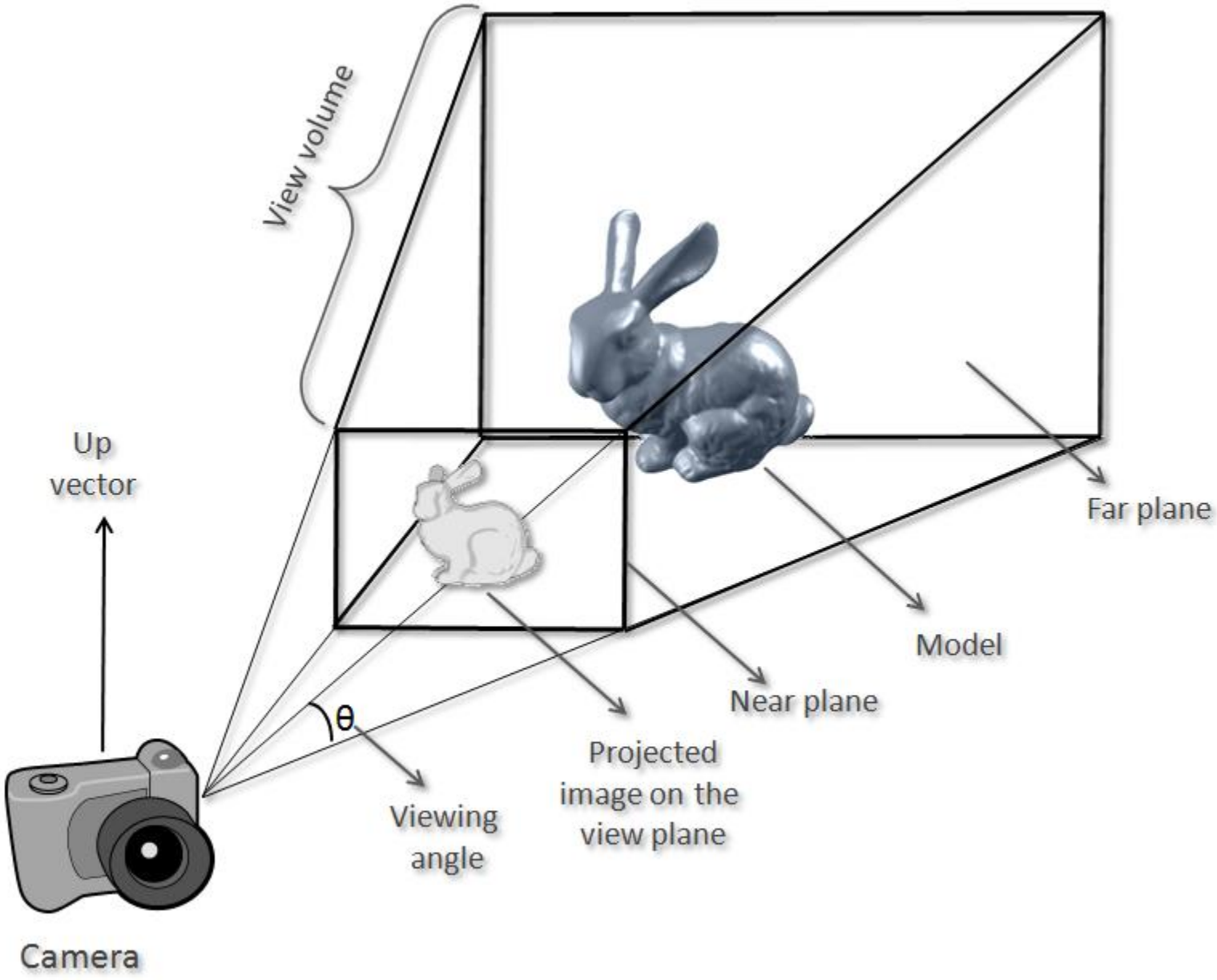
Decimation

- Often, mesh data are quite large and cannot be rendered or processed quickly enough for interactive application.
- Decimation (polygon reduction / mesh simplification / multiresolution modeling)
  : reduce # of triangles, while maintaining a faithful approximation to the original mesh
  - vtkDecimatePro
  - vtkQuadricClustering

```
// STL 파일 읽어오기
vtkSmartPointer<vtkSTLReader> stlReader =
    vtkSmartPointer<vtkSTLReader>::New();
stlReader->SetFileName( "../data/example.stl" );
stlReader->Update();

////////////////////////////////////////////////////////////////////////
// Filter 처리 vtkPolyData(dental.stl) →
// 1) vtkDecimationPro
vtkSmartPointer<vtkDecimatePro> decimatePro =
    vtkSmartPointer<vtkDecimatePro>::New();
decimatePro->SetInputConnection( stlReader->GetOutputPort() );
decimatePro->SetTargetReduction( 0.9 ); // 전제 mesh 10% 감소
decimatePro->PreserveTopologyOn();
decimatePro->Update();

// 2) vtkQuadricClustering
vtkSmartPointer<vtkQuadricClustering> qClustering =
    vtkSmartPointer<vtkQuadricClustering>::New();
qClustering->SetInputConnection( stlReader->GetOutputPort() );
qClustering->SetNumberOfDivisions( 10, 10, 10 );  // 분할 개수 설정 (생략가능)
qClustering->Update();

// Create a mapper and actor
vtkSmartPointer<vtkPolyDataMapper> mapper =
    vtkSmartPointer<vtkPolyDataMapper>::New();
// 두 가지 결과 중 하나 선택하여 렌더링
mapper->SetInputConnection( decimatePro->GetOutputPort() );
//mapper->SetInputConnection( qClustering->GetOutputPort() );

vtkSmartPointer<vtkActor> actor =
    vtkSmartPointer<vtkActor>::New();
actor->SetMapper( mapper );
```

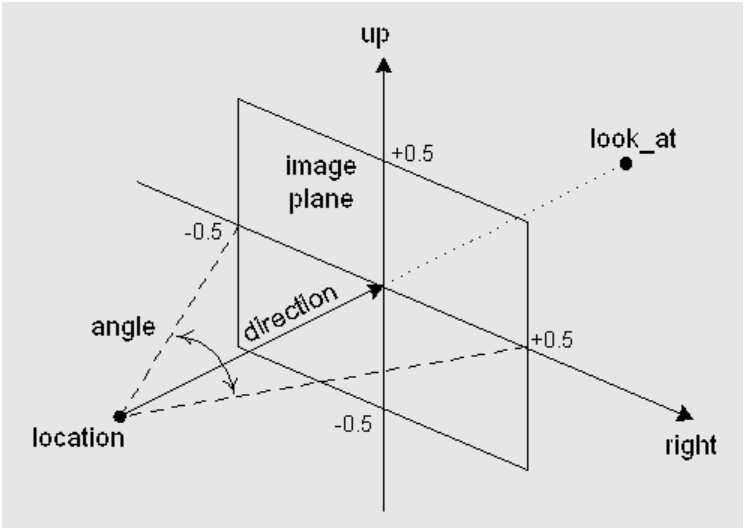
Controlling the Camera




View volume
Up
vector
Far plane
Model
Near plane
θ
Projected
image on the
view plane
Viewing
angle
Camera

- Controlling the View Direction

```
// Controlling the view direction
vtkCamera* cam1 = vtkCamera::New();
cam1->SetFocalPoint( 0, 0, 0 );
cam1->SetPosition( 1, 1, 1 );
cam1->ComputeViewPlaneNormal();
cam1->SetViewUp( 1, 0, 0 );
cam1->OrthogonalizeViewUp();

// Set active camera
ren1->SetActiveCamera( cam1 );
ren1->ResetCamera();
```



```
// Visualize
vtkSmartPointer<vtkRenderer> renderer =
    vtkSmartPointer<vtkRenderer>::New();
renderer->AddActor(actor);
renderer->SetBackground(.1, .2, .3); // Background color dark blue
renderer->ResetCamera();

vtkSmartPointer<vtkCamera> cam =
    renderer->GetActiveCamera();        // renderer에서 카메라 받아오기
cam->SetClippingRange( 0.1, 10 );   // 그려질 depth 영역 설정
cam->SetFocalPoint( 0, 0, 0 );          // 카메라가 바라보는 지점
cam->SetViewUp( 0, 1, 0 );              // 카메라의 upvector 설정
cam->SetPosition( 0, 0, 5 );            // 카메라 위치 설정
//cam->ParallelProjectionOn();          // on: orthogonal view / off: perspective view

////////////////////////////////////////////////////////////////////////
//rendering
m_vtkWindow->AddRenderer(renderer);
m_vtkWindow->Render();
```

Controlling 3D Props

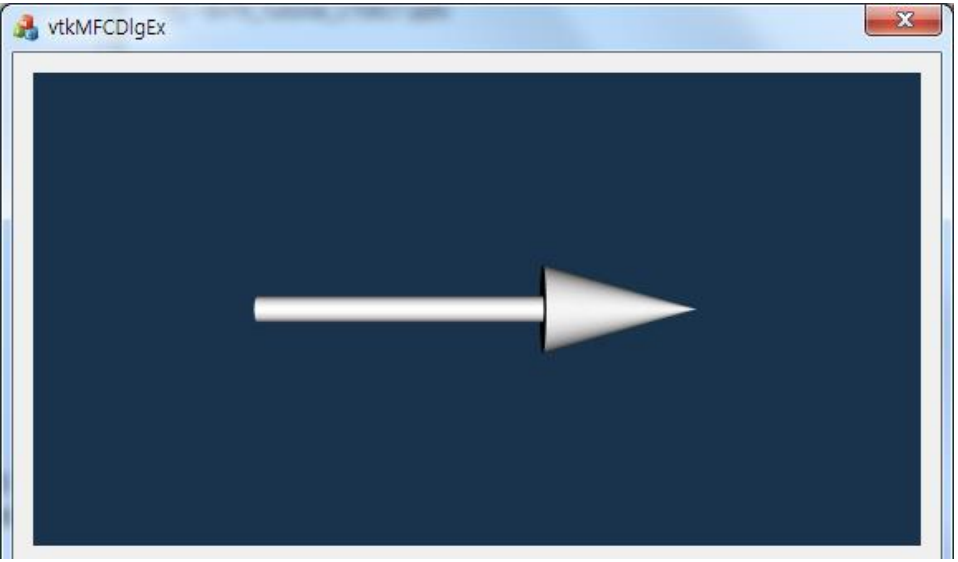
- Specifying the position of a vtkProp3D
  - SetPosition(x, y, z)
  - AddPosition(deltaX, deltaY, deltaZ)
  - RotateX(theta), ..
  - SetOrientation(x, y, z)
  - AddOrientation(a1, a2, a3)
  - RotateWXYZ(theta, x, y, z) → rotate around the x-y-z vector
  - Scale(sx, sy, sz)
  - SetOrigin(x, y, z)

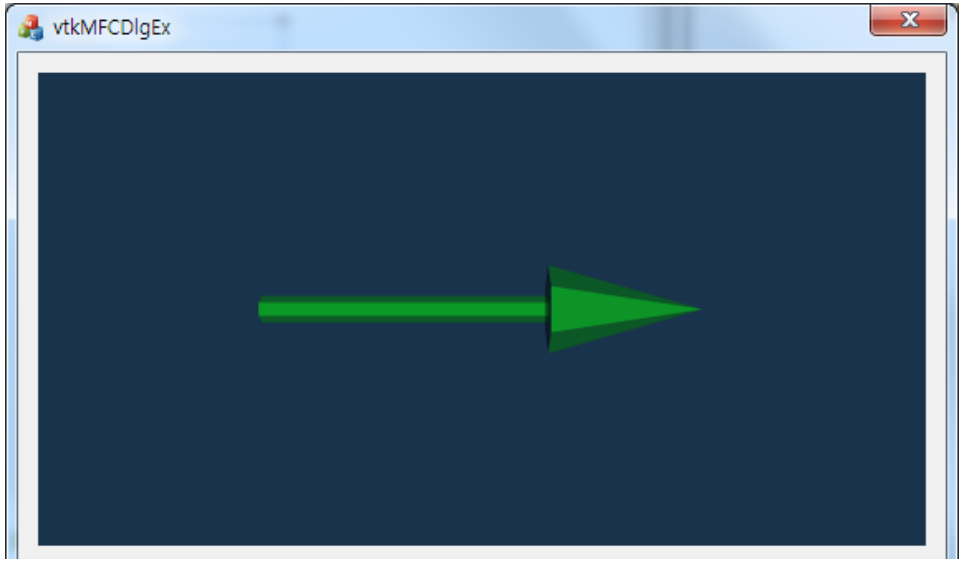
Note:

The order of applications affects the resulting actor position!

Actors

- Actor properties





```cpp
void CvtkMFCDlgExDlg::OnButtonExVtkproperty()
{
    vtkSmartPointer<vtkArrowSource> arrow =
        vtkSmartPointer<vtkArrowSource>::New();             // 화살표 source

    vtkSmartPointer<vtkPolyDataMapper> mapper =
        vtkSmartPointer<vtkPolyDataMapper>::New();          // PolyData mapper 생성
    mapper->SetInputConnection(arrow->GetOutputPort());  // Mapper에 PolyData 연결

    vtkSmartPointer<vtkActor> actor =
        vtkSmartPointer<vtkActor>::New();                       // Actor 생성
    actor->SetMapper(mapper);                                   // Actor에 Mapper 연결

    actor->GetProperty()->SetColor( 0, 1, 0 );  // 색상 설정
    actor->GetProperty()->SetOpacity( 0.5 );        // 불투명도 설정 0.0:투명 ~ 1.0:불투명
    actor->GetProperty()->SetPointSize( 1.0 );  // Vertex 사이즈 설정
    actor->GetProperty()->SetLineWidth( 1.0 );  // Line 두께 설정

    // VTK_POINTS, VTK_WIREFRAME, VTK_SURFACE
    actor->GetProperty()->SetRepresentation(VTK_SURFACE); // 그리기 방법 설정
    //actor->GetProperty()->SetTexture(pTexture);// vtkTexture (Texture Mapping)
    actor->GetProperty()->BackfaceCullingOn();   // Culling On/Off
    actor->GetProperty()->LightingOn();  // Lighting On/Off
    actor->GetProperty()->ShadingOn();   // Shading On/Off

    ////////////////////////////////////////////////////////////////////////
    // Visualize
    vtkSmartPointer<vtkRenderer> renderer =
        vtkSmartPointer<vtkRenderer>::New();
```

Level-Of-Detail Actor

- Lower-resolution model
  - point cloud (sampled from original data)

```
vtkLODActor* partActor = vtkLODActor::New();
partActor->SetMapper( partMapper );
partActor->SetNumberOfCloudPoints( 1000 );
```

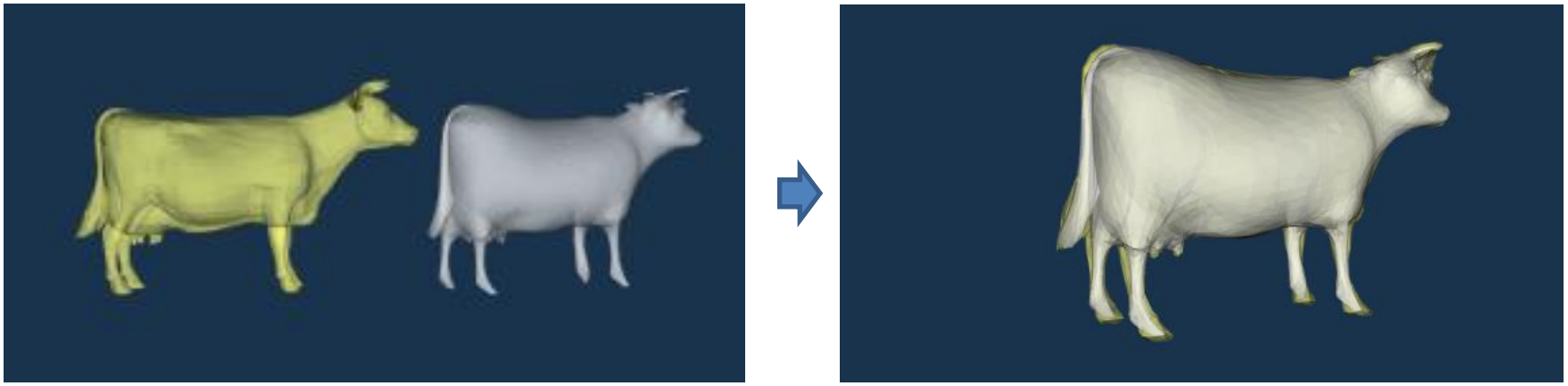
- Setting the desired frame rate in the rendering window

```
vtkRenderWindow->SetDesiredUpdateRate( 20.0 );
```

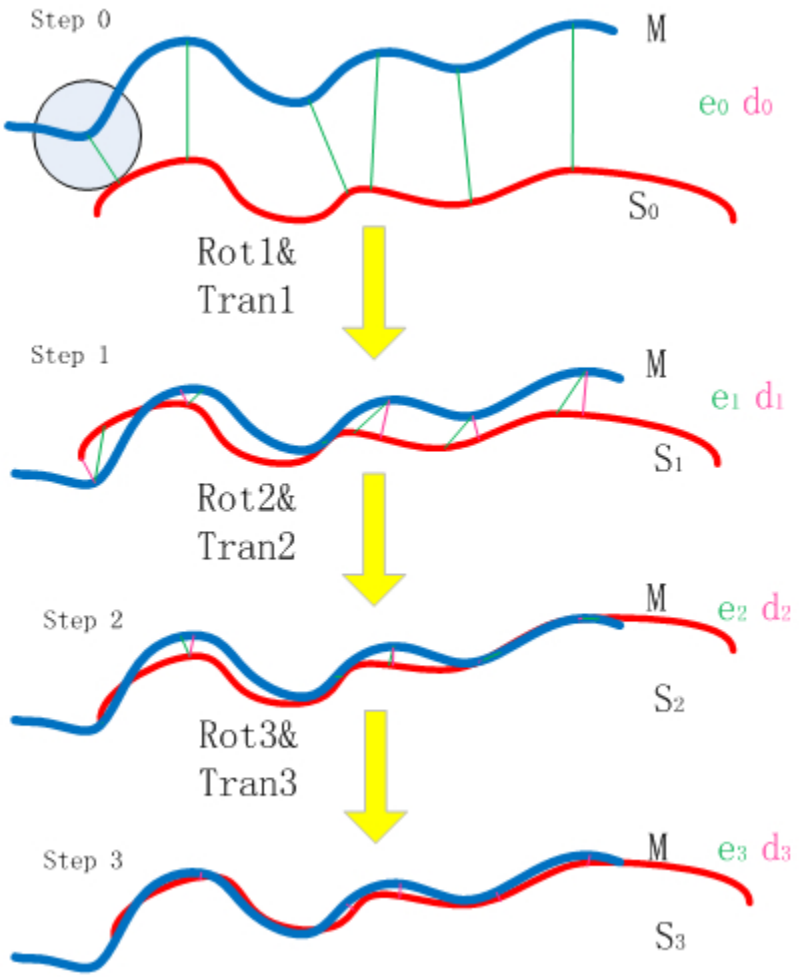
→ vtkLODActor will automatically select the appropriate LOD to yield the requested rate

c.f) vtkLODProp3D: both volume rendering and surface rendering

ICP(Iterative Closest Points) Registration



Iterative Closest Point algorithm-point cloud/mesh registration

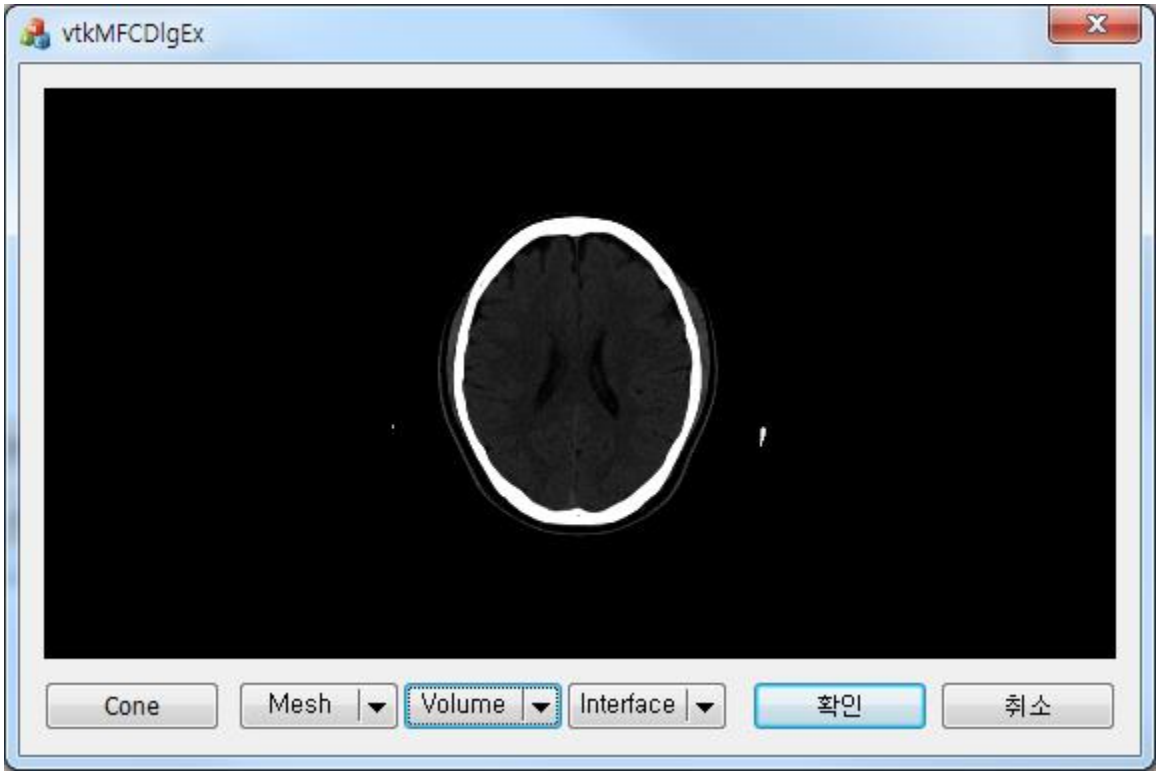


https://taylorwang.wordpress.com/2012/04/06/iterative-closest-point-algorithm-point-cloudmesh-registration/

```
// STL 파일 읽어오기
vtkSmartPointer<vtkSTLReader> stlReader1 =
    vtkSmartPointer<vtkSTLReader>::New();
stlReader1->SetFileName( "../data/example.stl" );
stlReader1->Update();

vtkSmartPointer<vtkSTLReader> stlReader2 =
    vtkSmartPointer<vtkSTLReader>::New();
stlReader2->SetFileName( "../data/example_smooth_transform.stl" );
stlReader2->Update();

// Create a mapper and actor
vtkSmartPointer<vtkPolyDataMapper> mapper1 =
    vtkSmartPointer<vtkPolyDataMapper>::New();
mapper1->SetInputConnection( stlReader1->GetOutputPort() );
vtkSmartPointer<vtkActor> actor1 =
    vtkSmartPointer<vtkActor>::New();
actor1->SetMapper( mapper1 );
actor1->GetProperty()->SetColor( 1.0, 1.0, 0.5 );
actor1->GetProperty()->SetOpacity( 0.5 );

vtkSmartPointer<vtkPolyDataMapper> mapper2 =
    vtkSmartPointer<vtkPolyDataMapper>::New();
mapper2->SetInputConnection( stlReader2->GetOutputPort() );
vtkSmartPointer<vtkActor> actor2 =
    vtkSmartPointer<vtkActor>::New();
actor2->SetMapper( mapper2 );
actor2->GetProperty()->SetOpacity( 0.5 );
```

```
////////////////////////////////////////////////////////////////////////////
// Visualize
vtkSmartPointer<vtkRenderer> renderer =
    vtkSmartPointer<vtkRenderer>::New();
renderer->AddActor( actor1 );
renderer->AddActor( actor2 );
renderer->SetBackground( .1, .2, .3 );
renderer->ResetCamera();

//rendering
m_vtkWindow->AddRenderer( renderer );
m_vtkWindow->Render();

// 1초 기다림
Sleep( 1000 );

// ICP 정합
vtkSmartPointer<vtkIterativeClosestPointTransform> ICP =
    vtkSmartPointer<vtkIterativeClosestPointTransform>::New();
ICP->SetSource( stlReader1->GetOutput() );          // sorce데이터 설정 (vtkPolydata)
ICP->SetTarget( stlReader2->GetOutput() );          // target데이터 설정(vtkPolydata)
ICP->GetLandmarkTransform()->SetModeToRigidBody();// 강체 변환
ICP->SetMaximumNumberOfIterations( 100 );       // 최대 100번 반복루프 (default:50번)
ICP->SetMaximumNumberOfLandmarks( 50 );     // 매칭 점 개수 설정 (default:200개)
ICP->Update();

actor1->SetUserTransform( ICP );        // 변환 transform 설정
m_vtkWindow->Render();
```

DICOM image reader & viewer



```
void CvtkMFCDlgExDlg::OnButtonExVtkimagedata()
{
    vtkSmartPointer<vtkDICOMImageReader> dcmReader =
        vtkSmartPointer<vtkDICOMImageReader>::New();
    dcmReader->SetFileName("../data/CT/CT.00002.00020.dcm");
    dcmReader->Update();

    // Visualize
    vtkSmartPointer<vtkImageViewer2> imageViewer =
        vtkSmartPointer<vtkImageViewer2>::New();
    imageViewer->SetInputConnection( dcmReader->GetOutputPort() );
    imageViewer->SetRenderWindow( m_vtkWindow );
    imageViewer->Render();
}
```

Marching Cubes Algorithm

- Lorenson and Cline, 1987
- Creates a constant density surface from a 3D array of data
- Create a triangular mesh that will approximate the iso-surface
- Can use a general purpose graphics pipeline
- Slow for typical medical applications

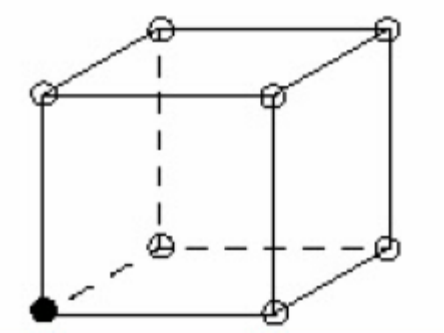
  ← too many polygons
- Drawback: Possible holes in the model

Marching Cubes - Steps

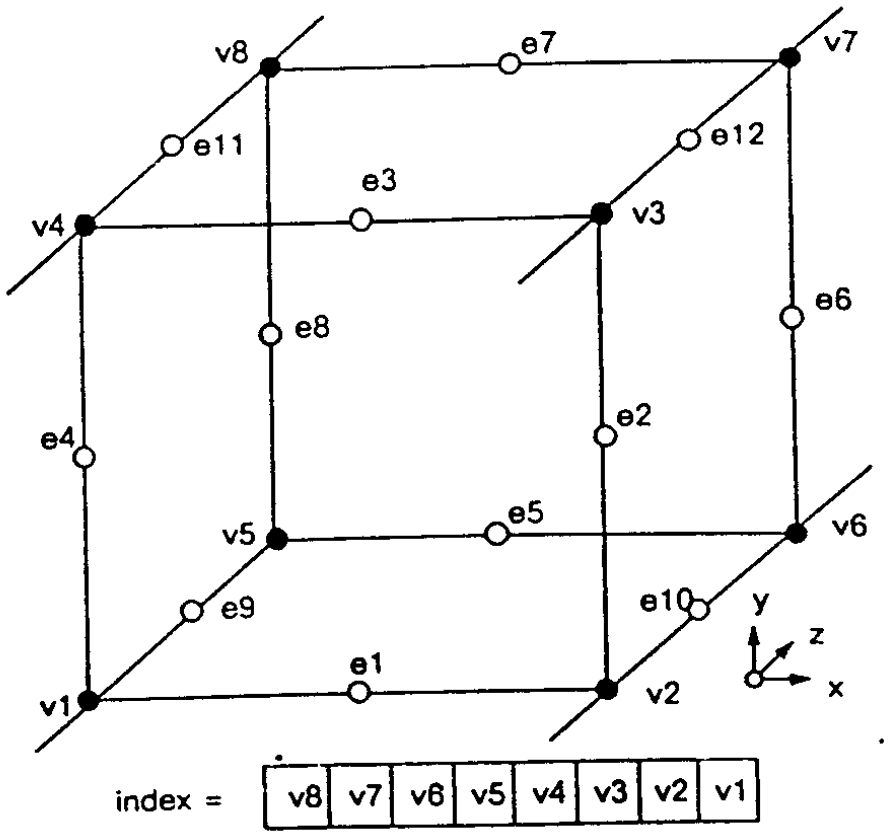
- Create cells (cubes)
- Classify each vertex
- Build an index
- Get edge list based on table look-up
- Interpolate triangle vertices
- Interpolate normals
- Obtain polygon list and do shading in image space

Marching Cubes Algorithm

- For each voxel, set vertex flags
  - Each vertex of the voxel is assigned a binary (0 or 1) value based on whether scalar value is higher or lower than surface value.
  - If vertices are classified as all 0's or all 1's, the algorithm moves to the next voxel.



Set vertex flags

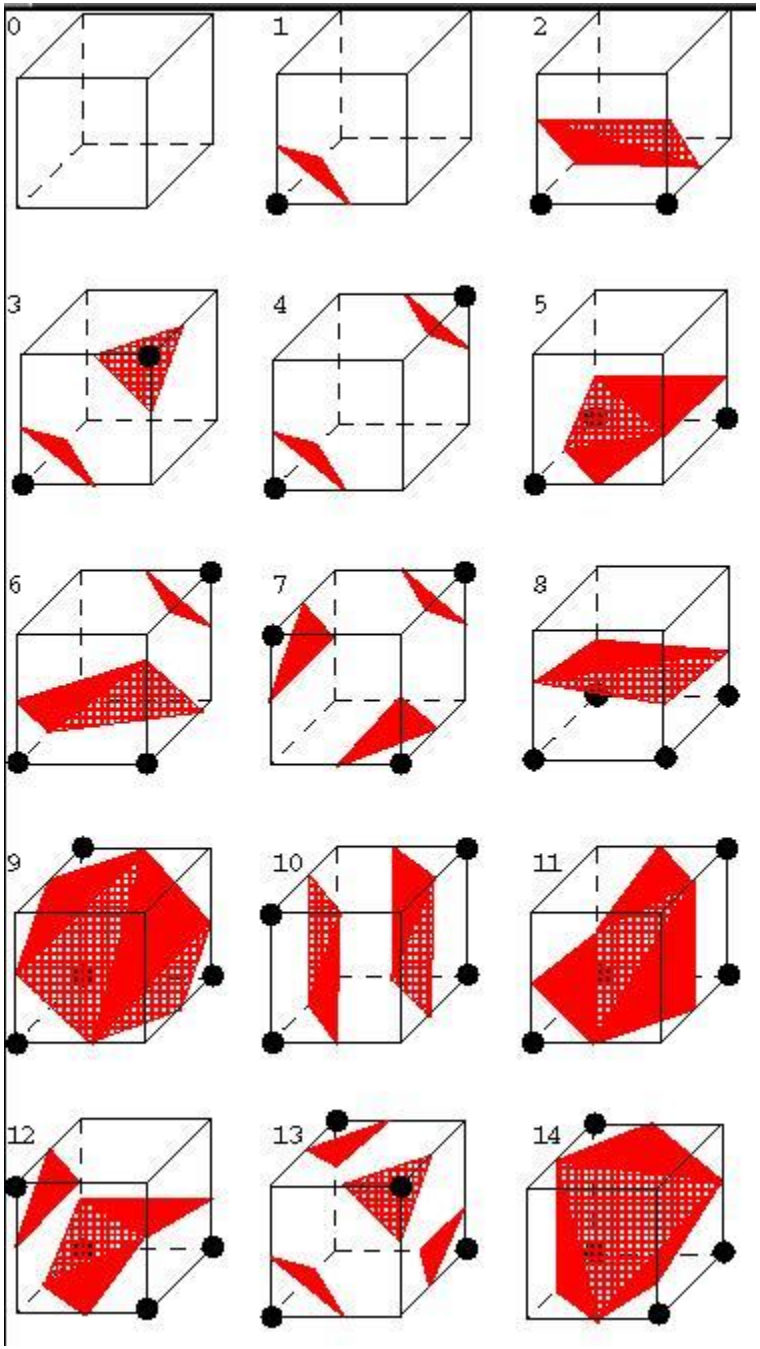


- Classify each vertex of the cube and label 1 or 0 as to whether it lies inside or outside the surface.
- Build an index

Get edge list

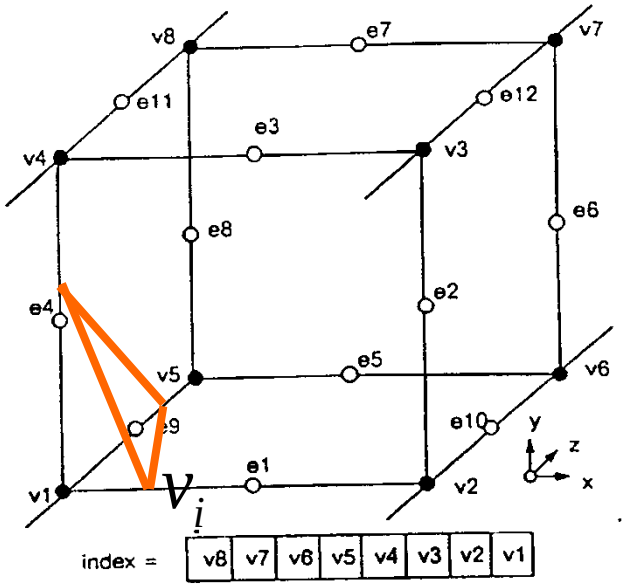
For a given index, edge table returns a list of edges that contains a triangle vertex.

Because rotation symmetry and complementary, 256 cases are reduced to 15 cases.



Surface intersection in a cube

- Create an index for each case

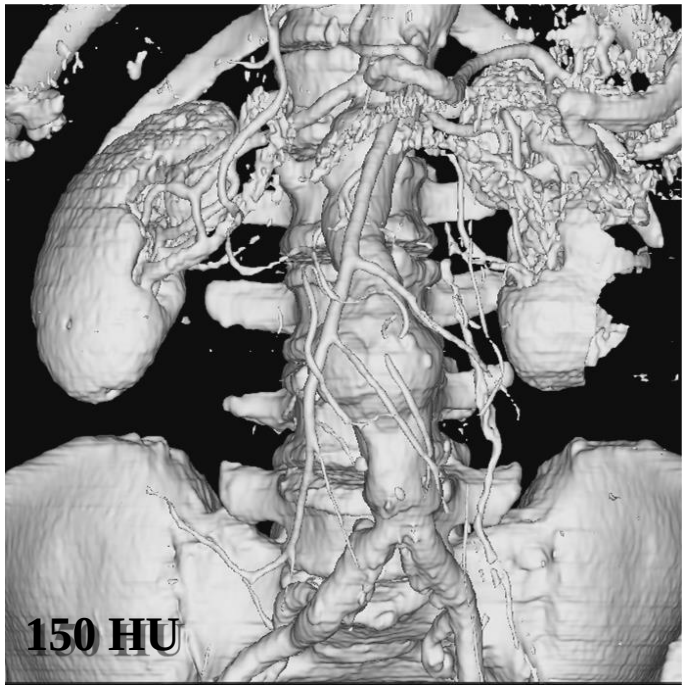


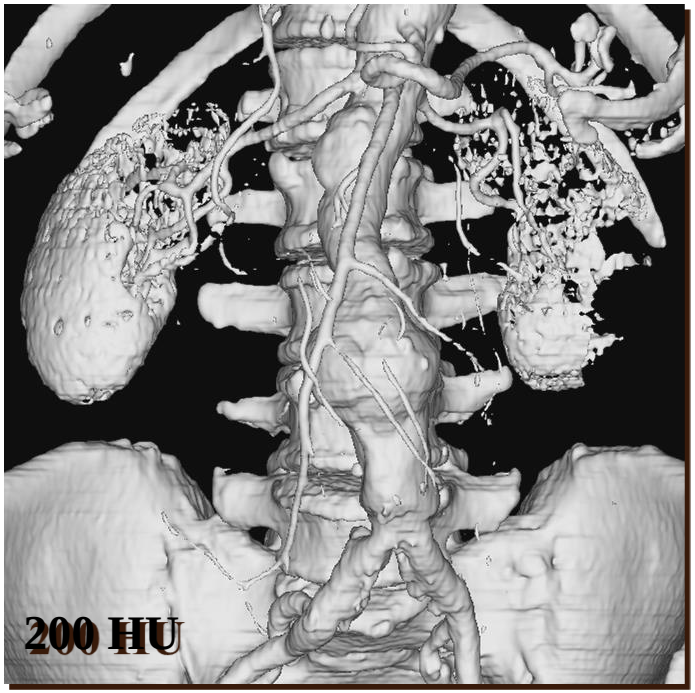
- Interpolate surface intersection along each edge

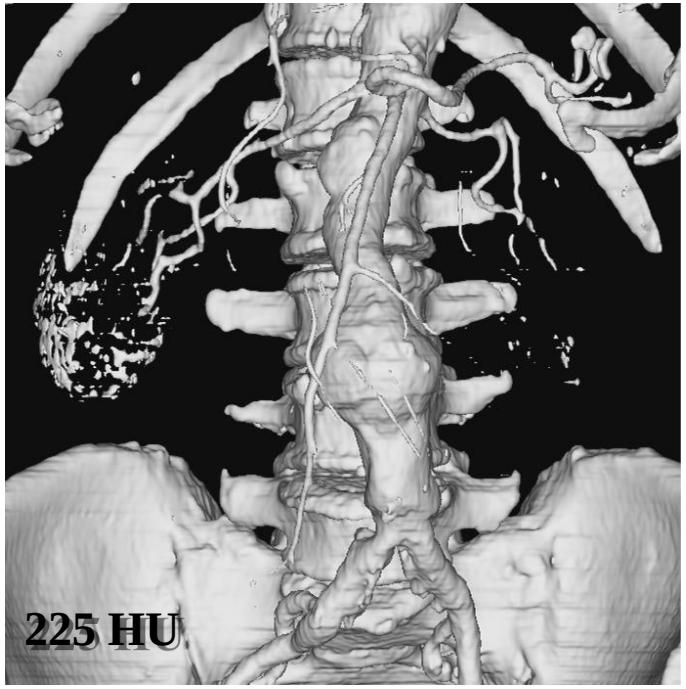
$$v_i = v_1(1-t) + tv_2$$

$$t = \frac{v_1 - v_i}{v_1 - v_2}$$

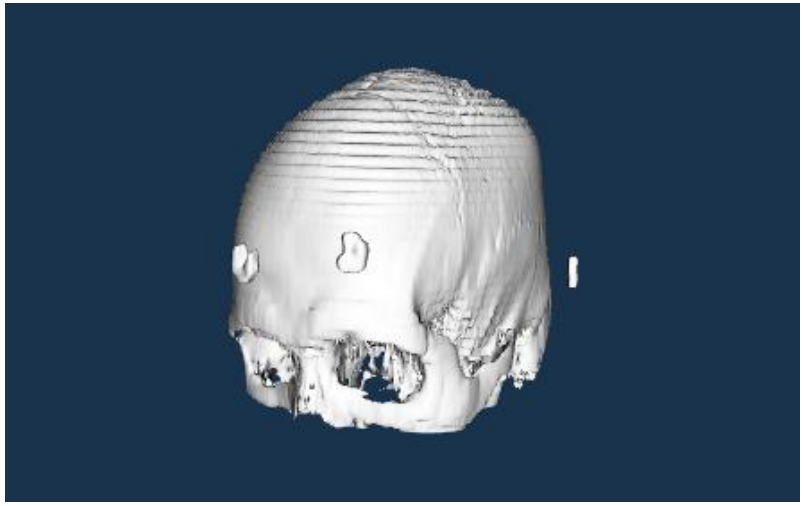
Shaded Surface Display



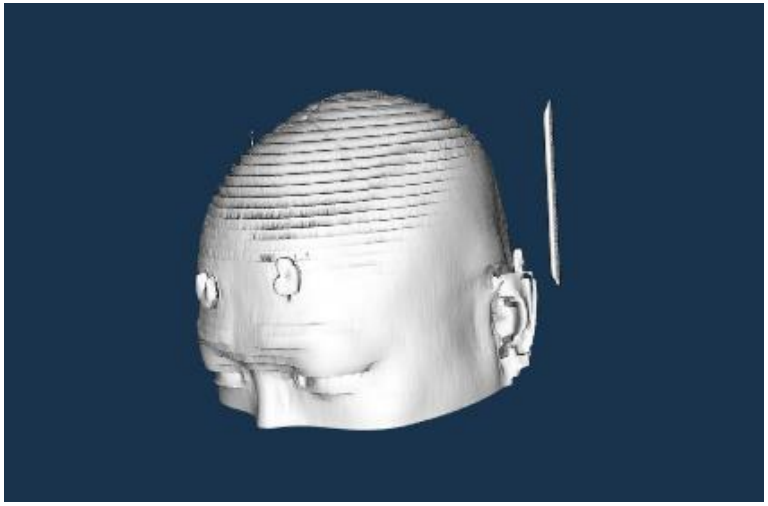




Marching Cubes



Iso-value = 330



Iso-value = -100

```
void CvtkMFCDlgExDlg::OnButtonExMarchingcubes()
{
    vtkSmartPointer<vtkDICOMImageReader> dcmReader =
        vtkSmartPointer<vtkDICOMImageReader>::New();
    // folder에 있는 파일 모두 불러오기 (3차원 vtkImageData)
    dcmReader->SetDirectoryName( "../data/CT" );
    dcmReader->Update();

    ////////////////////////////////////////////////////////////////////////
    // Marching cube filter (vtkImageData → vtkPolyData)
    vtkSmartPointer<vtkMarchingCubes> pMarchingCube =
        vtkSmartPointer<vtkMarchingCubes>::New();
    pMarchingCube->SetInputConnection( dcmReader->GetOutputPort() );
    pMarchingCube->SetValue( 0, 330 );          // 첫번째 iso-value 설정 (id,value)
    pMarchingCube->ComputeScalarsOff();
    pMarchingCube->ComputeNormalsOn();  // Normal을 계산
    pMarchingCube->Update();

    // Create a mapper and actor
    vtkSmartPointer<vtkPolyDataMapper> mapper =
        vtkSmartPointer<vtkPolyDataMapper>::New();
    mapper->SetInputConnection( pMarchingCube->GetOutputPort() );
    vtkSmartPointer<vtkActor> actor =
        vtkSmartPointer<vtkActor>::New();
    actor->SetMapper( mapper );

    ////////////////////////////////////////////////////////////////////////
    // Visualize
    vtkSmartPointer<vtkRenderer> renderer =
        vtkSmartPointer<vtkRenderer>::New();
    renderer->AddActor( actor );
    renderer->SetBackground( .1, .2, .3 );
    renderer->ResetCamera();

    //rendering
```

VTK for Scientific Visualization

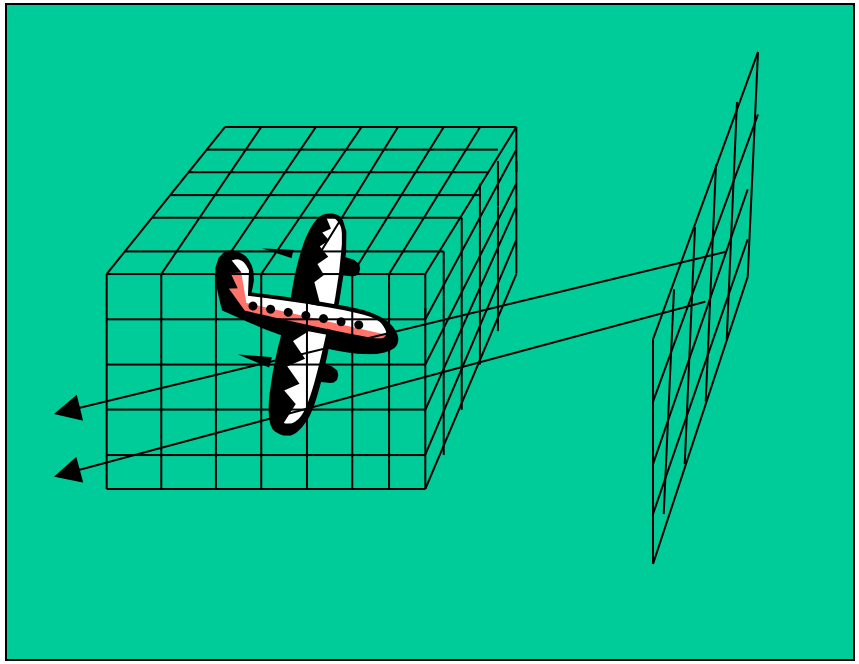
VOLUME RENDERING

Direct Volume Rendering (DVR): Ray-casting

This set of slides references slides used by Prof. Torsten Moeller (Simon Fraser), Prof. Han-Wei Shen (Ohio State).

Basic Idea

Based on the idea of ray tracing

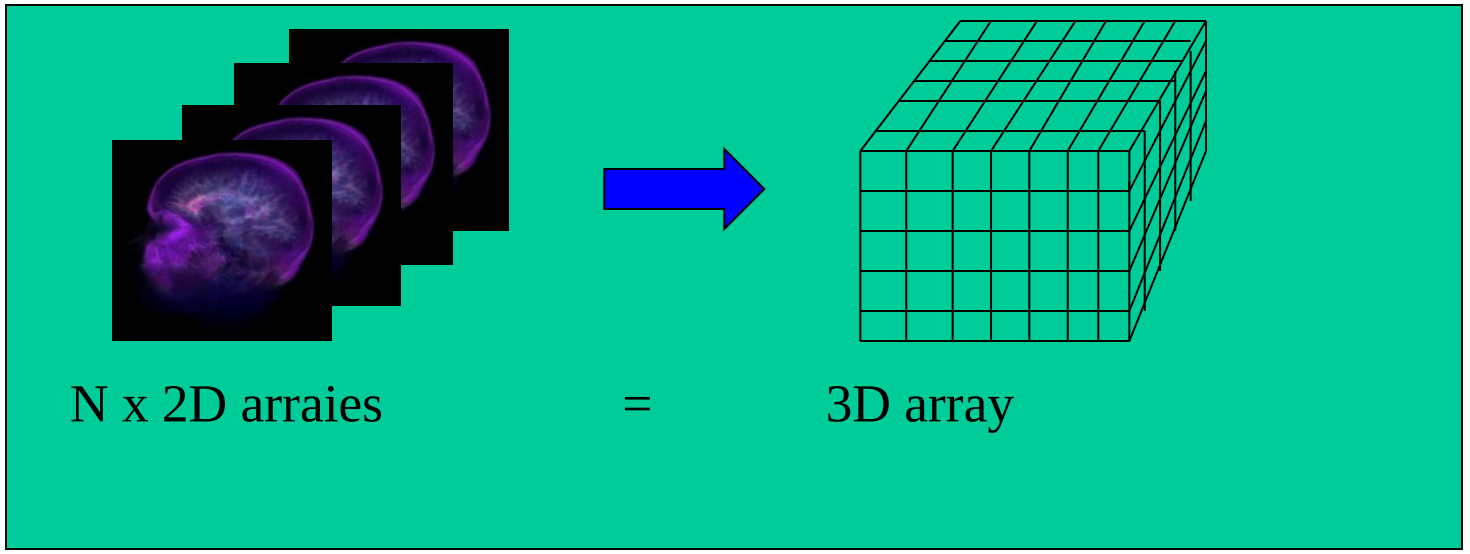


- Trace from eat each pixel as a ray into object space
- Compute color value along the ray
- Assign the value to the pixel

Data Representation

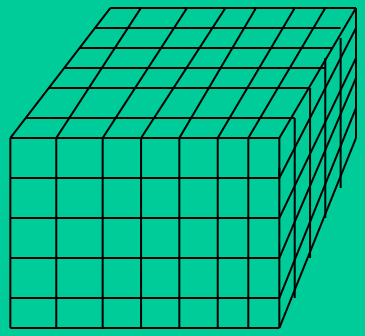
•3D volume data are represented by a finite number of cross sectional slices (hence a 3D raster)

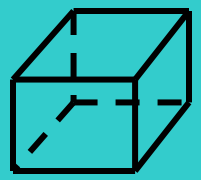
•On each volume element (voxel), stores a data value (if it uses only a single bit, then it is a binary data set. Normally, we see a gray value of 8 to 16 bits on each voxel.)



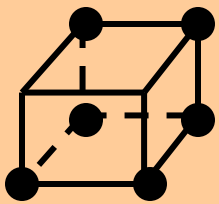
Data Representation (2)

What is a Voxel? – Two definitions





A voxel is a cubic cell, which
has a single value cover
the entire cubic region



A voxel is a data point
at a corner of the cubic cell
The value of a point inside the
cell is determined by interpolation

Ray Casting

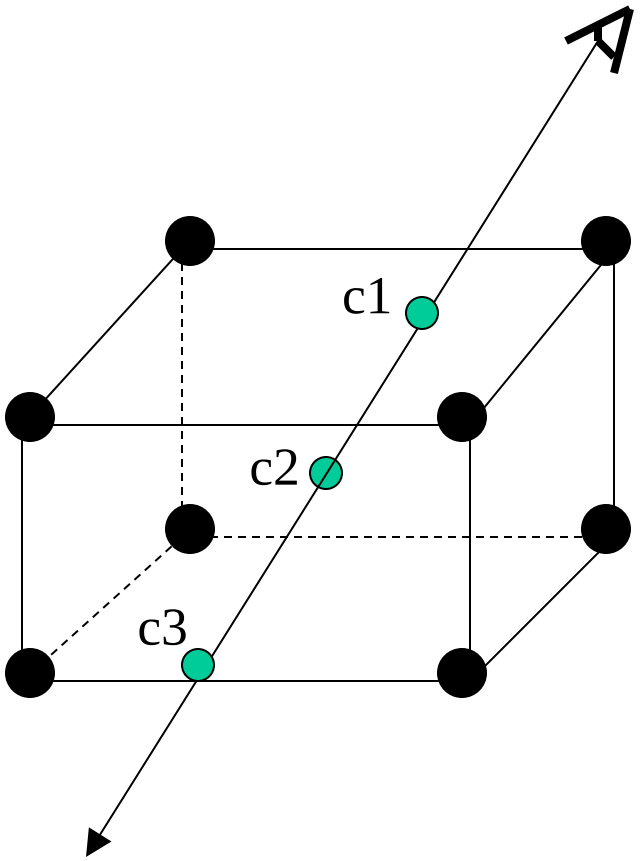
- Stepping through the volume: a ray is cast into the volume, sampling the volume at certain intervals
- The sampling intervals are usually equi-distant, but don’t have to be (e.g. importance sampling)
- At each sampling location, a sample is interpolated / reconstructed from the grid voxels
- popular filters are: nearest neighbor (box), trilinear (tent), Gaussian, cubic spline

Basic Idea of Ray-casting Pipeline

- Data are defined at the corners of each cell (voxel)

- The data value inside the voxel is determined using interpolation (e.g. tri-linear)

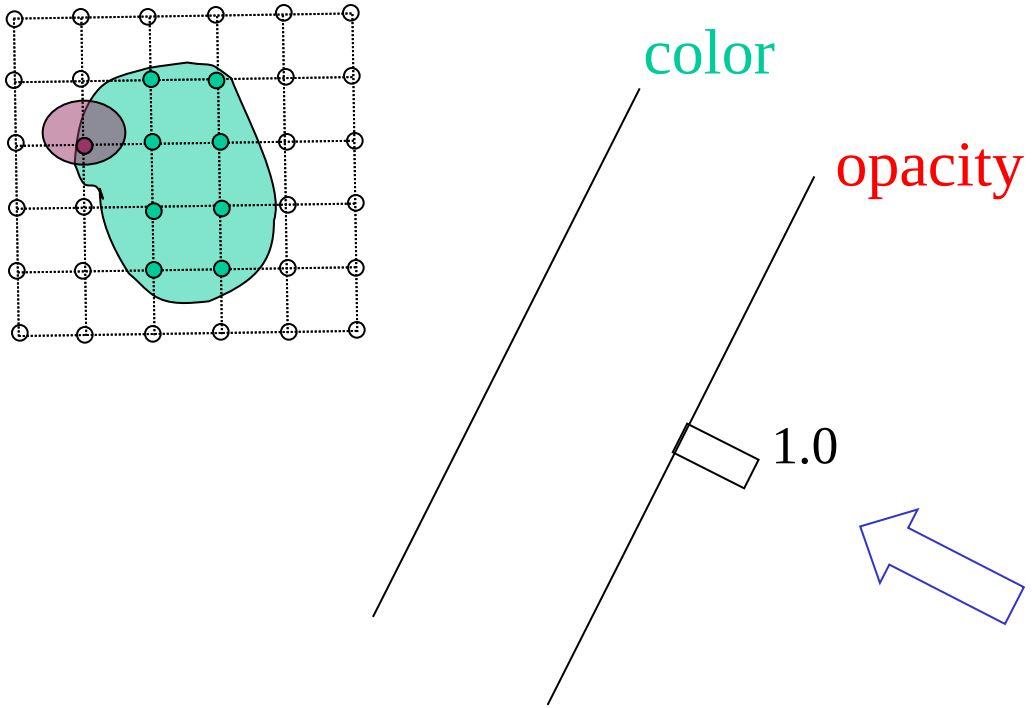
- Composite colors and opacities along the ray path



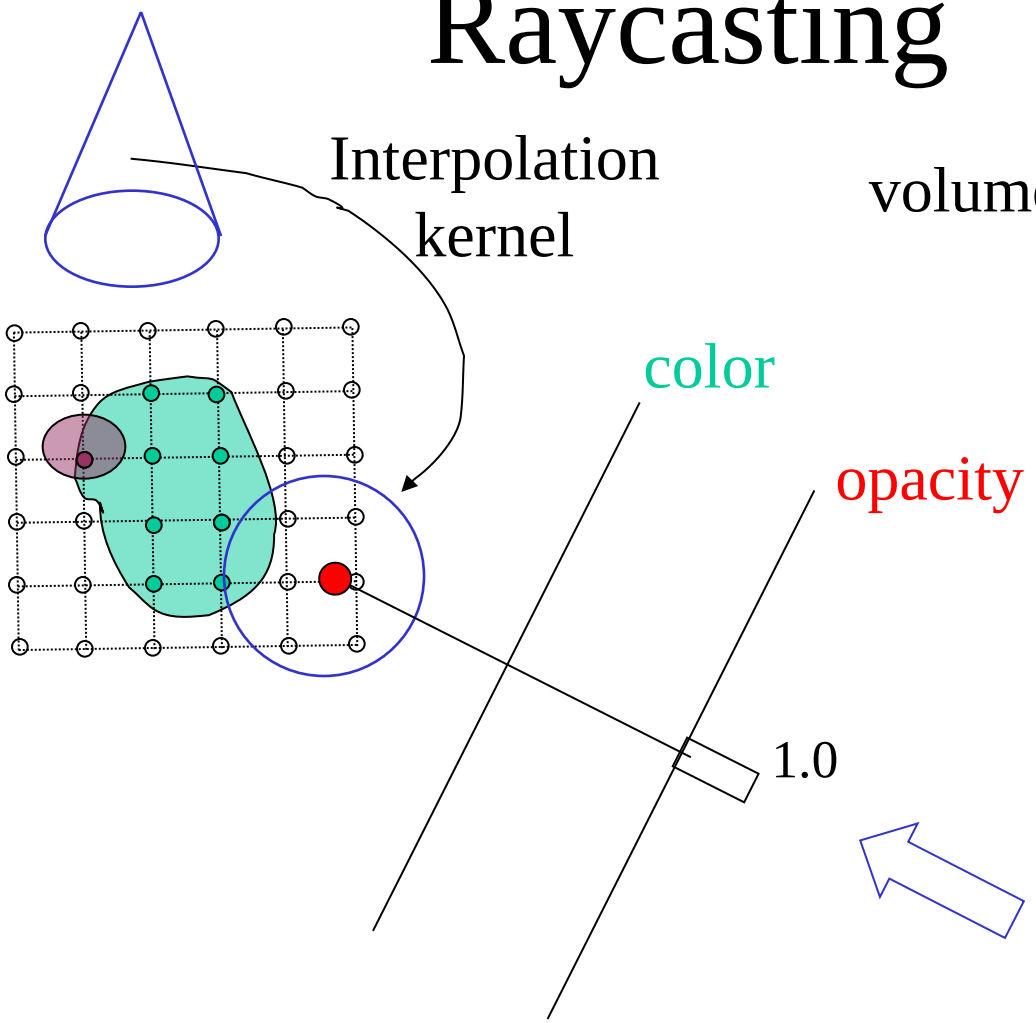
- Can use other ray-traversal schemes as well

Raycasting

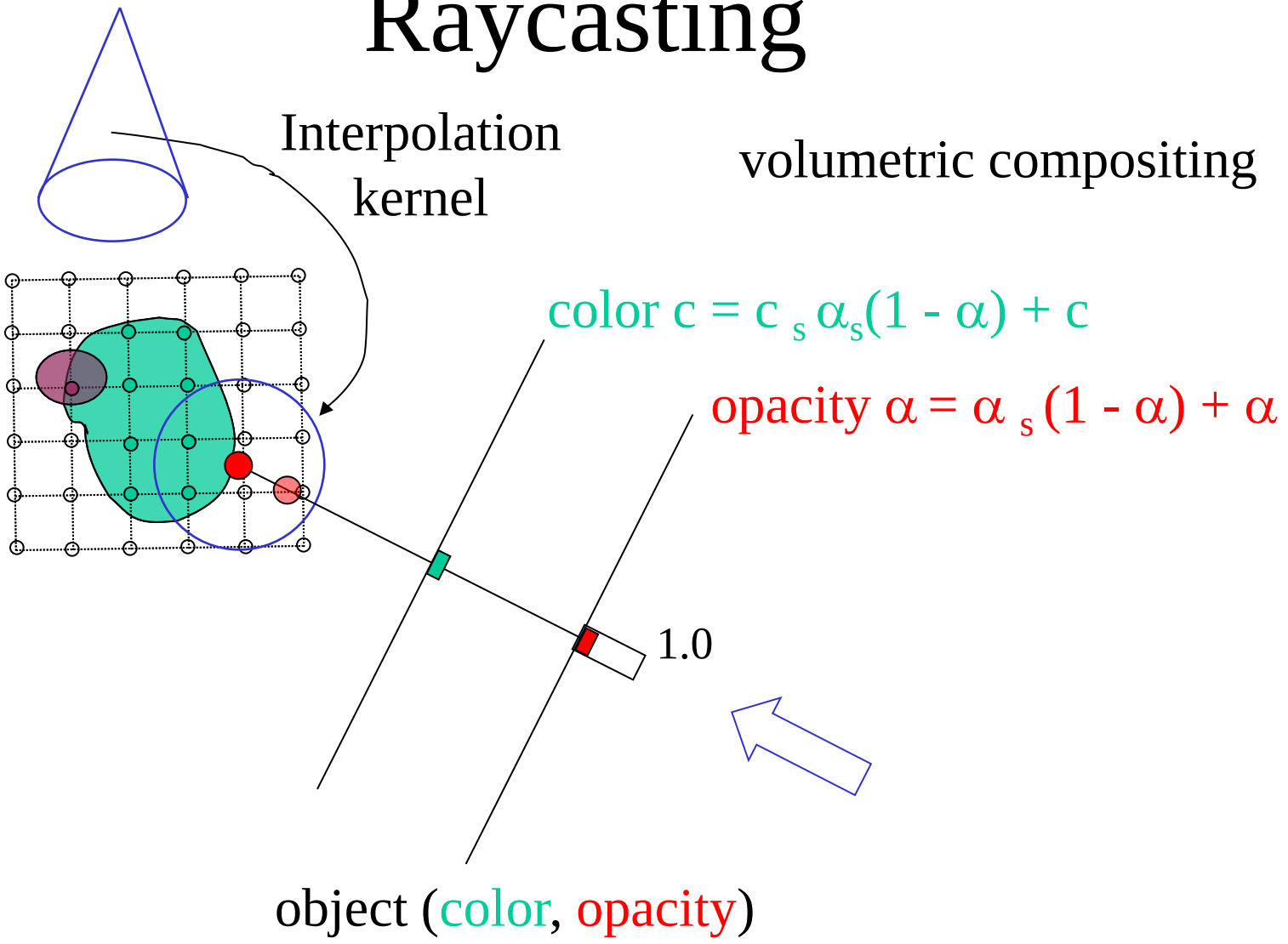
volumetric compositing



Raycasting

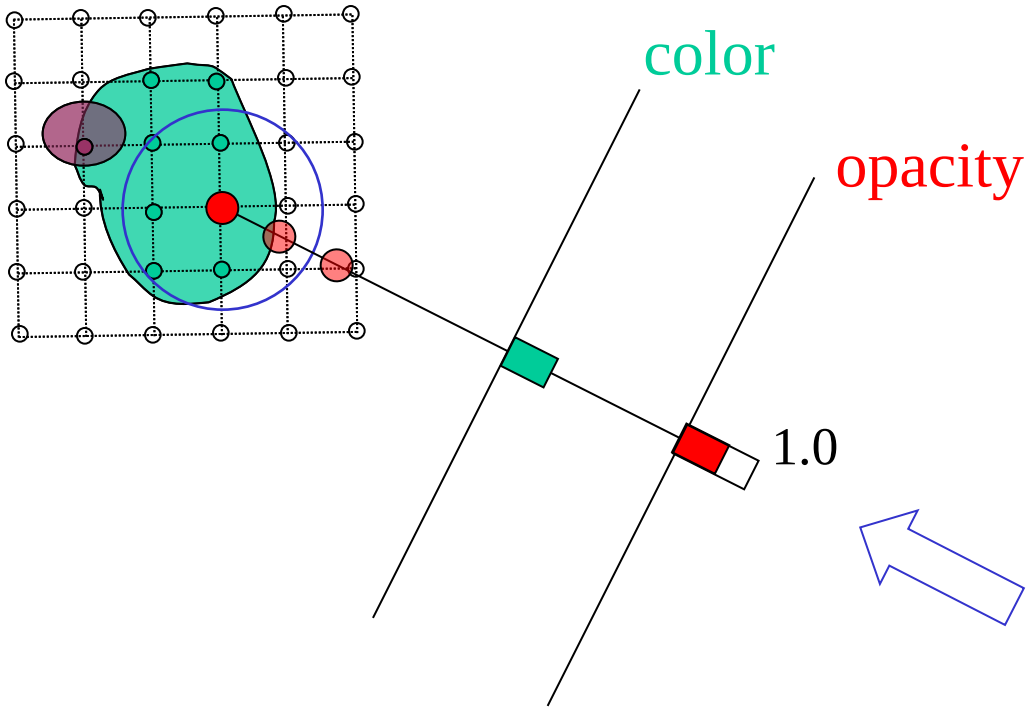

Interpolation
kernel
volumetric compositing
color
opacity
1.0
object (color, opacity)

Raycasting


Interpolation kernel
volumetric compositing
color c = c s αs(1 - α) + c
opacity α = α s (1 - α) + α
1.0
object (color, opacity)

Raycasting

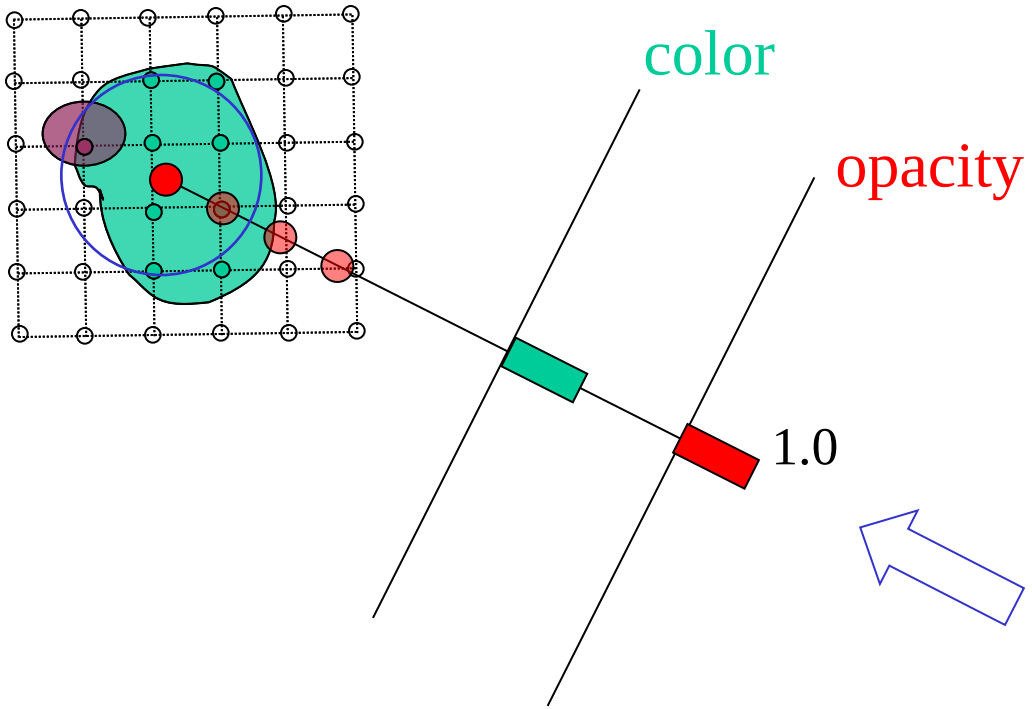
volumetric compositing



object (color, opacity)

Raycasting

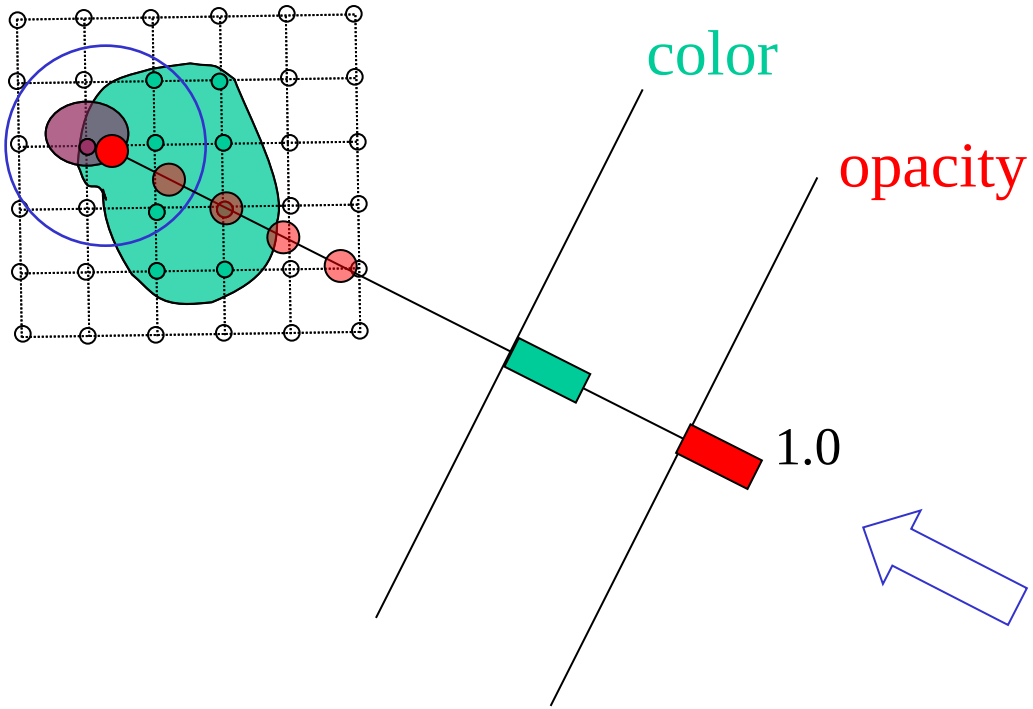
volumetric compositing



object (color, opacity)

Raycasting

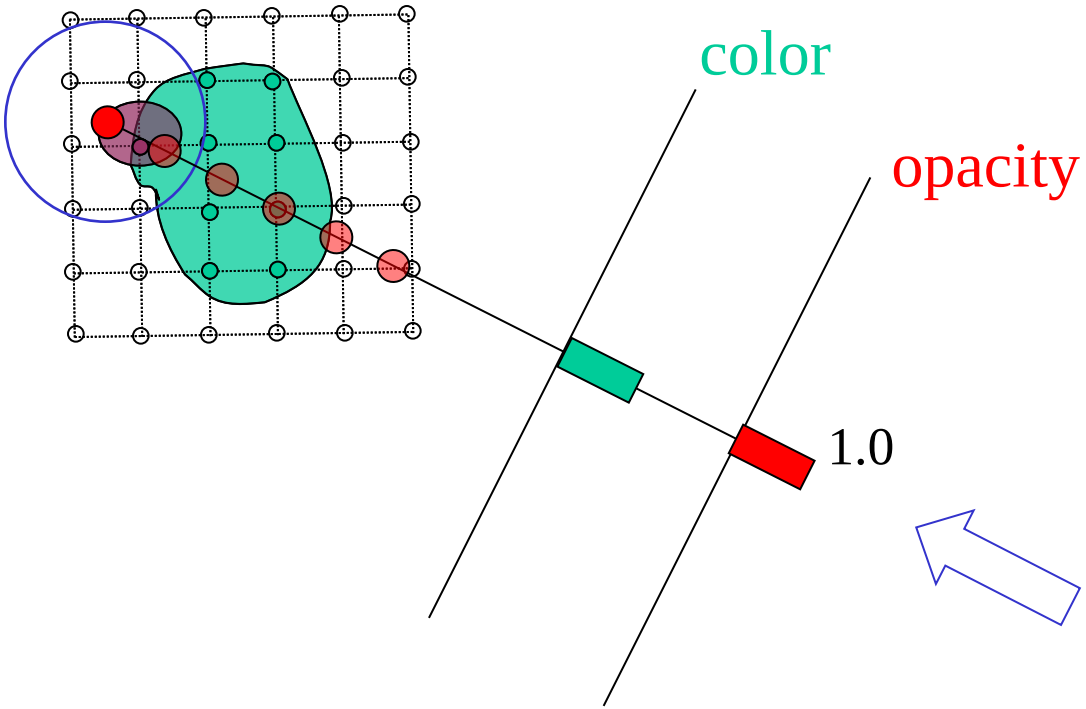
volumetric compositing



object (color, opacity)

Raycasting

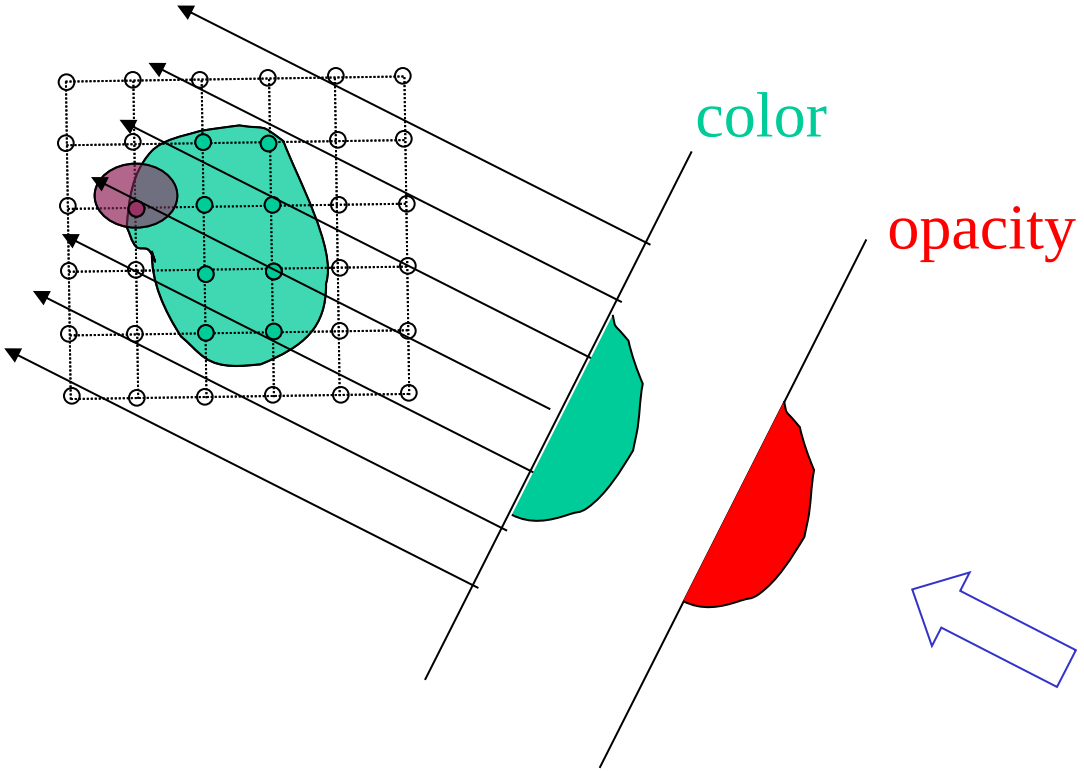
volumetric compositing



object (color, opacity)

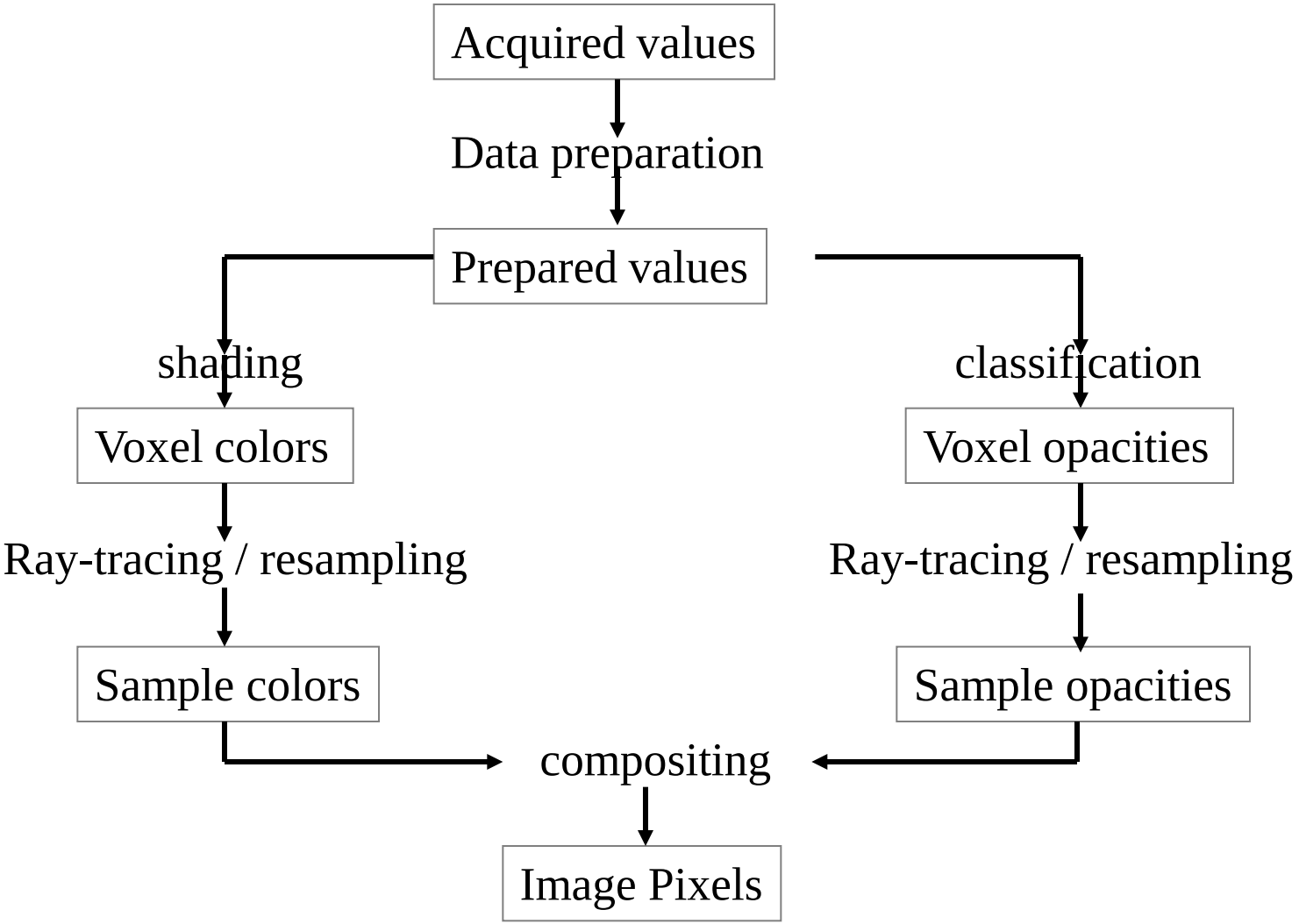
Raycasting

volumetric compositing



object (color, opacity)

Volume Rendering Pipeline



Common Components of General Pipeline

- Interpolation/reconstruction
- Classification or transfer function
- Gradient/normal estimation for shading

Classification/Transfer Function

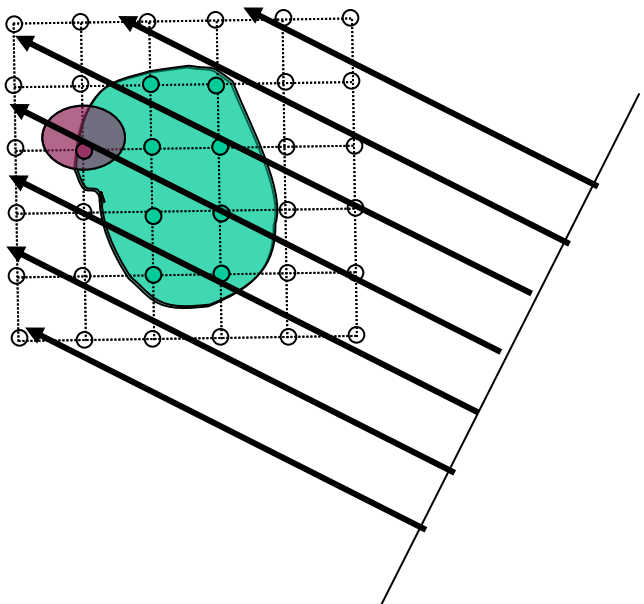
- Maps raw voxel value into presentable entities: color, intensity, opacity, etc.

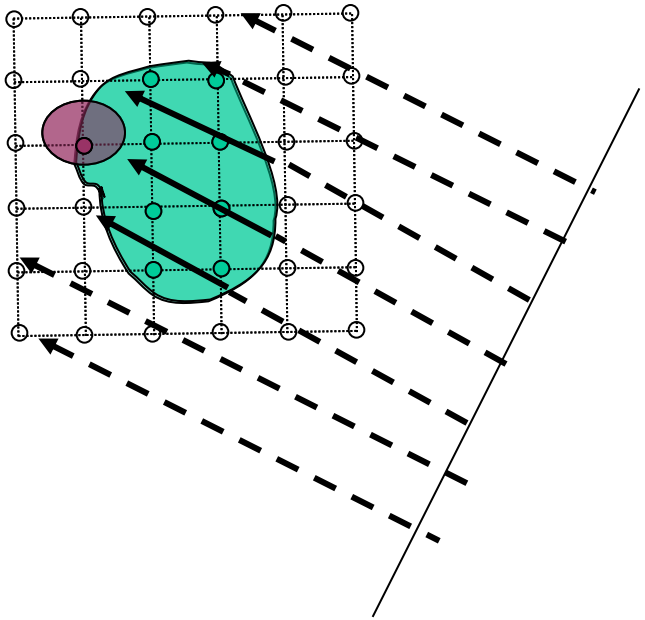
  Raw-data $\rightarrow$ material (R, G, B, a, K_a, K_d, K_s, ...)

- Often use look-up tables (LUT) to store the transfer function that are discovered

Improvements

- Levoy 1990
- front-to-back with early ray termination
  a = 0.95
- hierarchical octree data structure
  - skip empty cells efficiently





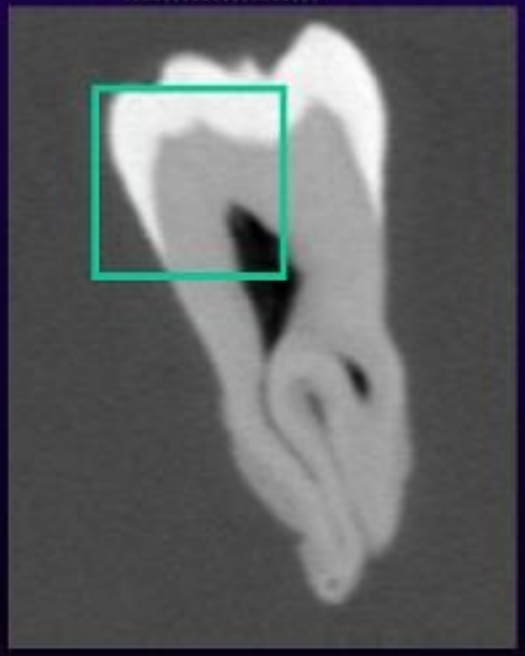
Classification

- Assign opacity (and color) value to each voxel
- Separate voxels into different feature classes by applying a ***transfer function***

Voxel →

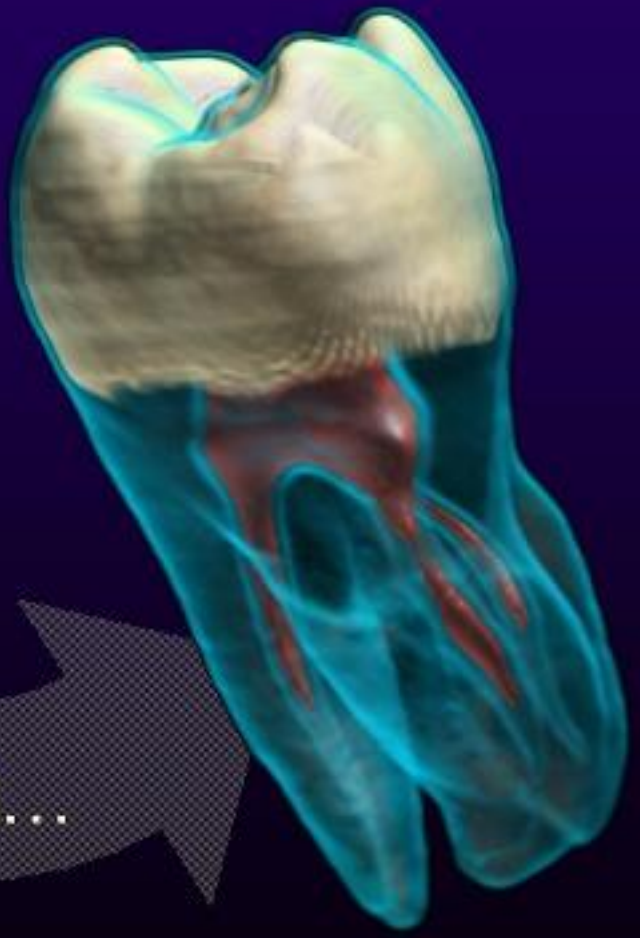
Transfer Functions (TFs)

Map data value *f* to color and opacity

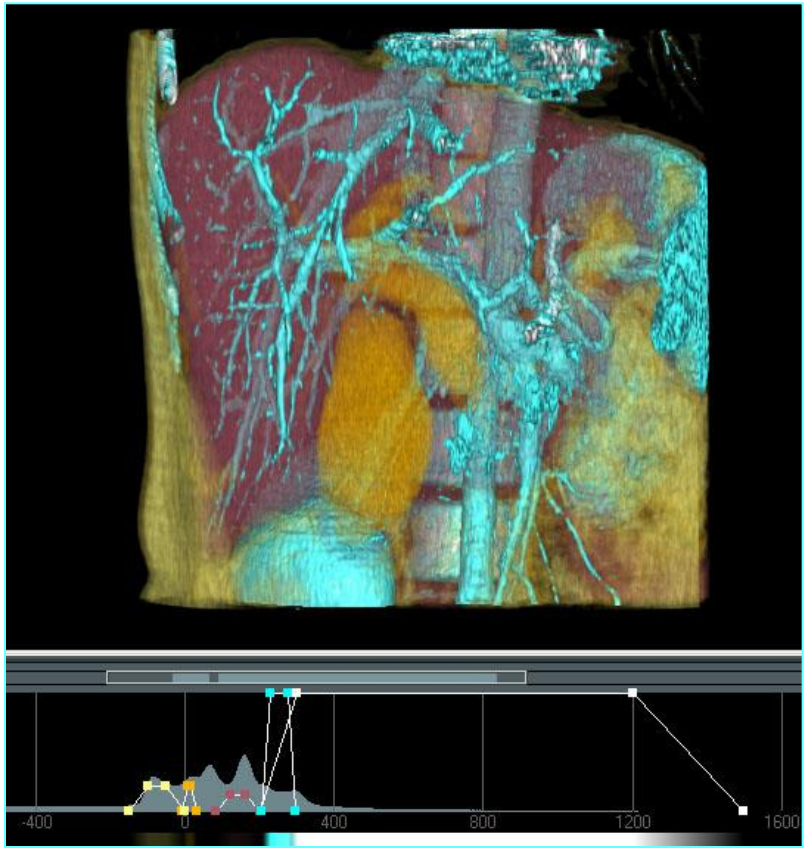


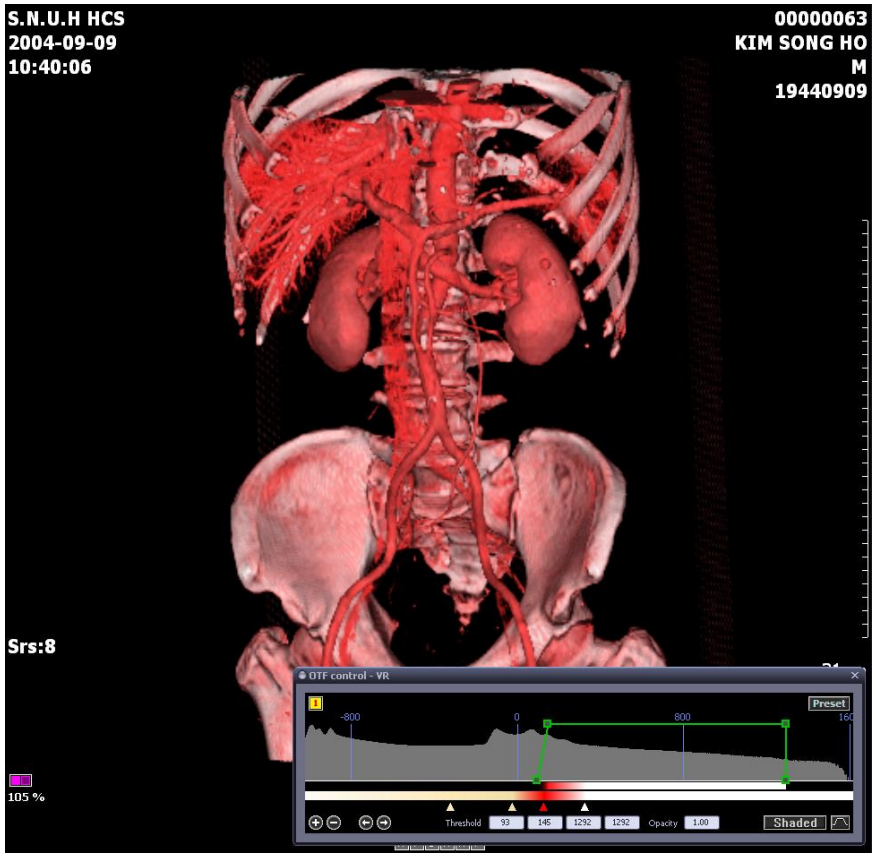
Human Tooth CT

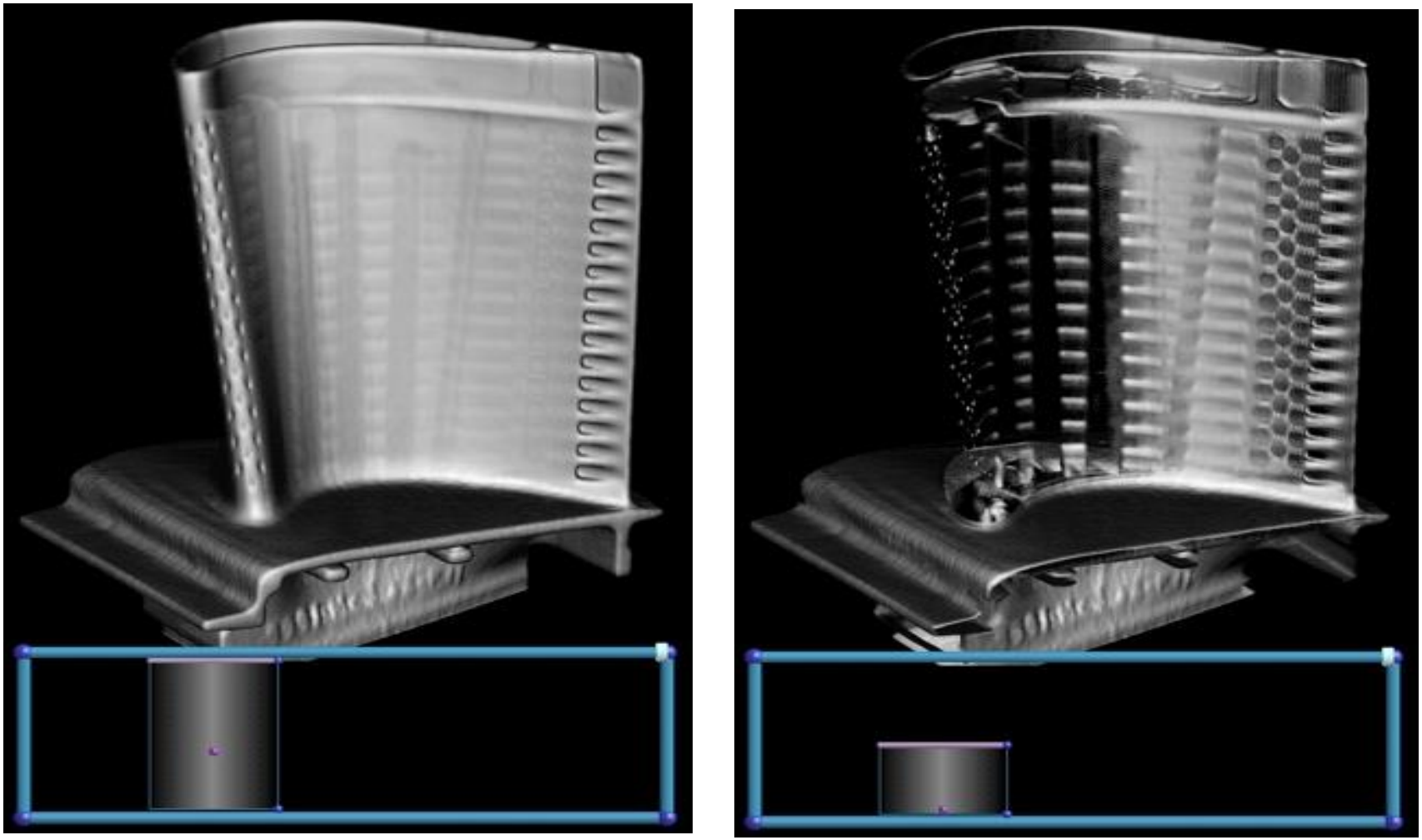
Shading,
Compositing…



Transfer function



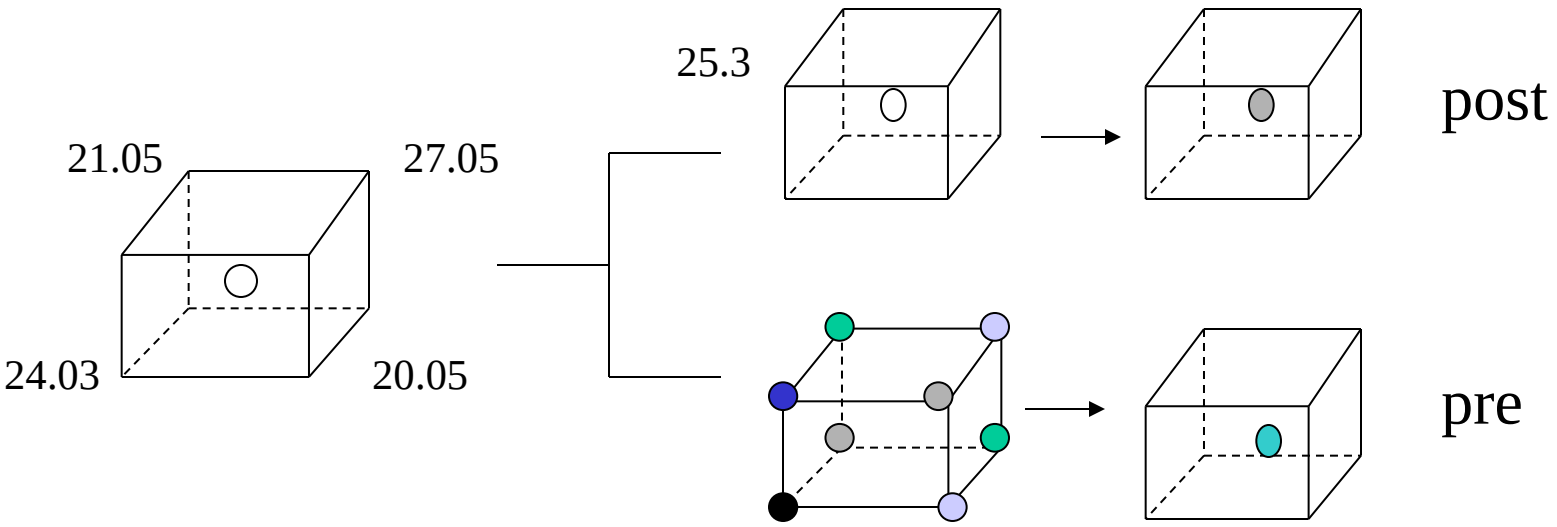




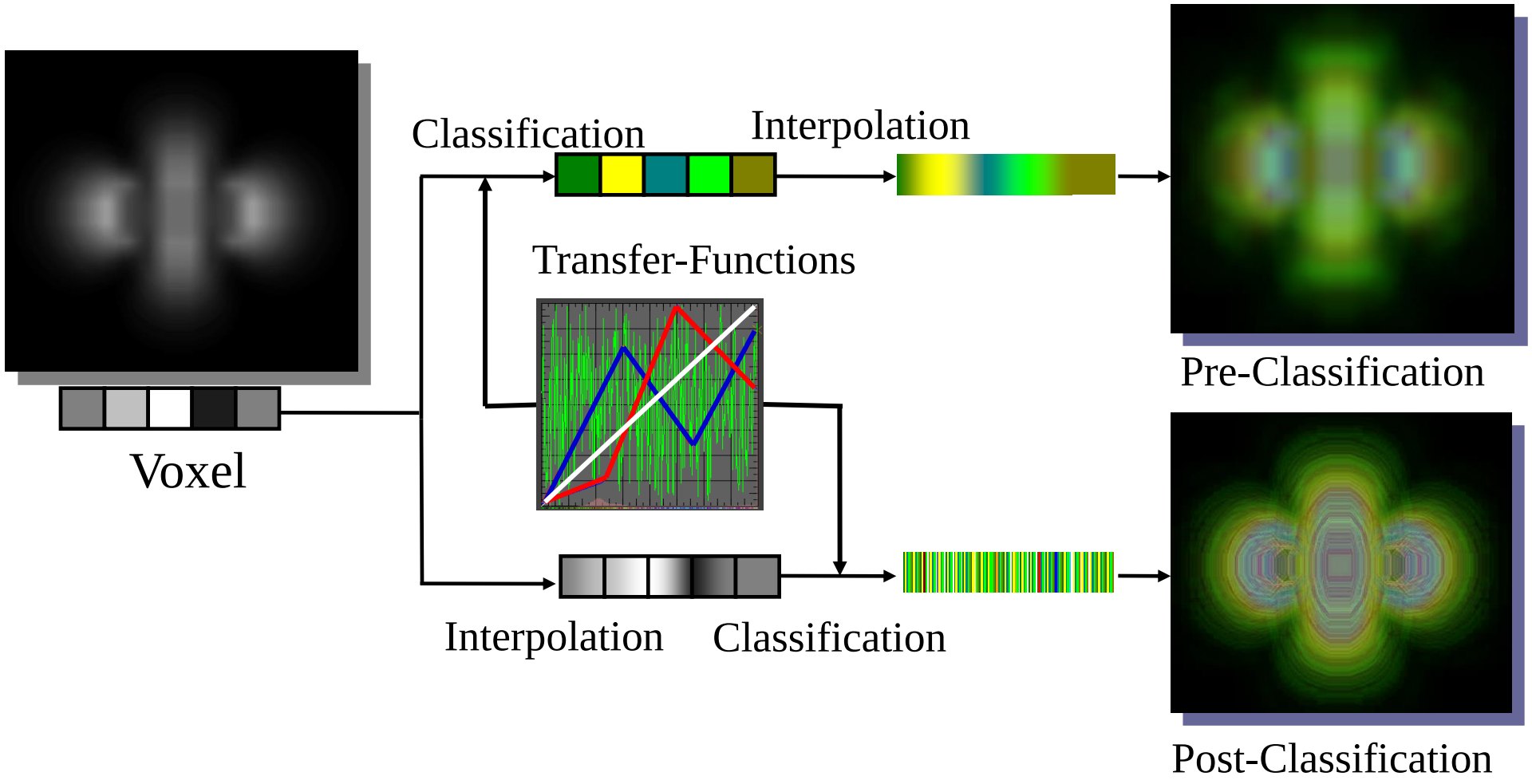
A 1D transfer function (a) is emulated by assigning opacity regardless of gradient magnitude (vertical axis in lower frame). A 2D transfer function (b) giving opacity to only low gradient magnitudes reveals internal structure.

Classification Order

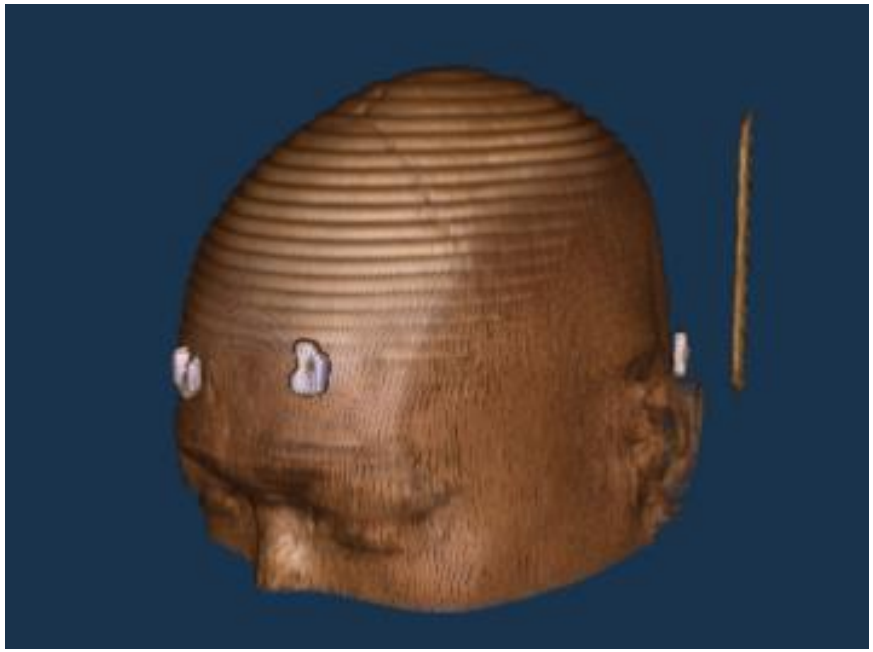
- **Pre classification**: Classification first, and then Filter → tend to excessive blurring when the image resolution exceeds that of the volume
- **Post classification**: filter first, then classification

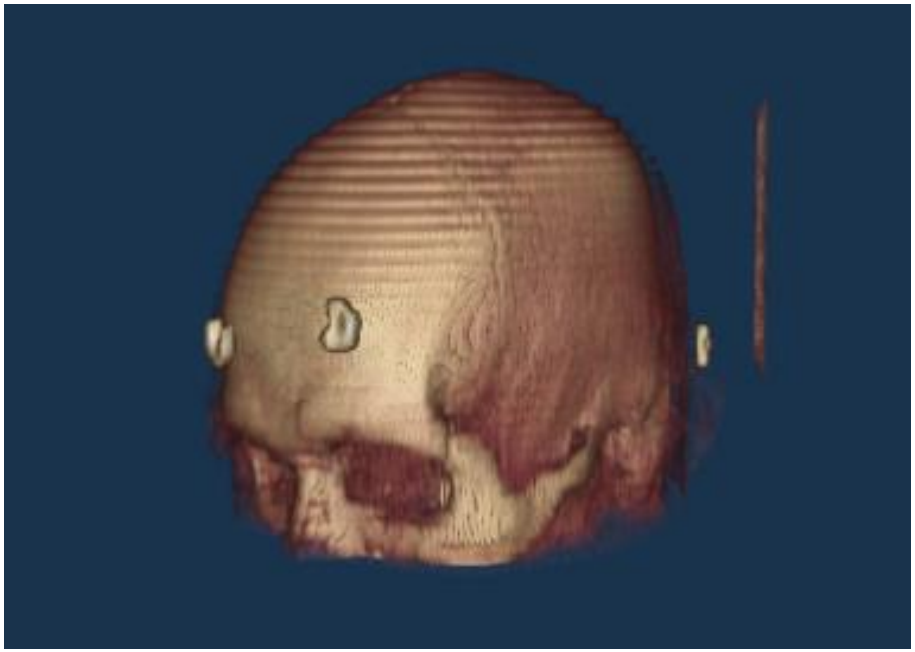


Pre- and Post-Classification

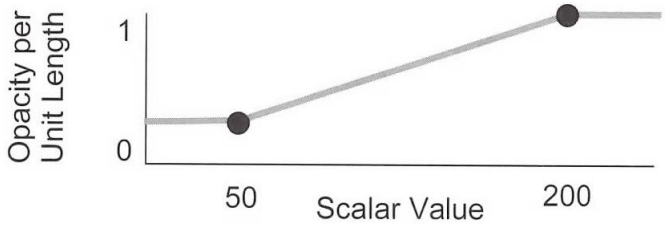


Volume rendering by ray-casting

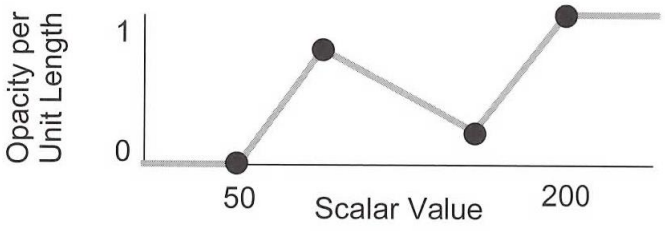




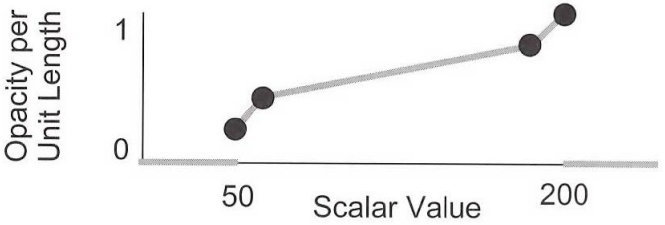
Controlling OTF in VTK




```
vtkPieceWiseFunction* tfun = vtkPiecewiseFunction::New();
    tfun->AddPoint (  50, 0.2 );
    tfun->AddPoint( 200, 1.0 );
```




```
    tfun->RemovePoint( 50 );
    tfun->AddPoint(  50, 0.0 );
    tfun->AddSegment( 100, 0.8, 150, 0.2 );
```




```
    tfun->AddPoint(  50, 0.2 );
    tfun->AddSegment( 60, 0.4, 190, 0.8 );
    tfun->ClampingOff();
```

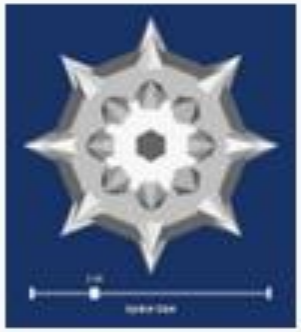
```
vtkColorTransferFunction* ctfun = vtkColorTransferFunction ::New();
    ctfun->AddRGBPoint (     0, 1.0, 0.0, 0.0 );
    ctfun->AddRGBPoint ( 127, 0.0, 1.0, 0.0 );
    ctfun->AddRGBPoint ( 255, 0.0, 0.0, 1.0 );
```

```
void CvtkMFCDlgExDlg::OnButtonExVolumeRendering()
{
    // CT 영상 불러오기
    vtkSmartPointer<vtkDICOMImageReader> dcmReader =
        vtkSmartPointer<vtkDICOMImageReader>::New();
    dcmReader->SetDirectoryName( "../data/CT" );
    dcmReader->Update();

    // Rendering Opacity 값 설정
    vtkSmartPointer<vtkPiecewiseFunction> compositeOpacity =
        vtkSmartPointer<vtkPiecewiseFunction>::New();
    // 1) CT MUSCLE
    compositeOpacity->AddPoint( -3024, 0 );
    compositeOpacity->AddPoint( -155.41, 0 );
    compositeOpacity->AddPoint( 217.64, 0.68 );
    compositeOpacity->AddPoint( 419.74, 0.83 );
    compositeOpacity->AddPoint( 3071, 0.80 );

    // Rendering Color 값 설정
    vtkSmartPointer<vtkColorTransferFunction> color =
        vtkSmartPointer<vtkColorTransferFunction>::New();
    // 1) CT MUSCLE
    color->AddRGBPoint( -3024, 0, 0, 0 );
    color->AddRGBPoint( -155.41, .55, .25, .15 );
    color->AddRGBPoint( 217.64, .88, .60, .29 );
    color->AddRGBPoint( 419.74, 1, .94, .95 );
    color->AddRGBPoint( 3071, .83, .66, 1 );
```

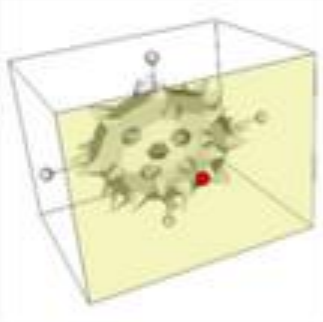
```
    // Smart Volume Mapper 사용
    vtkSmartPointer<vtkSmartVolumeMapper> volumeMapper =
        vtkSmartPointer<vtkSmartVolumeMapper>::New();
    volumeMapper->SetInputConnection( dcmReader->GetOutputPort() );
    volumeMapper->SetBlendModeToComposite();    // 블렌딩 모드 설정 (composite first)

    // Volume 그리기 속성 설정
    vtkSmartPointer<vtkVolumeProperty> volumeProperty =
        vtkSmartPointer<vtkVolumeProperty>::New();
    volumeProperty->ShadeOn();
    volumeProperty->SetInterpolationType( VTK_LINEAR_INTERPOLATION );
    volumeProperty->SetColor( color );
    volumeProperty->SetScalarOpacity( compositeOpacity ); // composite first

    // vtkVolume은 Volume rendering을 위한 Actor역할을 한다
    vtkSmartPointer<vtkVolume> volume =
        vtkSmartPointer<vtkVolume>::New();
    volume->SetMapper( volumeMapper );
    volume->SetProperty( volumeProperty );

    ////////////////////////////////////////////////////////////////////////
    // Visualize
    vtkSmartPointer<vtkRenderer> renderer =
        vtkSmartPointer<vtkRenderer>::New();
    // vtkVolume은 AddActor가 아닌 AddViewProp 함수로 추가
    renderer->AddViewProp( volume );
    renderer->SetBackground( .1, .2, .3 );
    renderer->ResetCamera();

    //rendering
    m_vtkWindow->AddRenderer( renderer );
    m_vtkWindow->Render();
}
```

VTK widgets

3D Slider widget

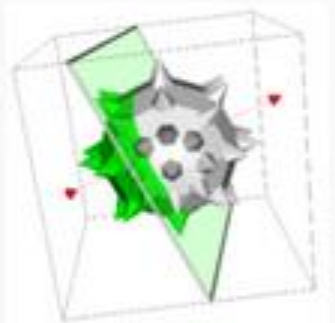


2D Slider widget

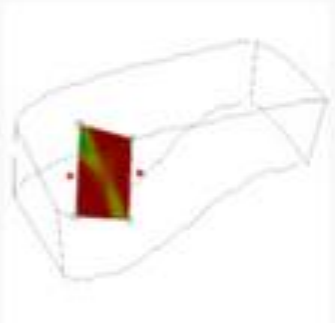


Box widget

Line widget



Implicit plane widget



Plane widget

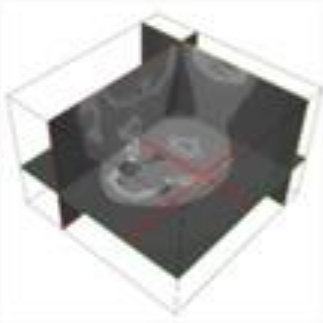


Image plane widget

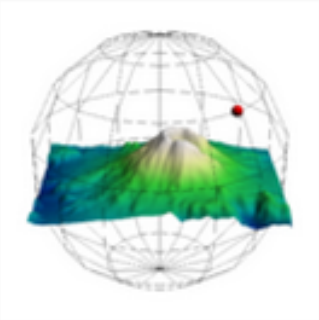
Point (or handle) widget

Implicit plane widget

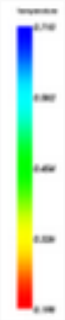
Plane widget

Image plane widget

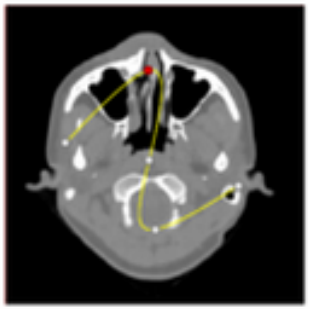
Point (or handle) widget



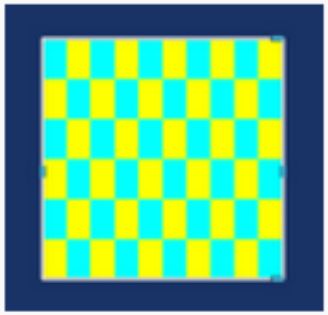
Sphere widget



Scalar bar widget



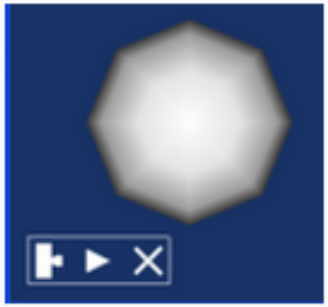
Spline widget



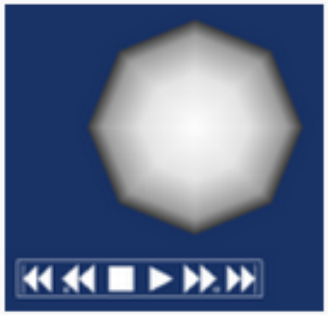
Checkerboard widget

Rectilinear wipe widget

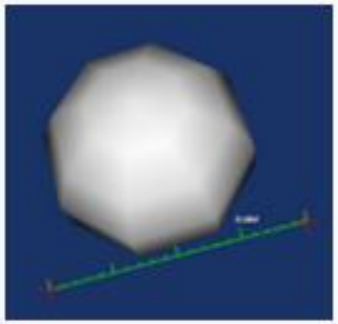
Text widget



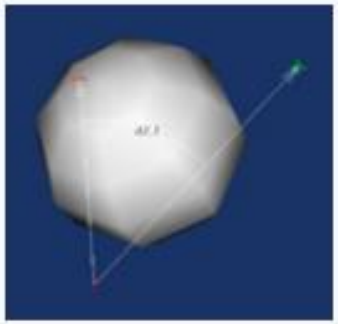
Camera widget



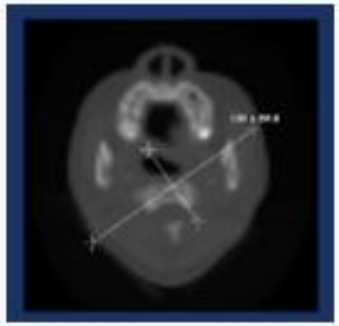
Playback widget



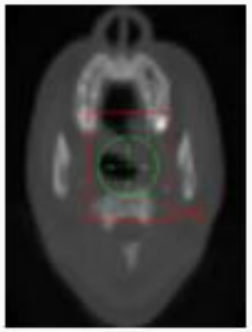
Distance widget



Angle widget

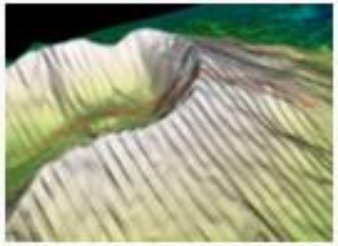


BiDimensional widget



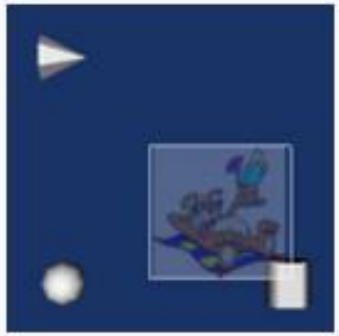
Affine widget

Contour widget

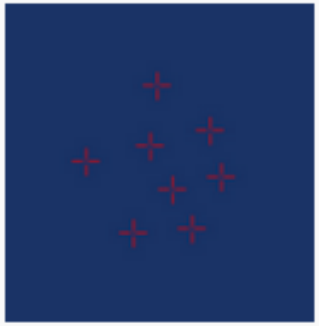


Tracing contours on polygonal surfaces

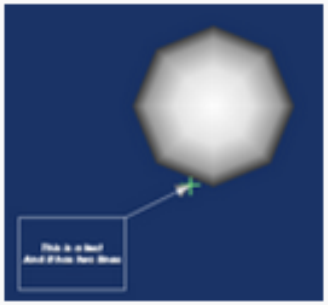
Live wiring with the contour widget



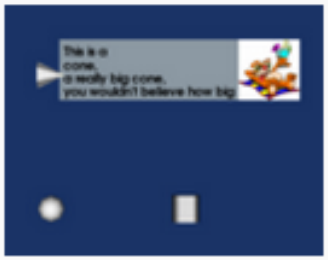
Logo widget



Seed widget

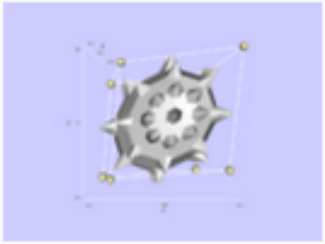


Caption widget

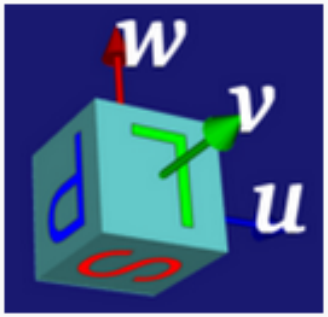


Balloon widget

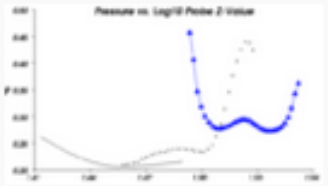
Tensor probe widget



Parallelopiped widget



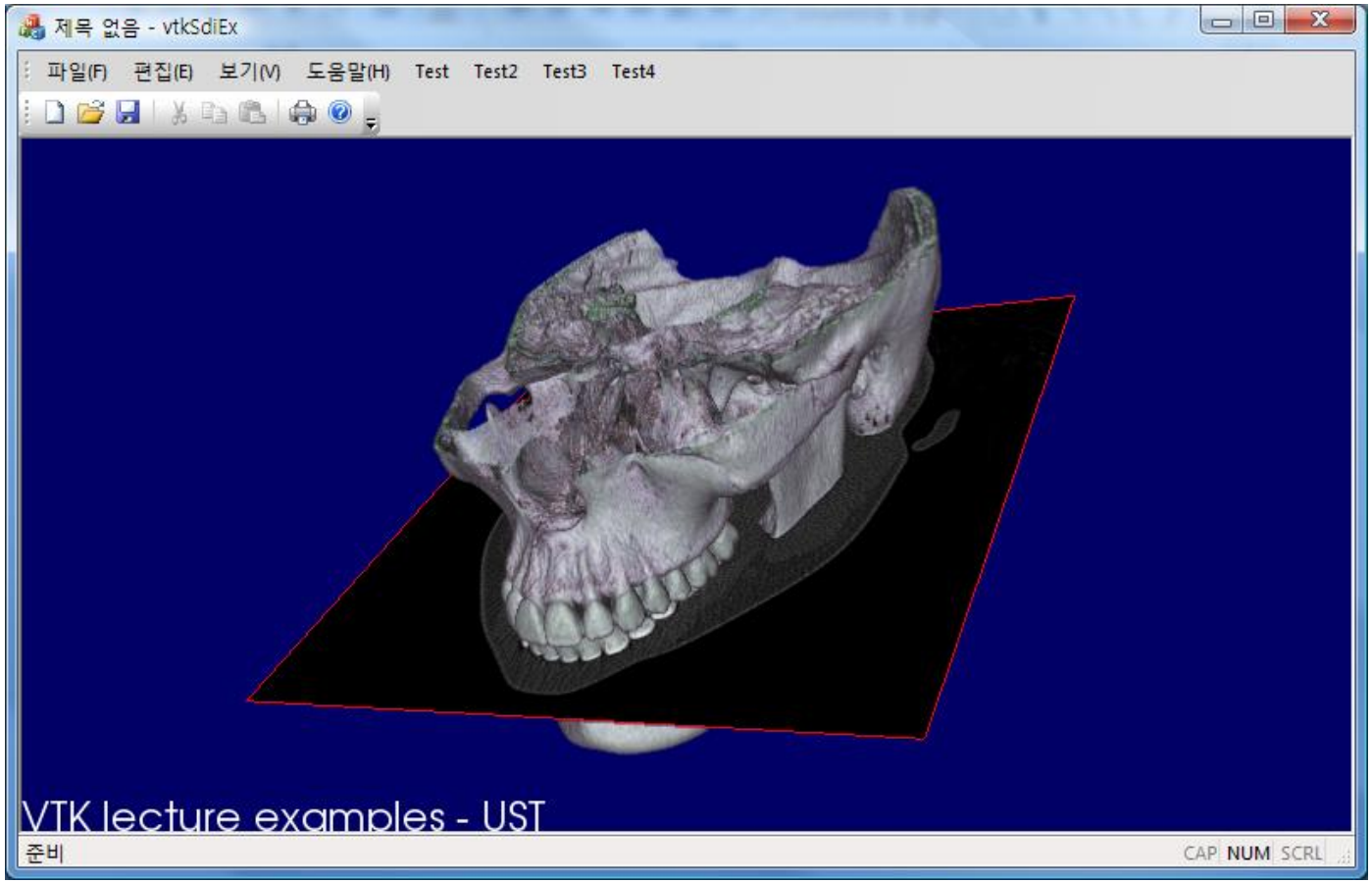
Orientation marker widget



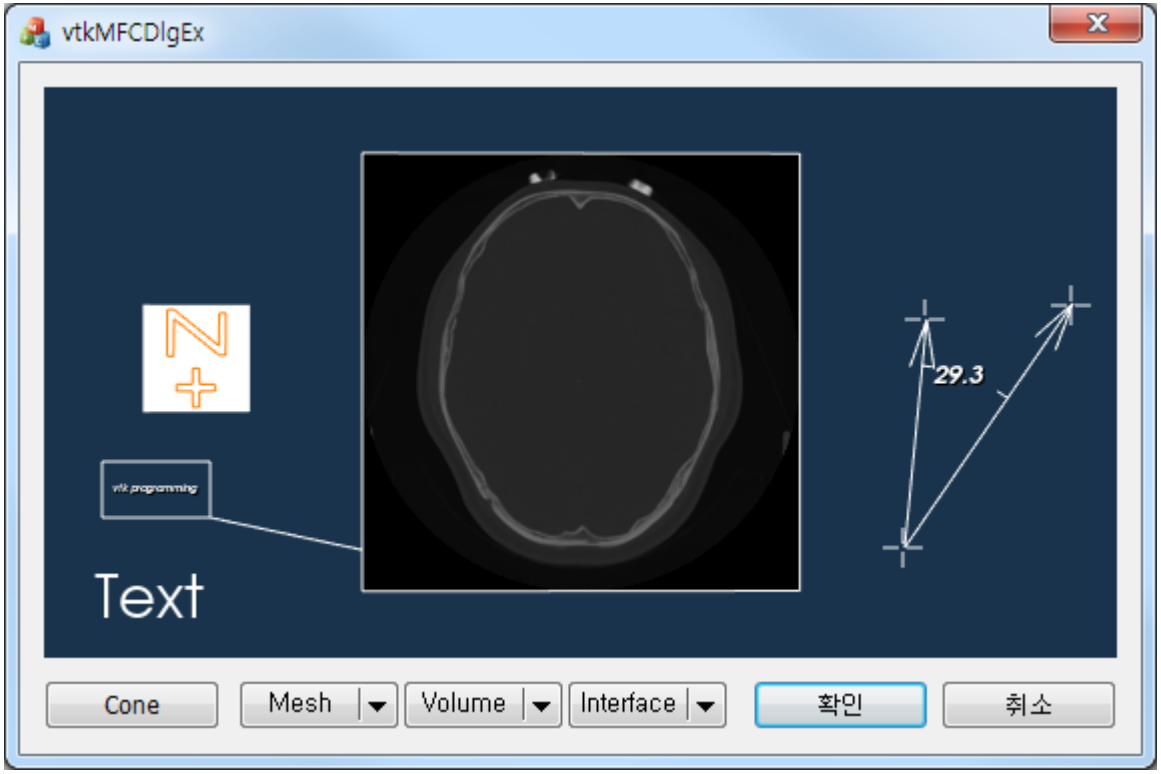
XYPlot widget

Compass widget

vtkImagePlaneWidget



Widgets


vtkMFCDlgEx
29.3
vtk programming
Text
Cone
Mesh
Volume
Interface
확인
취소

```
////////////////////////////////////////////////////////////////////////
// 1) Angle Widget 설정
m_angleWidget = vtkSmartPointer<vtkAngleWidget>::New();
m_angleWidget->SetInteractor( m_vtkWindow->GetInteractor() );
m_angleWidget->CreateDefaultRepresentation();
m_angleWidget->On();

////////////////////////////////////////////////////////////////////////
// 2) vtkImagePlaneWidget 설정 (볼륨의 단면)
vtkSmartPointer<vtkDICOMImageReader> dcmReader =
    vtkSmartPointer<vtkDICOMImageReader>::New();
dcmReader->SetDirectoryName( "../data/CT" );
dcmReader->Update();

m_imageWidget = vtkSmartPointer<vtkImagePlaneWidget>::New();
m_imageWidget->SetInteractor( m_vtkWindow->GetInteractor() );
m_imageWidget->SetInputData( dcmReader->GetOutput() );
m_imageWidget->RestrictPlaneToVolumeOn();
m_imageWidget->SetPlaneOrientationToZAxes();
m_imageWidget->SetSliceIndex( 20 );
m_imageWidget->On();

renderer->ResetCamera();
m_vtkWindow->Render();
```

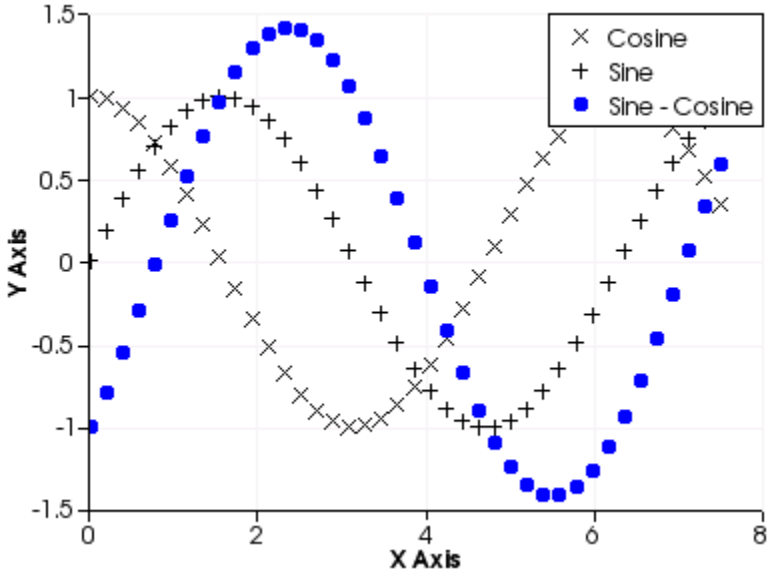
```
////////////////////////////////////////////////////////////////////////////
// 3) vtkCaptionWidget 텍스트 설정하여 생성하기
vtkSmartPointer<vtkCaptionActor2D> captionActor =
    vtkSmartPointer<vtkCaptionActor2D>::New();
captionActor->SetCaption( "vtk programming" );
captionActor->GetTextActor()->GetTextProperty()->SetJustificationToCentered();
captionActor->GetTextActor()->GetTextProperty()->SetVerticalJustificationToCentered();

m_captionWidget = vtkSmartPointer<vtkCaptionWidget>::New();
m_captionWidget->SetInteractor( m_vtkWindow->GetInteractor() );
m_captionWidget->SetCaptionActor2D( captionActor );
m_captionWidget->On();

////////////////////////////////////////////////////////////////////////////
// 4) vtkOrientationMarkerWidget 설정하여 생성하기
vtkSmartPointer<vtkAnnotatedCubeActor> cube =
    vtkSmartPointer<vtkAnnotatedCubeActor>::New();

m_orientationWidget = vtkSmartPointer<vtkOrientationMarkerWidget>::New();
m_orientationWidget->SetOrientationMarker( cube );
m_orientationWidget->SetInteractor( m_vtkWindow->GetInteractor() );
m_orientationWidget->SetViewport( 0.0, 0.0, 0.2, 0.2 );
m_orientationWidget->SetEnabled( TRUE );
m_orientationWidget->On();

////////////////////////////////////////////////////////////////////////////
// 5) vtkTextWidget 설정하여 생성하기
vtkSmartPointer<vtkTextActor> textActor =
    vtkSmartPointer<vtkTextActor>::New();
textActor->SetInput( "Text" );

m_textWidget = vtkSmartPointer<vtkTextWidget>::New();
m_textWidget->SetInteractor( m_vtkWindow->GetInteractor() );
m_textWidget->SetTextActor( textActor );
m_textWidget->On();
```

VTK for Scientific Visualization

MISC.

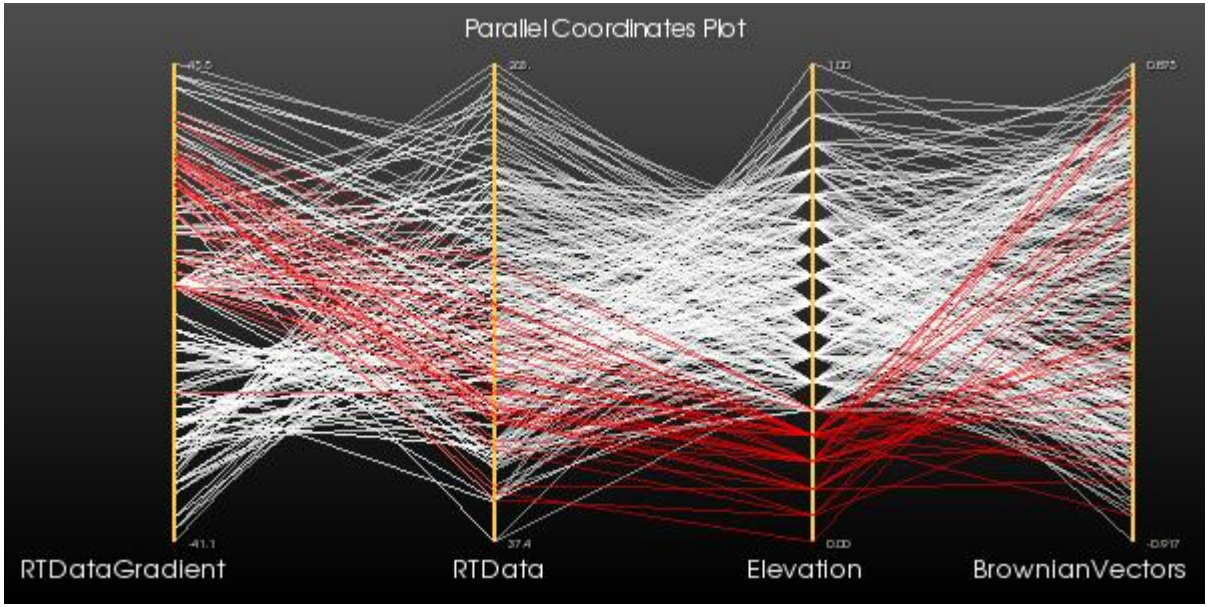
Scatter Plot

- http://www.vtk.org/Wiki/VTK/Examples/Python/Plotting/ScatterPlot



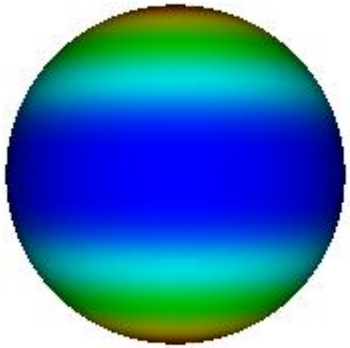
ParallelCoordinatesView

- http://www.vtk.org/Wiki/VTK/Examples/Python/Infovis/ParallelCoordinatesView



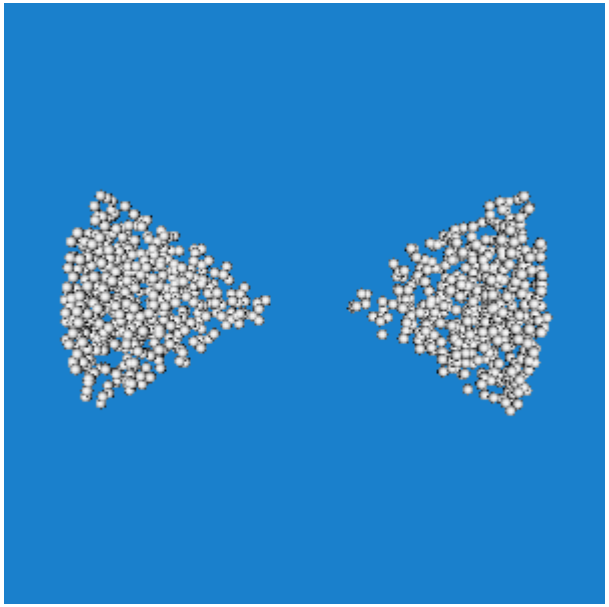
IsosurfaceSampling

- http://www.vtk.org/Wiki/VTK/Examples/Cxx/Visualization/IsosurfaceSampling



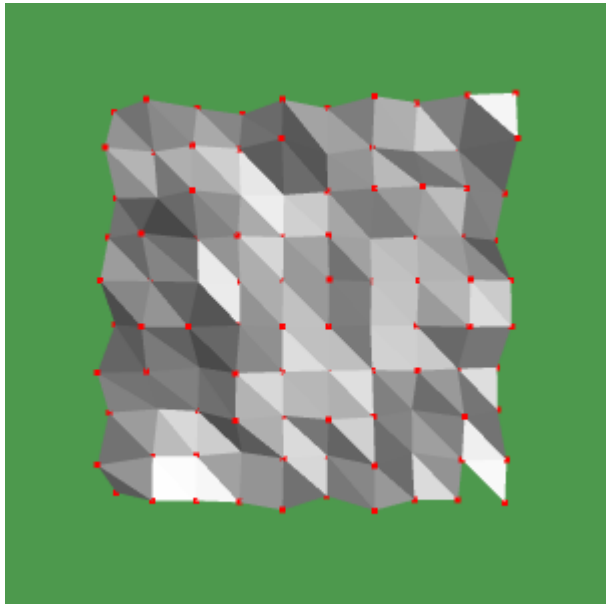
RandomProbe

- http://www.vtk.org/Wiki/VTK/Examples/Cxx/Visualization/RandomProbe



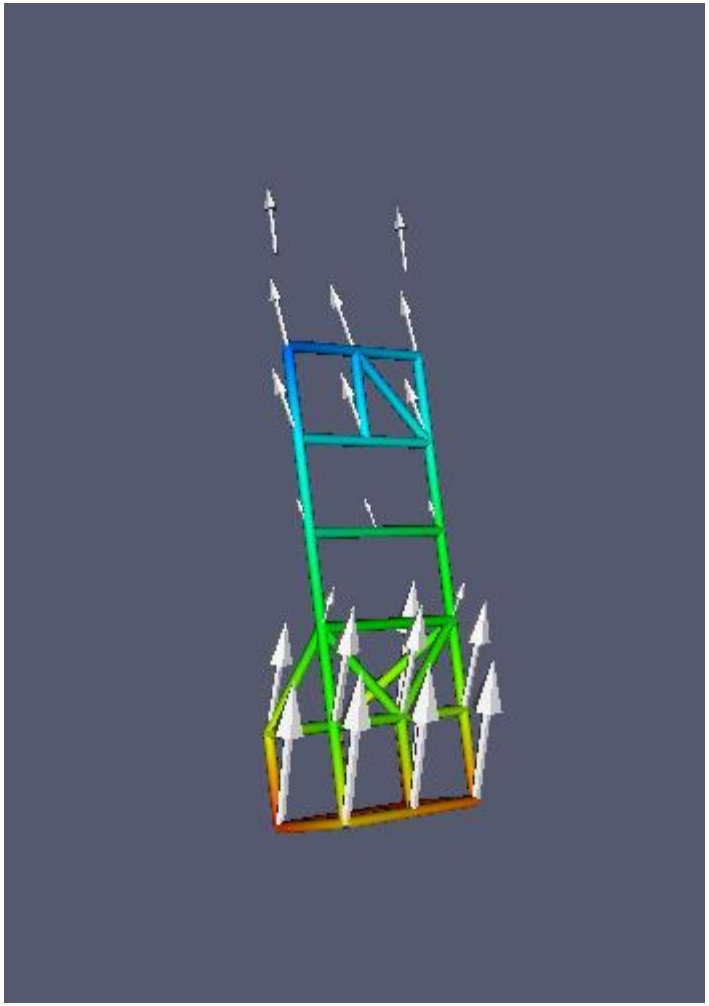
TriangulateTerrainMap

- http://www.vtk.org/Wiki/VTK/Examples/Cxx/Filters/TriangulateTerrainMap



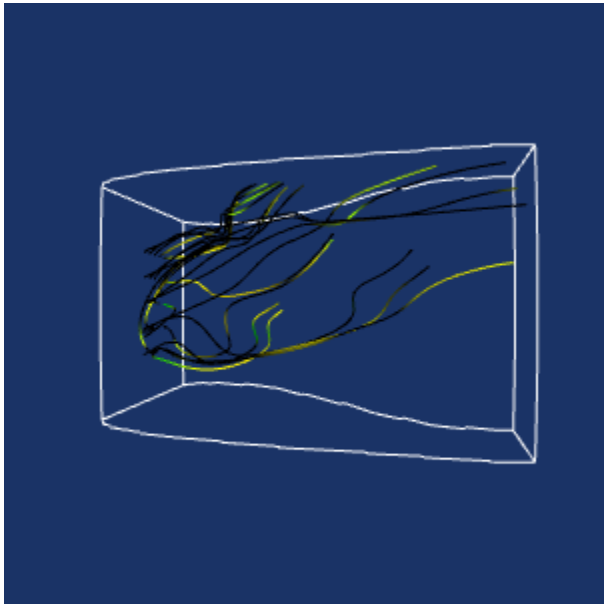
GradientFilter

- http://www.vtk.org/Wiki/VTK/Examples/Cxx/PolyData/GradientFilter



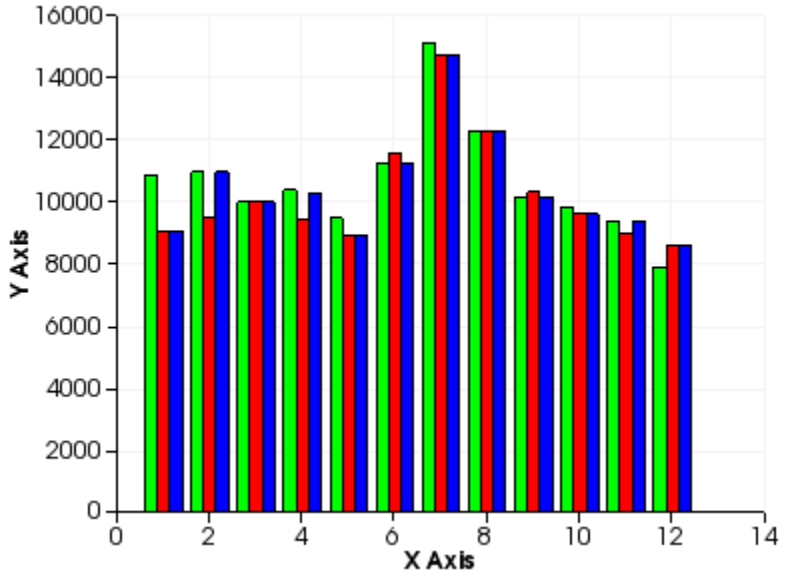
Streamlines

- http://www.vtk.org/Wiki/VTK/Examples/Python/Visualization/Streamlines



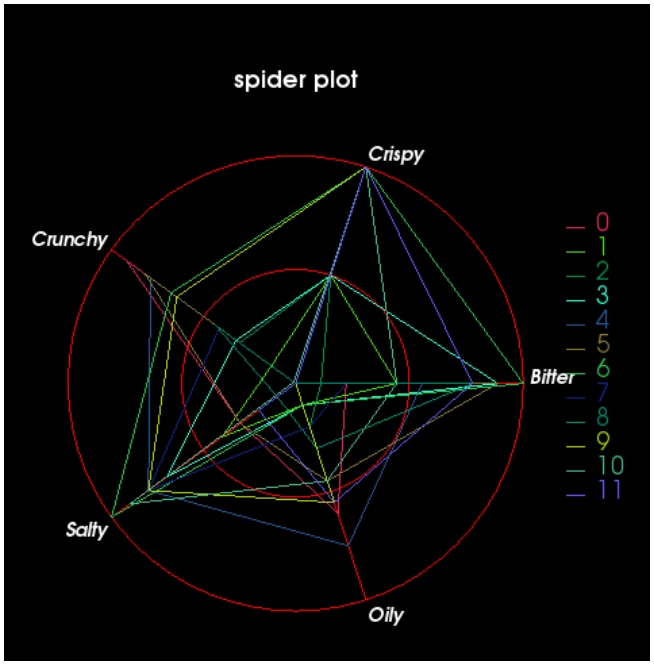
BarChart

- http://www.vtk.org/Wiki/VTK/Examples/Cxx/Plotting/BarChart



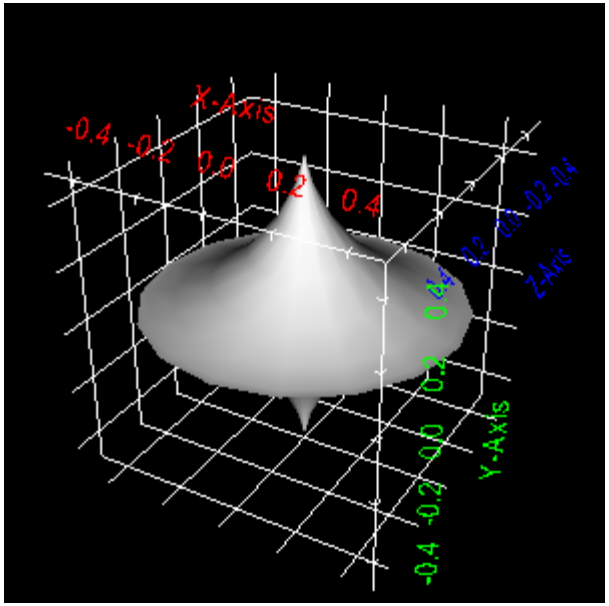
SpiderPlot

- http://www.vtk.org/Wiki/VTK/Examples/Python/Plotting/SpiderPlot



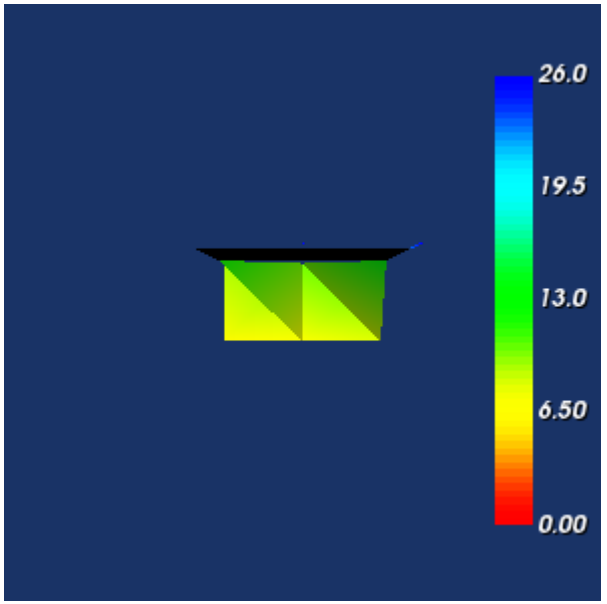
CubeAxesActor

- http://www.vtk.org/Wiki/VTK/Examples/Python/Visualization/CubeAxesActor



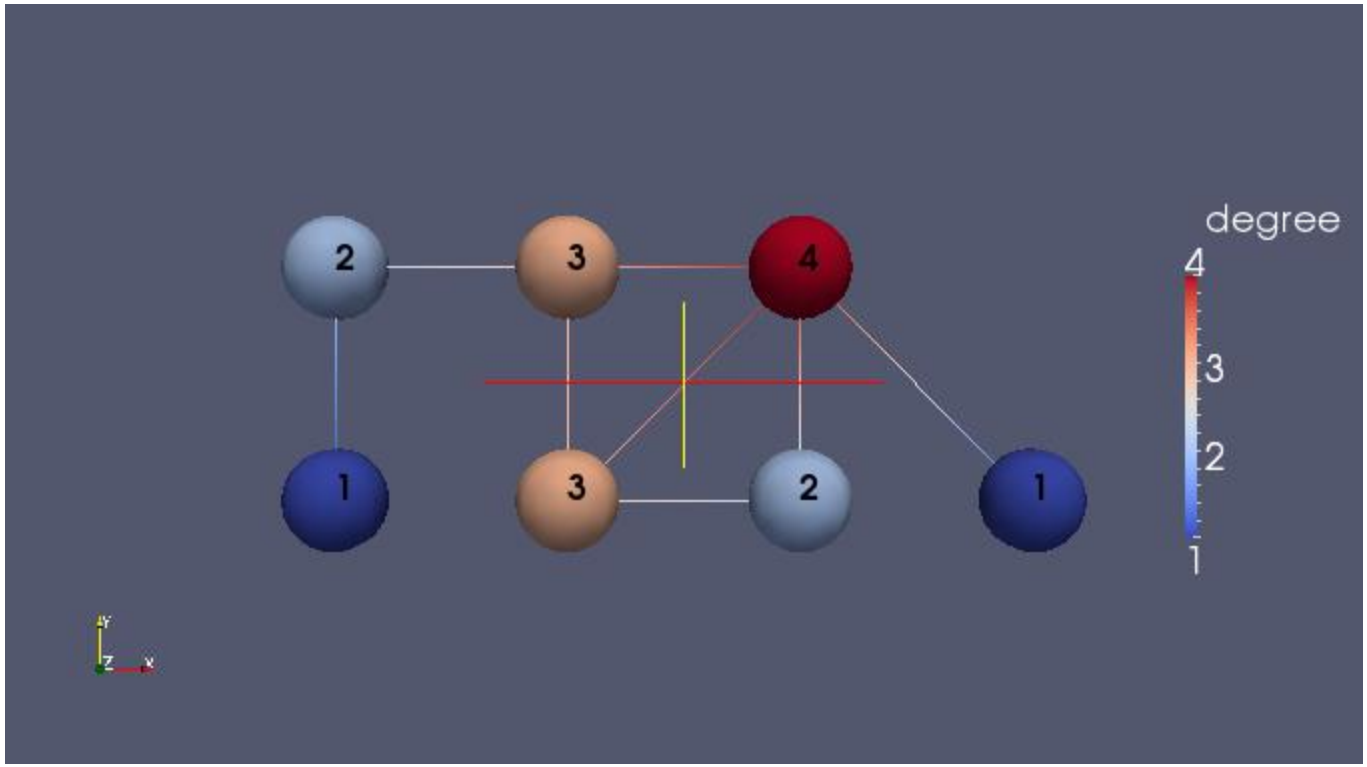
ScalarBarWidget

- http://www.vtk.org/Wiki/VTK/Examples/Python/Widgets/ScalarBarWidget



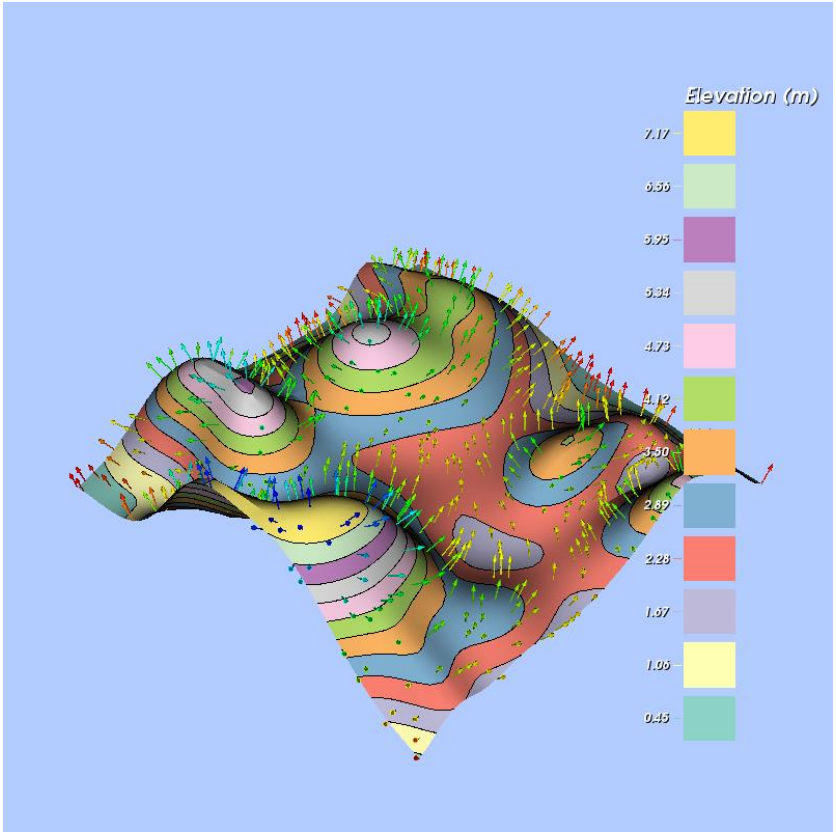
NOVCA Graph

- http://www.vtk.org/Wiki/VTK/Examples/Python/Graphs/NOVCAGraph



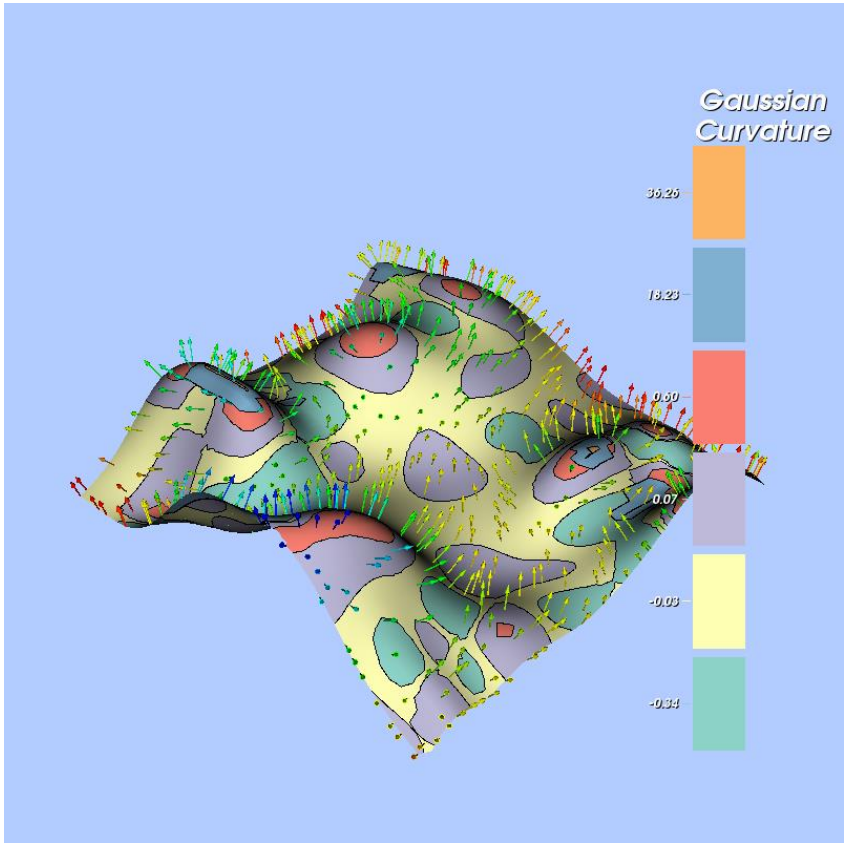
ElevationBandsWithGlyphs

- http://www.vtk.org/Wiki/VTK/Examples/Python/Visualization/ElevationBandsWithGlyphs

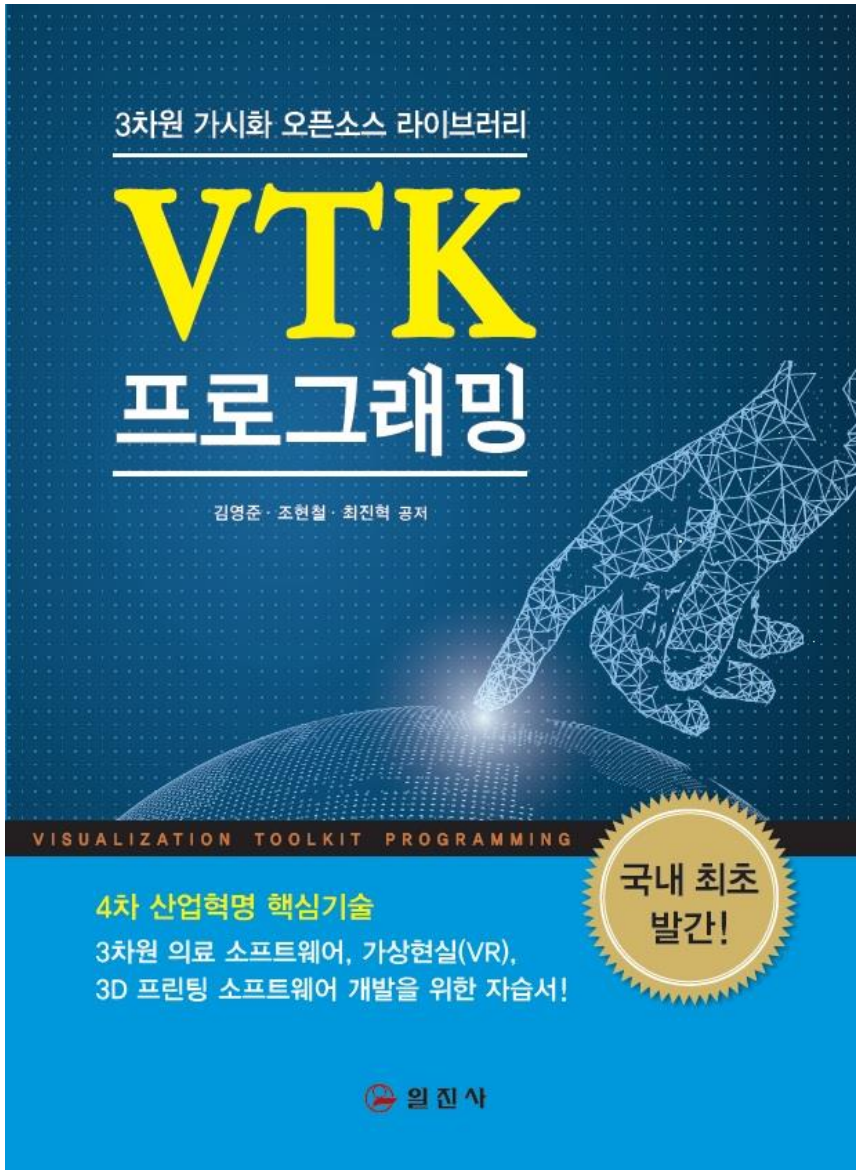


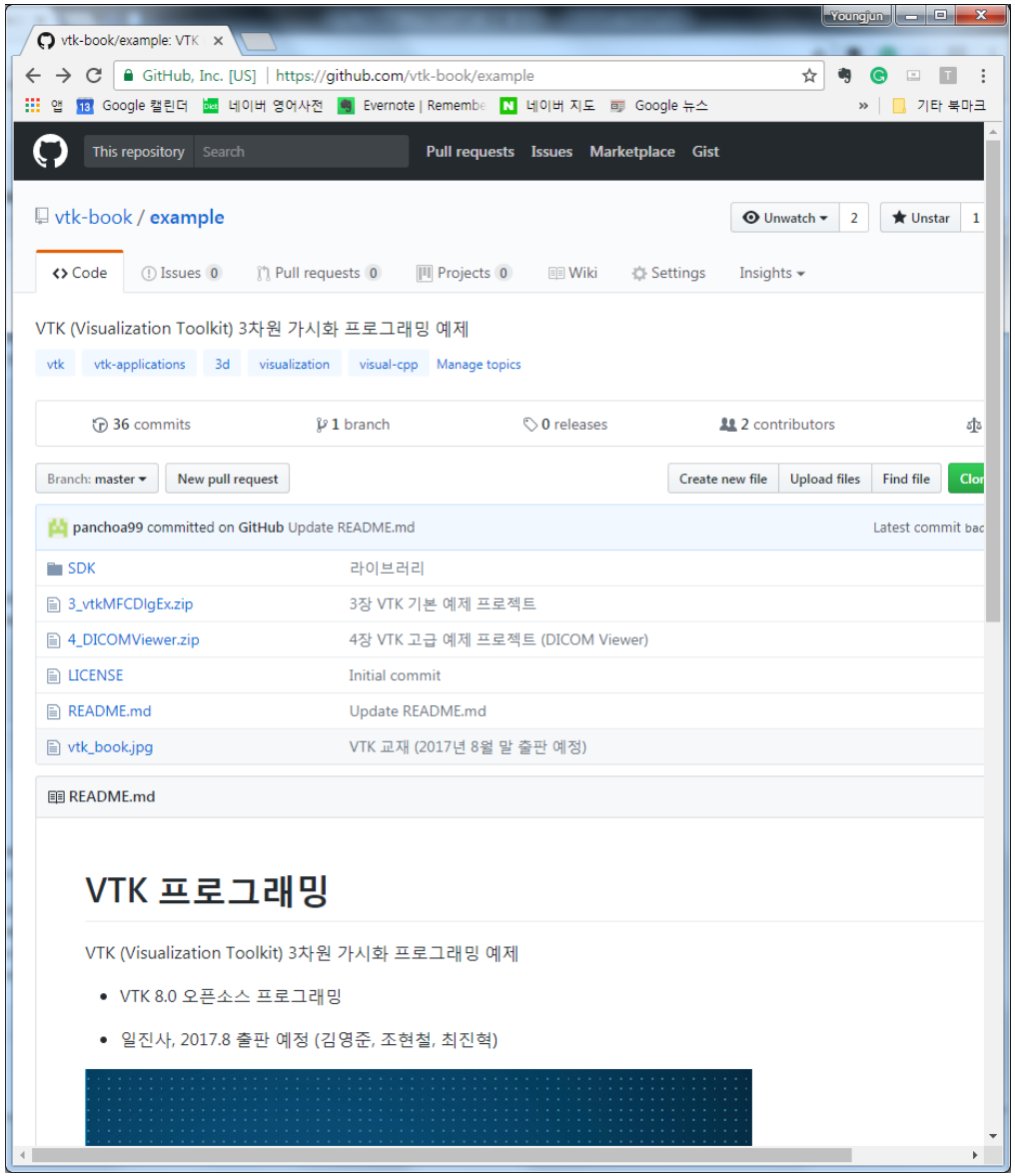
CurvatureBandsWithGlyphs

- http://www.vtk.org/Wiki/VTK/Examples/Python/Visualization/CurvatureBandsWithGlyphs



https://github.com/vtk-book/example





Chapter 04

DICOM Viewer 제작 (고급 응용 프로그램 예제)

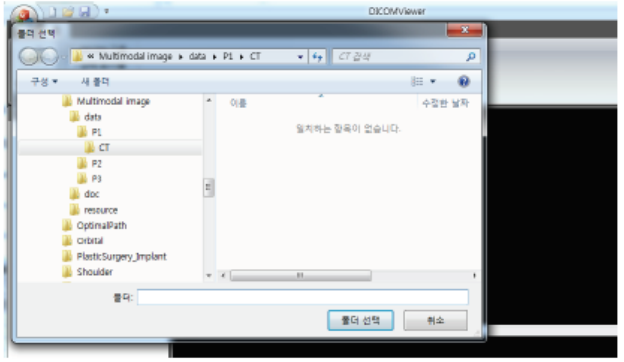
4-1 DICOM Viewer 소개

CT나 MRI 등의 영상 진단 장비에서 널리 사용되는 의료용 디지털 영상 및 통신(Digital Imaging and Communications in Medicine, DICOM) 표준 포맷으로 저장된 의료 영상 데이터를 읽고 탐색할 수 있는 프로젝트를 생성해 보자.

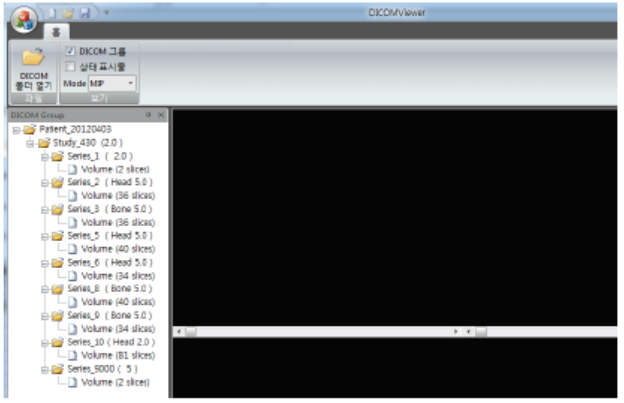
그림 4-1 간단한 DICOM Viewer 프로그램

이 프로젝트에서는 VTK와 더불어 DICOM 파일을 다룰 수 있는 GDCM 라이브러리(*https://sourceforge.net/projects/gdcm/*)를 사용한다. 이 프로젝트에서 제작할 프로그램에는 다음과 같은 기능을 포함할 것이다.

- **DICOM 폴더 읽기**
  선택한 폴더에서 DICOM 포멧 파일인 dcm 파일을 모두 로드



- **DICOM 태그 정보 읽기**
  DICOM 파일에서 필요한 태그 정보를 선택하여 읽기
- **DICOM 그룹 분류**
  DICOM 태그의 그룹 분류 정보를 읽어, 같은 그룹에 해당하는 파일을 모아 트리 구조로 표현



Thank you!

Contact Information

Youngjun Kim, Ph.D.

- Center for Bionics, Korea Institute of Science and Technology
- E-mail: junekim@kist.re.kr
- Tel.: 02-958-5606
- C.P.: 010-5234-5378