

# MITK in the context of NA-MIC

The Medical Imaging Interaction Toolkit

### Powerful toolkits for

- Visualization: VTK ([www.vtk.org](http://www.vtk.org))
- Segmentation/registration: ITK ([www.itk.org](http://www.itk.org))

### But:

insufficient support for interactive software

### MITK ...

- uses parts of NA-MIC: **ITK & VTK**
- adds features outside the scope of boths
- → is not at all a competitor to VTK or ITK


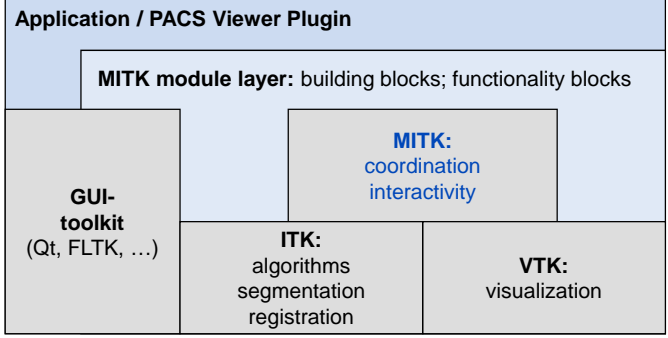
Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

**MITK**

dkfz.

### Medical Imaging Interaction Toolkit (MITK)

- open-source C++ toolkit based on ITK/VTK
- coordination of visualizations and interactions
- combine modules developed independently from each other

The diagram illustrates the MITK architecture. At the top is the 'Application / PACS Viewer Plugin' layer. Below it is the 'MITK module layer: building blocks; functionality blocks'. This layer is supported by three main components: 'GUI-toolkit (Qt, FLTK, ...)', 'MITK: coordination interactivity', and a base layer consisting of 'ITK: algorithms segmentation registration' and 'VTK: visualization'.

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

**MITK**

dkfz.

- Object oriented C++ Framework/Toolkit
- Supports
  - gcc 3.3, 4.2, VC7.1, VC8, VC9
  - Latest VTK release
  - Latest two ITK releases
- MITK-Core does not depend on a GUI toolkit
- MITK-Application-Level provides
  - Qt3 base application
  - Many Qt3 widgets
  - FLTK example
  - Qt4 is work in progress

Ivo Wolf  
Medizinische und Biologische Informatik  
Heidelberg

**dkfz.**

## Re-use of design and concepts

```

    graph TD
      FLTK --> FLmitk
      ITK --> MITK
      FLmitk --> MITK
      MITK --> VTK
      style MITK fill:#add8e6
  
```

- MITK's core is GUI independent

Ivo Wolf  
Medizinische und Biologische Informatik  
Heidelberg

**dkfz.**

## Tools and software process

**CMake:**  
config and build system

**ITK Modules**

- Data Representation Objects
- Image Representation Objects
- Mesh Representation Objects
- Path Representation Objects
- Geometry Representation Objects
- Data Access Objects

**Doxygen:**  
documentation

**Subversion:**  
version management

**SourceForge:**  
mailing list

**Bugzilla:**  
bug tracking

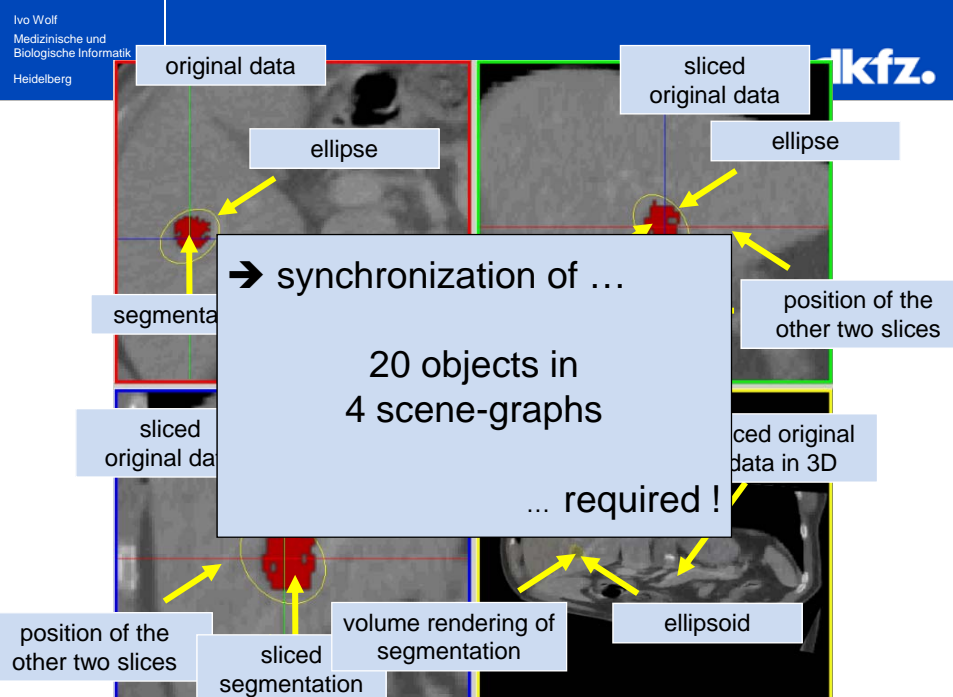
**DART:**  
automatic builds and test runs

# What MITK does – a quick overview



DEUTSCHES  
KREBSFORSCHUNGSZENTRUM  
IN DER HELMHOLTZ-GEMEINSCHAFT

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg



original data

sliced original data

segmentation

sliced segmentation

volume rendering of segmentation

ellipse

ellipsoid

position of the other two slices

position of the other two slices

position of the other two slices


position of the other two slices

→ synchronization of ...

20 objects in 4 scene-graphs

... required !

sliced original data in 3D



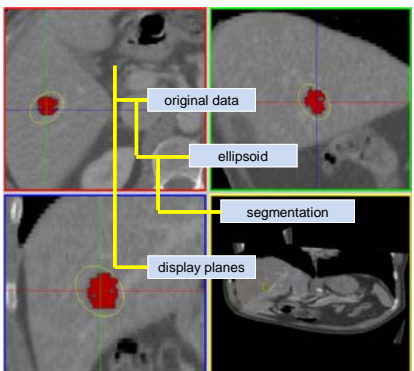
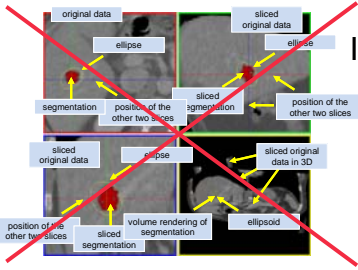
Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

Getting out of the maze ...

dkfz.

Instead of creating **many** scene-graphs  
with **even more** elements ...

... create a **single data-tree**  
with a **few data-objects!**



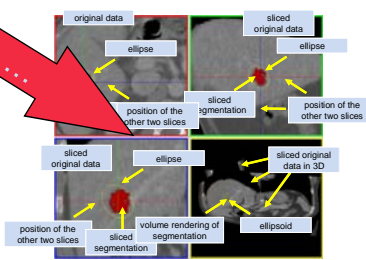
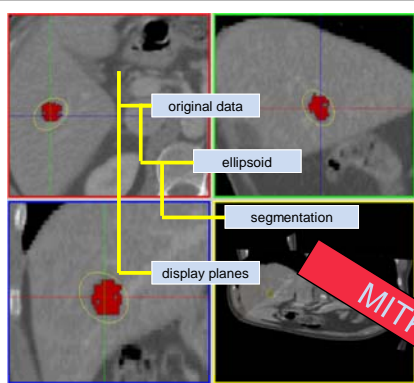
Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

MITK:  
Data-tree instead of scene-graphs

dkfz.

MITK takes the data-tree ...  
and builds ...  
→ VTK scene graphs

MITK creates ...



Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## Data repository

dkfz.

- Repositories for sharing data objects between modules
- Any number of data objects
- Any kind of data objects
- Data objects with geometry frame (bounding-box, transform, etc.)

```

graph LR
    Root[ ] --- A[Abdominal CT (Image)]
    Root --- B[Liver (Surface)]
    Root --- C[Tumor (Surface)]
    Root --- D[Vessels (Graph)]
    Root --- E[MRI (Image)]
    Root --- F[Helper Objects]
    Root --- G[Landmarks (Points)]
  
```

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## Rendering the data-tree

dkfz.

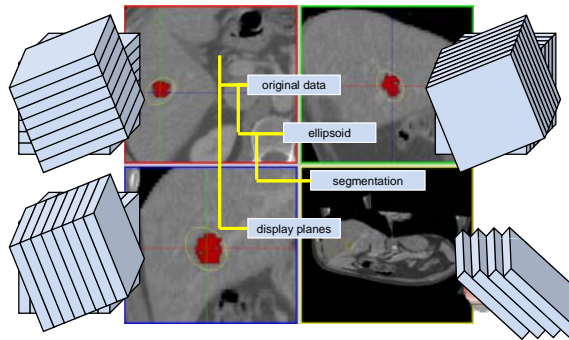
**RenderWindows:**

- single RenderWindow class
- different types of views
  - 2D/3D
  - special views definable (e.g., for AR)
- point to the data repository
  - any number of views on the data:
 

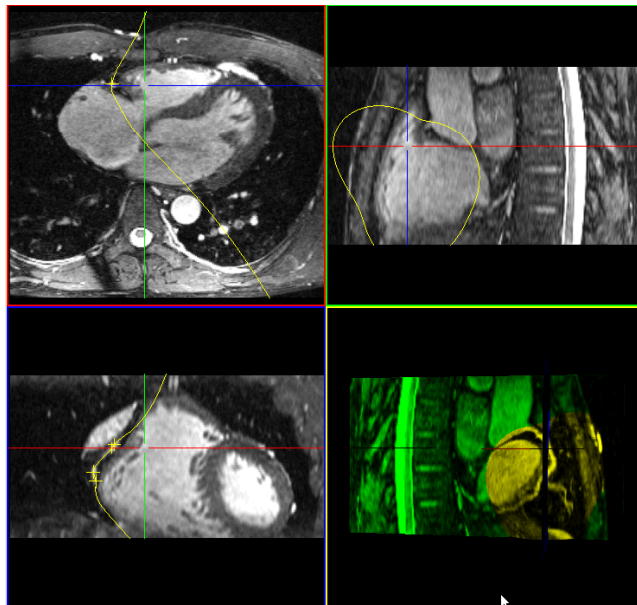
```

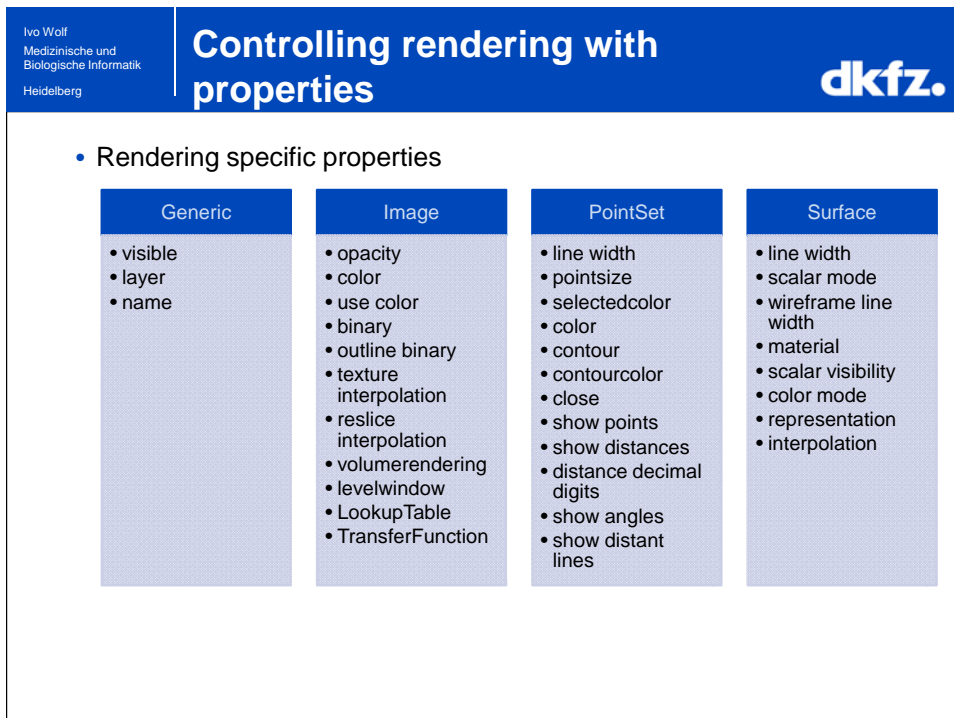
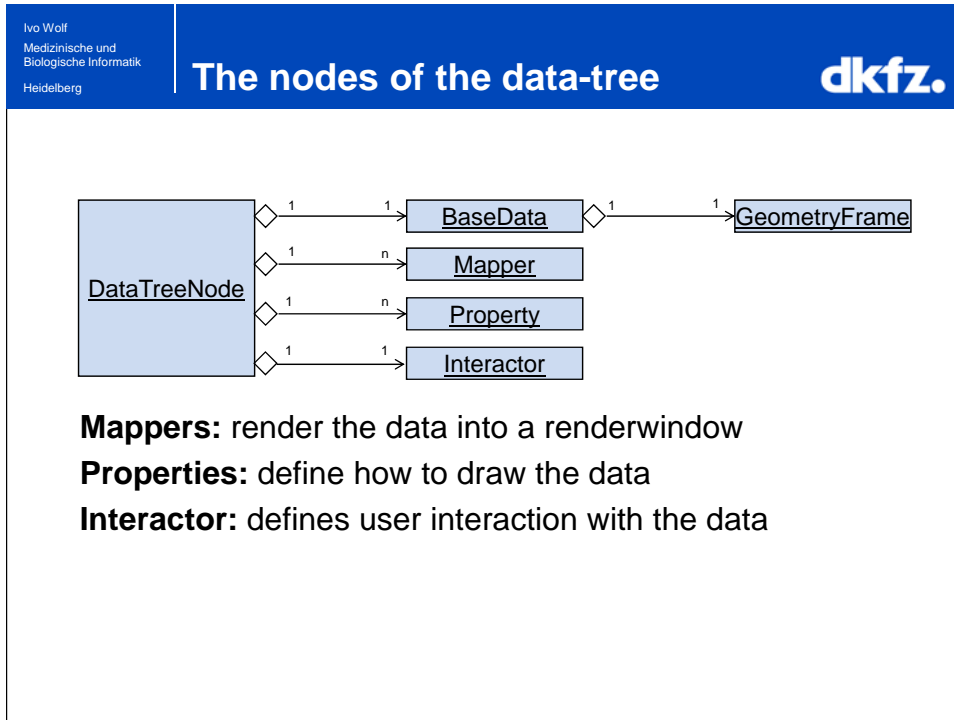
renderer1->SetData(repository);
renderer2->SetData(repository);
...
          
```

## Defining how we want to see the data ...



## Render and interact on curved planes

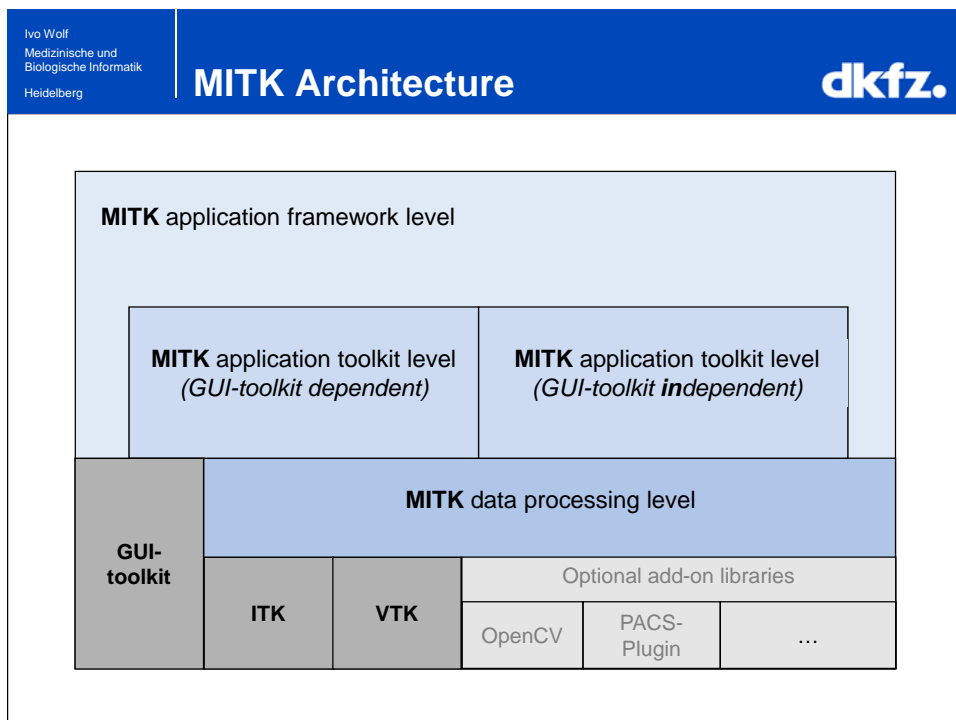






# MITK Architecture


**DEUTSCHES  
KREBSFORSCHUNGSZENTRUM  
IN DER HELMHOLTZ-GEMEINSCHAFT**



Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## Data Processing level

dkfz.

- access to ITK and VTK data structures and algorithms
- Access to other libraries (OpenCV, ANN, TinyXML,...)
- Tree / Graph data structures and algorithms
- Spatial object location (Geometries)
- Time steps for data objects
- Loading / saving of different file formats
- Interface to tracking systems

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## Application toolkit level

dkfz.

- Rendering
  - Mappers, Update Management, Render Properties
- Data Management
  - Object Container, Object Properties, Scene Management
- Interaction
  - Statemachine based
- Undo/Redo
- Processing of tracking data
- Qt Widgets
  - TreeNodeSelector, StandardViews, PropertyEditor, LevelWindow, Renderwindow, SlicerControls, Navigationviews,...

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## Application framework

dkfz.

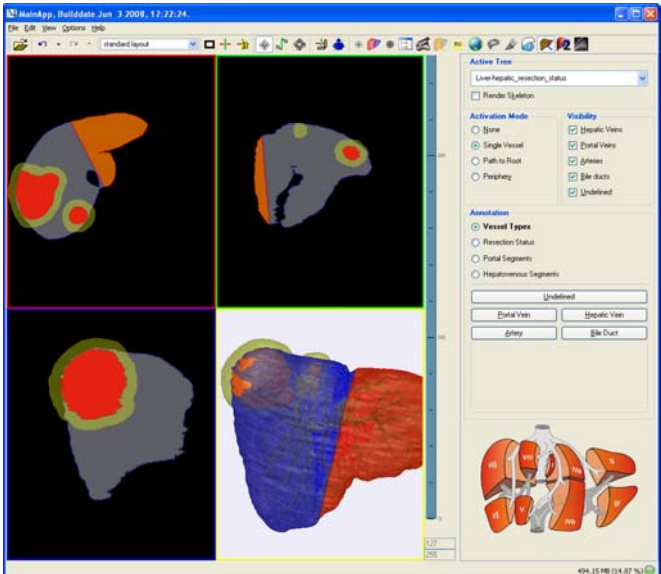
Base application (*MITK-MainApp*):

- Container for functionalities
  - independent „Plug-Ins“ for specific problems
- Shared repository for data objects
- Persistence:
  - Application state can be saved and restored on next startup
- Interface to CHILI-PACS Workstation

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## MITK - MainApp

dkfz.



The screenshot displays the MITK - MainApp interface. The main window is titled "MainApp - Builddate: Jun 3 2009, 17:22:24". It features a 2x2 grid of medical images: top-left shows a cross-section of a liver with red and yellow regions; top-right shows a similar cross-section with a different segmentation; bottom-left shows a cross-section with a large red region; bottom-right shows a 3D reconstruction of a liver with blue and red surfaces. To the right of the grid is a control panel with the following sections:

- Active Tools:** Liver-Resection, resection\_status, Flesher Selection.
- Activation Mode:**
  - Slice
  - Single Vessel
  - Path to Root
  - Postprocessing
- Visibility:**
  - Dorsal Veins
  - Dorsal Veins
  - Splenic
  - Bile ducts
  - Undefined
- Annotations:**
  - Vessel Types
  - Resection Status
  - Portal Segments
  - Hepaticovenous Segments
- Buttons:**
  - Undefined
  - Dorsal Vein
  - Splenic Vein
  - Spleen
  - Bile Duct

At the bottom right, there is a small 3D anatomical diagram of the liver and its associated vessels. The status bar at the bottom indicates "494.15 MB (14.87 %)".

Ivo Wolf  
Medizinische und Biologische Informatik  
Heidelberg

## MITK functionality modules

**dkfz.**

**Functionality** = a module with ...

- an identification (icon/tooltip/...)
- a workspace area
- a control area
- a option dialog
- a help page (manual)
- the algorithmic implementation

workspace area      control area

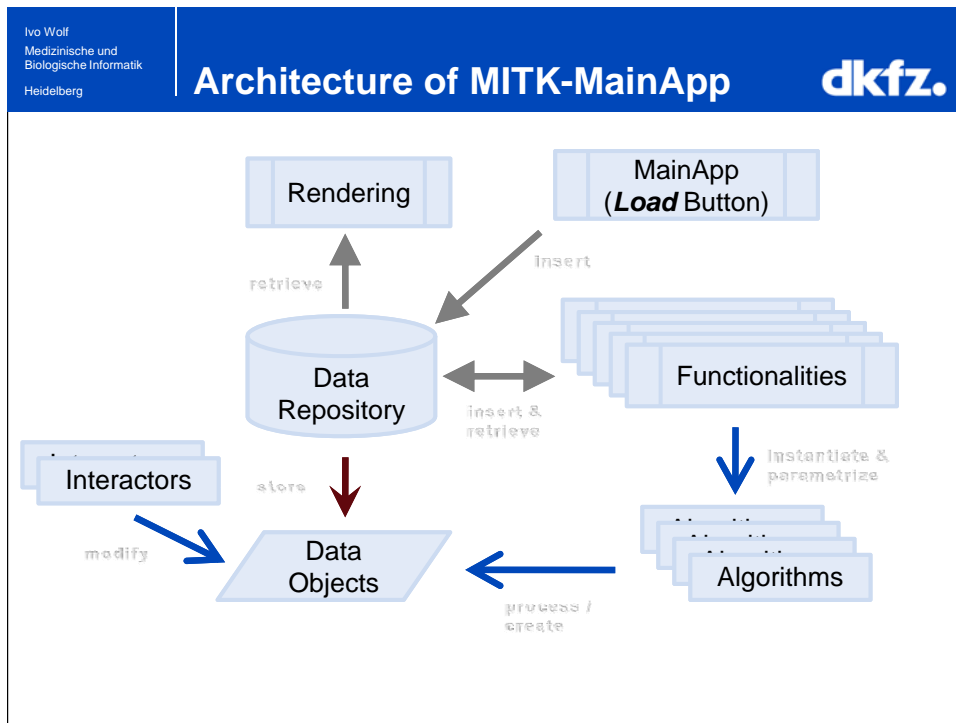
Ivo Wolf  
Medizinische und Biologische Informatik  
Heidelberg

## Combining functionality blocks

**dkfz.**

- Functionalities are independent from each other
- They communicate via the data repository

Heart-ID694119  
 attractors  
 epicardium  
 masked epicardium  
 local threshold region  
 masked local threshold region  
 simple-mesh model



## How to get started

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

www.mitk.org

dkfz.

Download options:

- anonymous svn:  
svn co <http://svn.mitk.org/trunk/mitk/>
- zipped archive (v 0.8)  
<https://sourceforge.net/projects/mitk/>

Tutorial:  
<http://mitk.org/documentation/>

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

9-step tutorial

dkfz.

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## A small functionality

dkfz.

We'll have a look at a very simple functionality for region growing:

0. (create a functionality)
1. select an image
2. set some seed points
3. react, when a GUI button is pressed
4. run a region grower from ITK
5. display the result in MITK

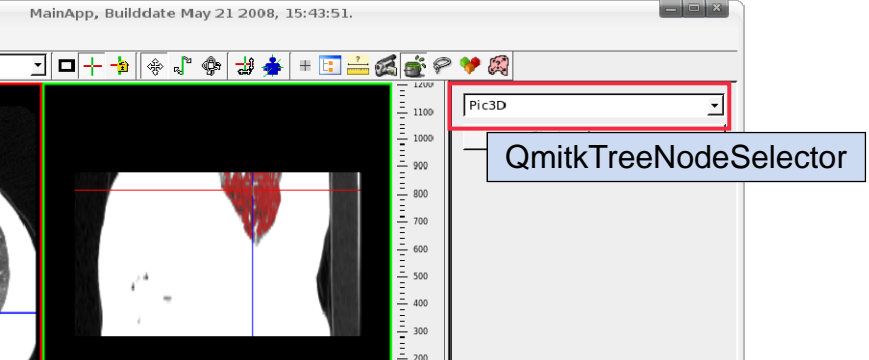
(can be downloaded at [mitk.org](http://mitk.org))

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## A small functionality

dkfz.

1. selection of an image
2. set some seed points
3. react, when a GUI button is pressed
4. run a region grower from ITK
5. display the result in MITK



The screenshot shows the MITK application window titled "MainApp, Builddate May 21 2008, 15:43:51.". The main view displays a 3D visualization of a heart. A red box highlights a dropdown menu in the top right corner of the application window, which is labeled "Pic3D". A callout box points to this dropdown menu with the text "QmitkTreeNodeSelector".

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

**A small functionality**

dkfz.

1. selection of an image
2. set some seed points
3. react, when a GUI button is pressed
4. run a region grower from ITK

**PointSetInteractor**

```

QmitkRegionGrowing.cpp (~:/mitk/extern/src/QFunctiona
File Edit View Terminal Tabs Help
92
93 void QmitkRegionGrowing::Activated()
94 {
95     QmitkFunctionality::Activated();
96
97     if ( m_PointSetNode.IsNull() )
98         // only once create a new DataTreeNode containing a PointSet with some interaction
99         {
100             // new node and data item
101             m_PointSetNode = mitk::DataTreeNode::New();
102             m_PointSetNode->GetPropertyList()->SetProperty("name", mitk::StringProperty::New("Seedpoints for region growing"));
103             m_PointSet = mitk::PointSet::New();
104             m_PointSetNode->SetData( m_PointSet );
105
106             // new behaviour/interaction for the pointset node
107             m_Interaction = mitk::PointSetInteractor::New("pointsetinteractor", m_PointSetNode);
108             mitk::GlobalInteraction::GetInstance()->AddInteractor( m_Interaction );
109
110             // add the pointset to the data tree (for rendering)
111             GetDataTreeIterator()->Add( m_PointSetNode );
112         }
113     }
114
"--/mitk/extern/src/QFunctionalities/QmitkRegionGrowing/QmitkRegionGrowing.cpp" 238 lines --45%-- 109,0-1 42%

```

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

**A small functionality**

dkfz.

1. selection of an image
2. set some seed points
3. react, when a GUI button is pressed
4. run a region grower from ITK
5. display the result in MITK

**Qt "connections"**

```

QmitkRegionGrowing.cpp (~:/mitk/ek...onanties/qmitkregiongrowing) + view
File Edit View Terminal Tabs Help
71
72 void QmitkRegionGrowing::CreateConnections()
73 {
74     if ( m_Controls )
75     {
76         connect( (QObject*)(m_Controls->btnRegionGrow), SIGNAL(clicked()),
77                 this, SLOT(DoRegionGrowing()) );
78     }
79 }
80
80,0-1 30%

```



Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

**A small functionality**

dkfz.

1. selection of an image
2. set some seed points
3. react, when a GUI button is pressed
4. **run a region grower from ITK**
5. display the result in MITK

```

graph TD
    A[mitk::Image] --> B[AccessByItk macro]
    B --> C["templated method for ITK code  
itk::Image<TPixel, VImageDimension>"]
    C --> D[mitk::ImportItkImage()]
    D --> E[mitk::Image]
  
```

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

**A small functionality**

dkfz.

3. react, when a GUI button is pressed
4. run a region grower from ITK
5. **display the result in MITK**

```

QmitkRegionGrowing.cpp (~/mitk/extern/...Functionalities/QmitkRegionGrowing) - VIM
File Edit View Terminal Tabs Help
217 regionGrower->Update();
218
219
220 mitk::Image::Pointer resultImage = mitk::ImportItkImage( regionGrower->GetOutput() );
221 mitk::DataTreeNode::Pointer newNode = mitk::DataTreeNode::New();
222 newNode->SetData( resultImage );
223
224 // set some properties
225 mitk::DataTreeNodeFactory::SetDefaultImageProperties( newNode );
226 newNode->SetProperty("binary", mitk::BoolProperty::New(true));
227 newNode->SetProperty("name", mitk::StringProperty::New("dumb segmentation"));
228 newNode->SetProperty("color", mitk::ColorProperty::New(1.0,0.0,0.0));
229 //newNode->SetProperty("volumerendering", mitk::BoolProperty::New(true));
230 newNode->SetProperty("layer", mitk::IntProperty::New(1));
231 newNode->SetProperty("opacity", mitk::FloatProperty::New(0.5));
232
233 // add result to data tree
234 mitk::DataStorage::GetInstance()->Add( newNode );
235
236 mitk::RenderingManager::GetInstance()->RequestUpdateAll();
237 }
238
238,0-1 Bot
  
```

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

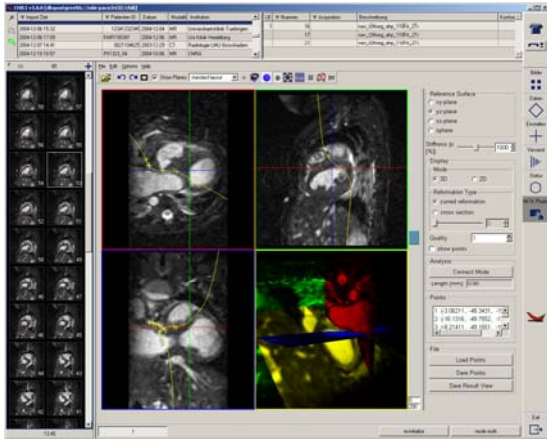
## Embedding in PACS/tele-conferencing system

dkfz.

Integration in PACS/telemedicine system CHILI® as a PlugIn:

- PACS
- Connection to modalities
- DICOM import/export
- DICOM "unification"
- Data transfer
- Tele-radiology
- Management of results from image processing

→ facilitates clinical integration



The screenshot shows a software interface for medical image processing. It features a central display area with four quadrants showing different views of a medical scan. To the left is a vertical strip of smaller image thumbnails. On the right, there is a control panel with various settings, including 'Distance Surface', 'Display Mode', 'Information Type', and 'Quality'. The interface is designed for clinical use, likely for tele-radiology or image analysis.

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## Navigation / IGT with MITK

dkfz.

- Tracking component allows access to different tracking systems:
  - NDI Polaris/Aurora
  - Microntracker
  - Our own video based Inside-Out-Tracking algorithm
- Filter pipelines for tracking coordinates (Kalmanfilter,...)
- Logging & replay of tracking data
- Geometry classes to manage different coordinate systems
- (not yet open source)

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

# Examples of IGT applications with MITK

dkfz.

The collage consists of several panels: top-left shows a 3D brain model with fiber-like structures; top-middle is a circular control panel with a yellow arrow and a green circle; top-right is a CT scan of a chest with a green tube and red vessels; middle-left is a surgical view of a brain with a red laser line; middle-right is a 3D tree-like structure; bottom-left is a circular view of a tube with a red line; bottom-right is another circular view of a tube with a red line.

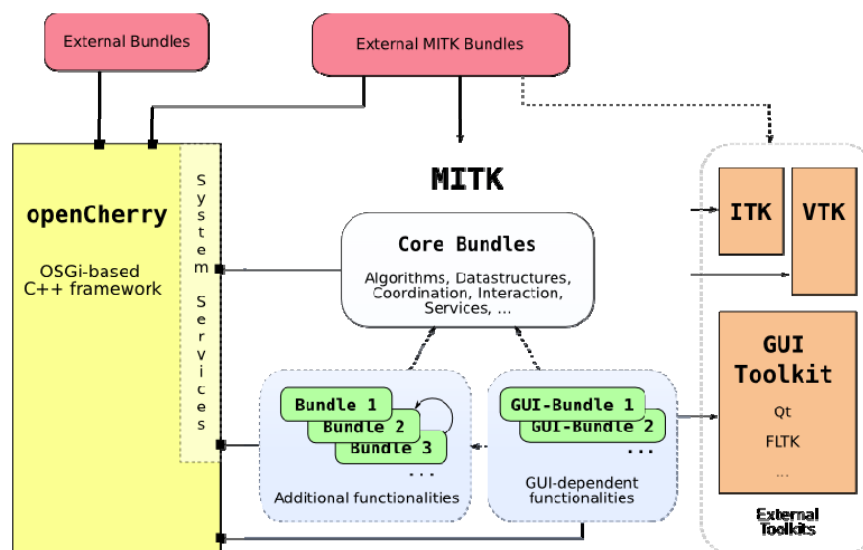
# The future

dkfz. DEUTSCHES  
KREBSFORSCHUNGSZENTRUM  
IN DER HELMHOLTZ-GEMEINSCHAFT

## OSGi-based Extensibility

- OSGi is a component model originally designed for Java
- Many ideas can be borrowed from OSGi for a C++ component model
- Basic building blocks are *bundles* (aka plugins) and *services*

## OSGi-based Extensibility



Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

## Other enhancements

dkfz.

**Hot topics:**

- Transition of the Qt3 MITK code to Qt4
- Application platform providing views/editors, perspectives and GUI-services (openCherry plugins)
- Python scripting

Ivo Wolf  
Medizinische und  
Biologische Informatik  
Heidelberg

dkfz.

# Thank you !

  
www.mitk.org