17. Oktober 2012

Image Accessors

Einarbeitungsprojekt

Joseph Görres

Abteilung für medizinische und biologische Informatik
Deutsches Krebsforschungszentrum (DKFZ), Heidelberg

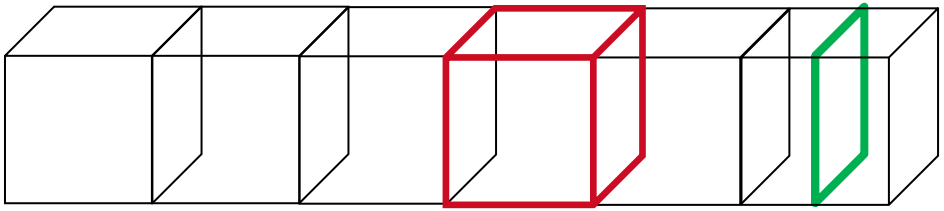
dkfz.
DEUTSCHES
KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

Task / Motivation

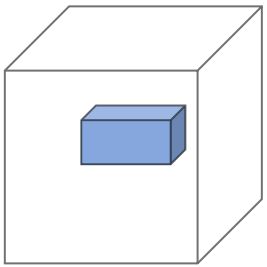
- Develop an interface for controlled and well-defined image access
  - Management and surveillance of image access
  - Thread-safe image access
  - Restrict image access to specific image areas
  - Simple and comfortable access with get- and set-methods

Access to a specific image part

- 4D, volume or slice
  - (called: ImageDataItems)



- Index gets dimensionality of the selected image part
  - (volume → 3D, slice → 2D)



- Extendable to arbitrary itk::ImageRegions
  - Connect with ITK streaming-pipeline
  - On-demand loading of particular image parts

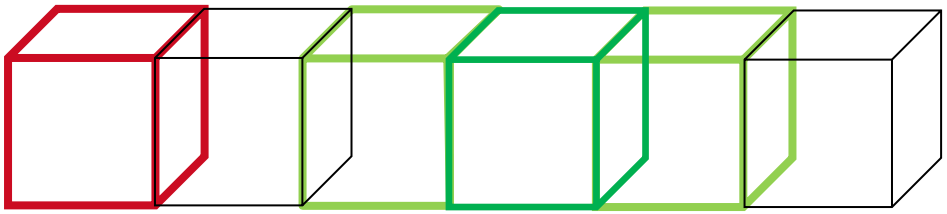
Comfortable pixel access

- **GetPixelByIndexSafe**(itk::Index<VDimension>)
  - Slow, but safe → Exception in case of overflow
- **GetPixelByIndex**(itk::Index<VDimension>)
  - Fast, only an offset is computed and the result is dereferenced
  - A consistent memory is not guaranteed for addresses apart your requested image area
- **SetPixelByIndexSafe**(itk::Index<VDimension>, TPixel)
- **SetPixelByIndex** (itk::Index<Vdimension>, Tpixel)

Thread-safe image access

- The image accessor class provides a lock-mechanism:
  - Arbitrary image areas can be locked
  - Decide between exclusive write-access and concurrent read-access

Write: No Overlaps allowed



Read: Overlaps among each other allowed:

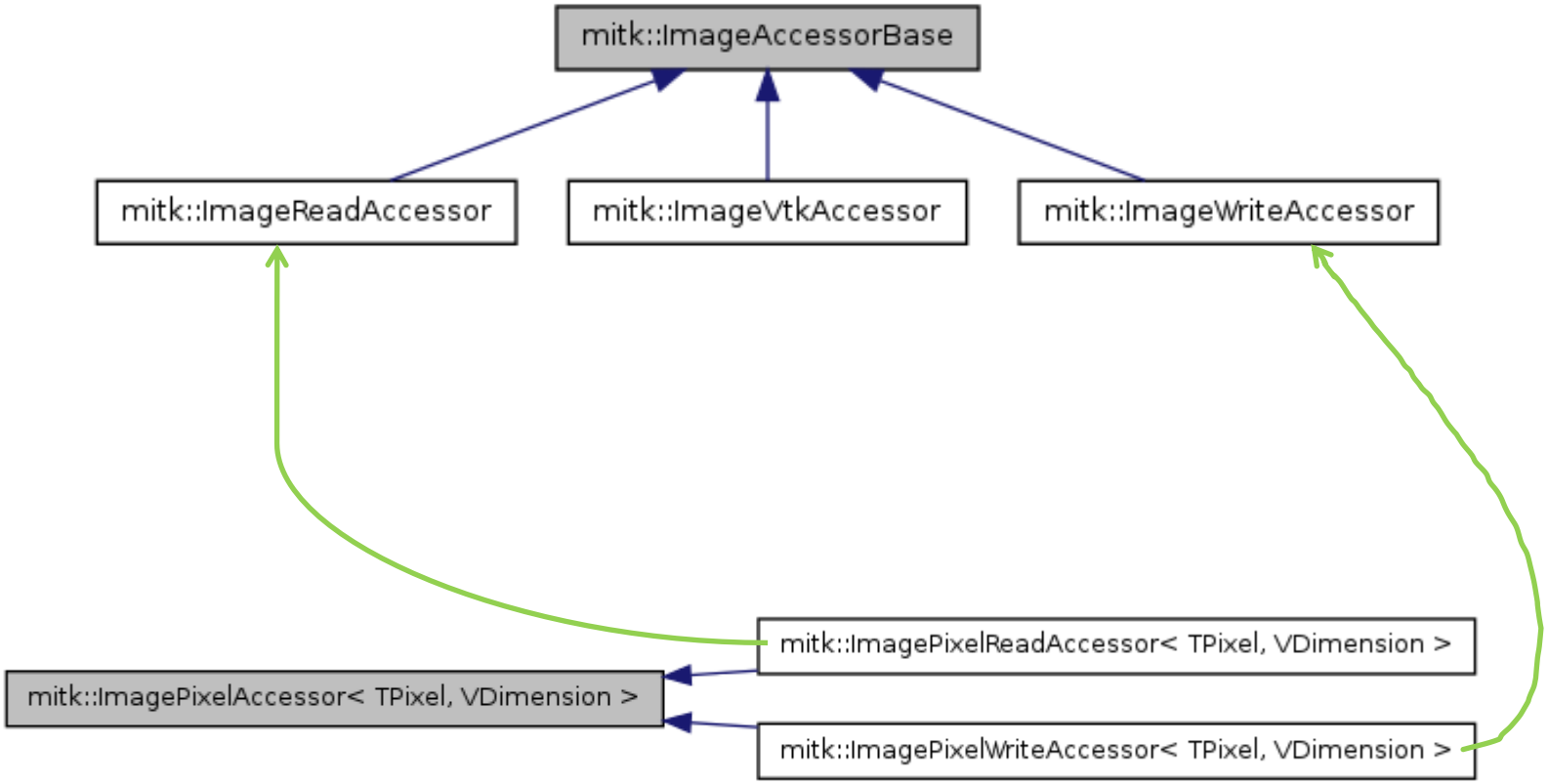
1 Volume overlaps
3 Volumes

  - In case of image area overlap
    - Wait automatically until image area is free
    - Or catch an exception

Management and surveillance

- When applying the interface, it is always known:
  - how many image accessors are present
  - which access rights were granted
  - which image areas are affected

Class hierarchy



Example: Access an image slice

```
#include <mitkImagePixelWriteAccessor.h>
#include <itkIndex.h>

// pre-condition: an mitk::Image::Pointer im exists
try {
  // request an ImageWriteAccessor for slice 3 of the image im
  mitk::ImagePixelWriteAccessor<int,2> iA(im,im->GetSliceData(2));
  // define pixel position
  itk::Index<2> idx = {{ 123, 92 }};
  // pixel write access
  iA.SetPixelByIndexSafe(idx,42);
}
catch(mitk::Exception e)
{
  // catch exception, e.g. invalid ImageAccessor
  // deal with the situation, not to have access
}
```

Integration concept

- The image accessor functionality is already included in the current master
- Former image access methods have been marked as deprecated (See Bug 13260):
  - Image: GetData(), GetPixelValueBy…()
  - ImageDataItem: GetData()
  - LegacyAdaptors: CastToIpPicDescriptor()
- How to replace deprecated image access: http://www.mitk.org/development/ImageAccessorIntegration

Additional information

- Implicit use of image accessors
  - AccessByItk Macros
  - CastToItkImage
    - Holds accessor until itkImage Smartpointer is deleted

- Doxygen: Related Pages → Concepts → MITK Image

http://docs.mitk.org/nightly-qt4/MitkImagePage.html
#MitkImagePage_AccessImageData