ITK vs. VTK SmartPointer

Andreas Fetzer

02.10.2013

What is a SmartPointer in general

- Every `new` requires a `delete`
- A SmartPointer wraps a bare C++ pointer
- Helps to manage the lifetime of the object it is pointing to
- No explicit delete is neccessary
- Somehow similar to java garbage collection
  - Difference: With SmartPointers you can exactly determine when the object should be deleted

ITK SmartPointer

- Now `new itkObject()` is possible

- Increments reference count on assignment (overloaded „=“ operator)

```
itkObject::Pointer o = itkObject::New()
```

- Pitfalls:

```
itk::Image<int, 3>* imageptr = itk::Image<int,3>::New();
```

VTK SmartPointer

- vtkObjects increment reference count on creation!
- Assignment to a SmartPoint also increments reference count
- Examples:
  - `vtkPolyData* pd_ptr = vtkPolyData::New();`
    `pd_ptr->Delete();`
  - `vtkSmartPointer<vtkPolyData> pd =`
    `vtkSmartPointer<vtkPolyData>::New();`
  - `vtkSmartPointer<vtkPolyData> pd = vtkPolyData::New()`
    Reference count 2!
  - `pd.TakeReference(pd_ptr) or static ::Take`

Fazit

- Use SmartPointer as often as possible and as consistent as possible
  - An indicator would be how often your clients have to call delete manually
- Good practice since years for bigger projects