8/25/2010

VTK Debug Leaks

Diana Wald

dkfz.
GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

What is a memory leak?

- Memory leaks occur when a section of code allocates a block of memory that is never reclaimed.

```
vtkRenderWindow* vtkRenWin = vtkRenderWindow::New();

mitk::VtkPropRenderer::Pointer br = mitk::VtkPropRenderer::New( "testingBR" , vtkRenWin,
                                                                myRenderingManager);

mitk::BaseRenderer::AddInstance (vtkRenWin,br);

myRenderingManager->AddRenderWindow (vtkRenWin);

. . . .

myRenderingManager->RemoveRenderWindow (vtkRenWin);

vtkRenWin->Delete();
```

One way to create a VTK object is:

```
vtkObject* myObject = new vtkObject;
```

But you must manually delete this object, otherwise it leads to a memory leak.

```
myObject->Delete();
```

Memory Management

- All objects in VTK are reference counted
- VTK automatically delete an object when it is no longer needed (reference count of an object falls to zero)
- VTK does with via calls to `New()` / `Delete()` and `Register()` / `UnRegister()` to increase and decrease the number of references to an object

Advantage of reference counting in VTK is that it allows data to be shared instead of duplicated.

VTK Smart Pointer

- With SmartPointers, the reference count of an object is automatically increased when assigned to a SmartPointer and automatically decreased when unassigned from a SmartPointer.
- Objects are unassigned from a SmartPointer whenever that SmartPointer is assigned to another object or to 0. The latter occurs automatically whenever a SmartPointer goes out of scope.

```
#include <vtkSmartPointer>
vtkSmartPointer<vtkTransform> transform = vtkSmartPointer<vtkTransform>::New();
```

No Delete() needs to be called. However, be careful and note that we are allocating vtkObjects and NOT just vtkSmartPointer.

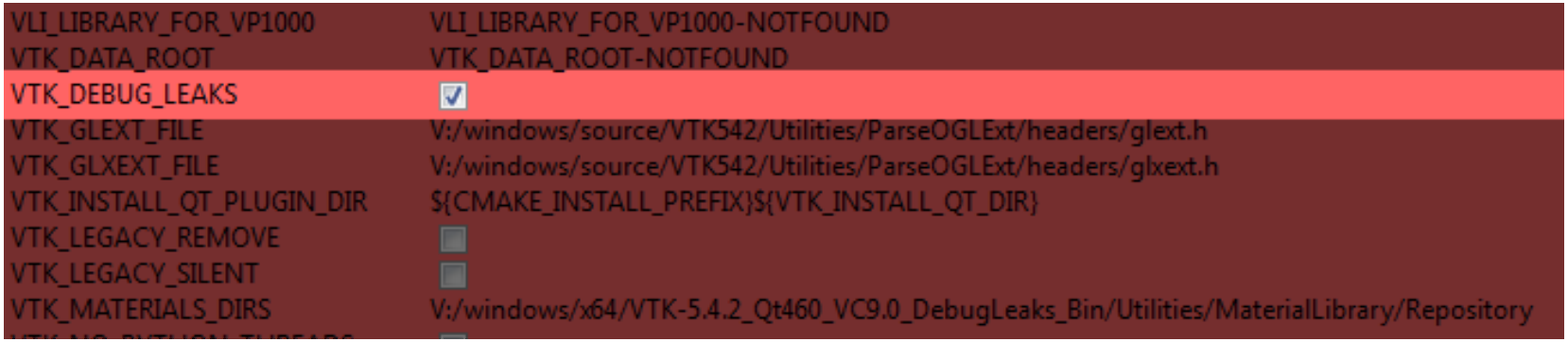
Never assign a raw object pointer to a smart pointer:

```
vtkSmartPointer<vtkTransform> transform = vtkTransform::New();
```

Reference count is increment and requiring an explicit Delete later

How to fix/find vtkDebugLeaks?

- Compile VTK with VTK_DEBUG_LEAKS switched on



- Start / Run Application or Test

- Output of all objects which were not deleted including information about the name of the vtkObject and the number of instances.

```
Class "vtkCellData" has 2 instances still around.
Class "vtkInformationIntegerVectorValue" has 3 instances still around.
Class "vtkInformationVector" has 9 instances still around.
Class "vtkPointData" has 2 instances still around.
Class "vtkTrivialProducer" has 1 instance still around.
Class "vtkPoints" has 9 instances still around.
Class "vtkInformation" has 15 instances still around.
Class "vtkLine" has 4 instances still around.
Class "vtkInformationStringVectorValue" has 1 instance still around.
Class "vtkInformationIntegerPointerValue" has 1 instance still around.
Class "vtkPolyData" has 1 instance still around.
Class "vtkPixel" has 2 instances still around.
Class "vtkIdList" has 8 instances still around.
Class "vtkDoubleArray" has 9 instances still around.
Class "vtkAlgorithmOutput" has 2 instances still around.
Class "vtkInformationIntegerValue" has 42 instances still around.
Class "vtkExtentTranslator" has 1 instance still around.
Class "vtkCellArray" has 2 instances still around.
Class "vtkVertex" has 1 instance still around.
Class "vtkImageData" has 1 instance still around.
Class "vtkInformationStringValue" has 2 instances still around.
Class "vtkFloatArray" has 3 instances still around.
Class "vtkInformationExecutivePortVectorValue" has 2 instances still around.
Class "vtkContourValues" has 1 instance still around.
Class "vtkIdTypeArray" has 2 instances still around.
Class "vtkShortArray" has 1 instance still around.
Class "vtkVoxel" has 1 instance still around.
Class "vtkFieldData" has 2 instances still around.
Class "vtkInformationDoubleVectorValue" has 2 instances still around.
Class "vtkStreamingDemandDrivenPipeline" has 2 instances still around.
Class "vtkMergePoints" has 1 instance still around.
Class "vtkMarchingCubes" has 1 instance still around.
Class "vtkInformationExecutivePortValue" has 2 instances still around.
```

Tips and Tricks

- Comment part of the code to find the position of the vtkDebugLeak

```
m_ResultNode->SetProperty("volumerendering",
                          mitk::BoolProperty::New(false));

vtkMarchingCubes* surfaceCreator = vtkMarchingCubes::New();
surfaceCreator->SetInput(m_ResultImage->GetVtkImageData());
surfaceCreator->SetValue(0, 1);

mitk::Surface::Pointer surface = mitk::Surface::New();
surface->SetVtkPolyData(surfaceCreator->GetOutput());

mitk::DataNode::Pointer surfaceNode = mitk::DataNode::New();
surfaceNode->SetData(surface);

m_DataStorage->Add(surfaceNode);

mitk::RenderingManager::GetInstance()->RequestUpdateAll();
```

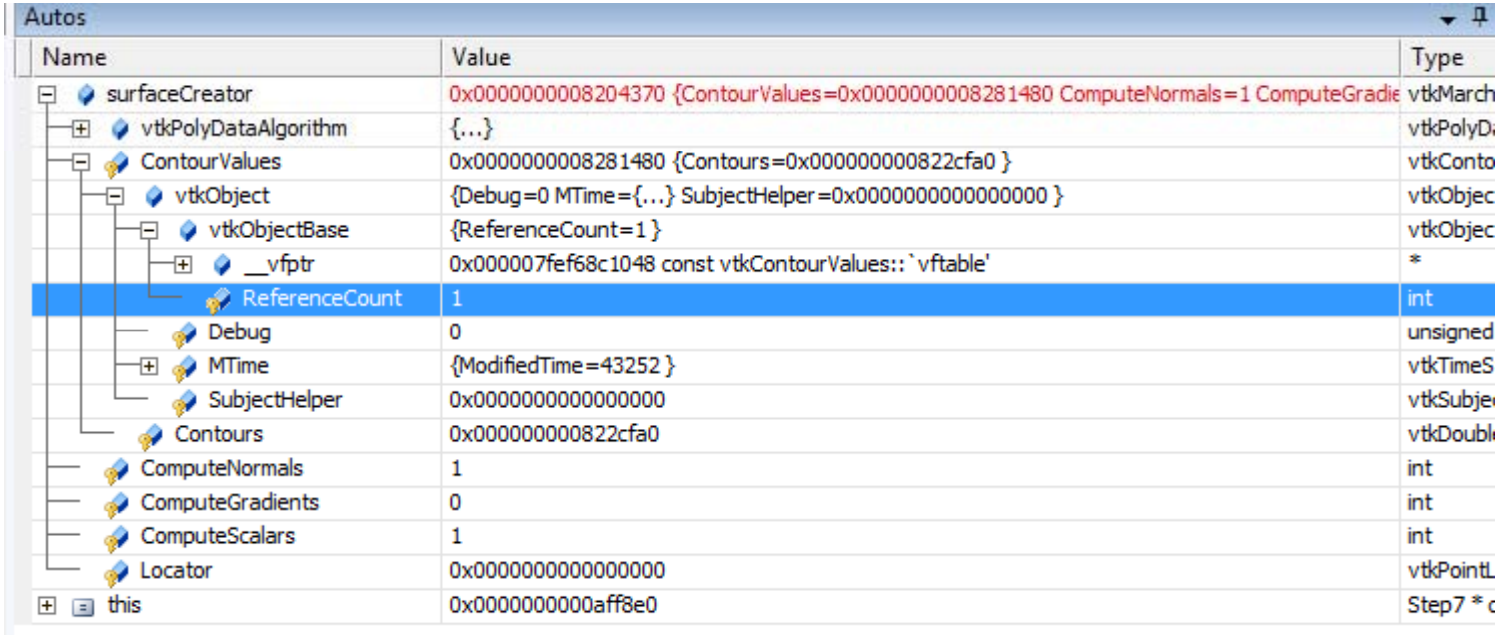
```
m_ResultNode->SetProperty("volumerendering",
                          mitk::BoolProperty::New(false));

// vtkMarchingCubes* surfaceCreator = vtkMarchingCubes::New();
// surfaceCreator->SetInput(m_ResultImage->GetVtkImageData());
// surfaceCreator->SetValue(0, 1);

// mitk::Surface::Pointer surface = mitk::Surface::New();
// surface->SetVtkPolyData(surfaceCreator->GetOutput());

// mitk::DataNode::Pointer surfaceNode = mitk::DataNode::New();
// surfaceNode->SetData(surface);
//
// m_DataStorage->Add(surfaceNode);

//mitk::RenderingManager::GetInstance()->RequestUpdateAll
```

```
m_ResultNode->SetProperty("volumerendering",
                          mitk::BoolProperty::New(false));

vtkMarchingCubes* surfaceCreator = vtkMarchingCubes::New();
// surfaceCreator->SetInput(m_ResultImage->GetVtkImageData());
// surfaceCreator->SetValue(0, 1);

// mitk::Surface::Pointer surface = mitk::Surface::New();
// surface->SetVtkPolyData(surfaceCreator->GetOutput());

// mitk::DataNode::Pointer surfaceNode = mitk::DataNode::New();
// surfaceNode->SetData(surface);
//
// m_DataStorage->Add(surfaceNode);

//mitk::RenderingManager::GetInstance()->RequestUpdateAll();
```

```
m_ResultNode->SetProperty("volumerendering",
                          mitk::BoolProperty::New(false));

vtkMarchingCubes* surfaceCreator = vtkMarchingCubes::New();
surfaceCreator->SetInput(m_ResultImage->GetVtkImageData());
surfaceCreator->SetValue(0, 1);

mitk::Surface::Pointer surface = mitk::Surface::New();
surface->SetVtkPolyData(surfaceCreator->GetOutput());

mitk::DataNode::Pointer surfaceNode = mitk::DataNode::New();
surfaceNode->SetData(surface);

m_DataStorage->Add(surfaceNode);

mitk::RenderingManager::GetInstance()->RequestUpdateAll();
surfaceCreator->Delete();
```

Tips and Tricks

- Check reference count of an object



- Use Valgrind (http://valgrind.org/)

Further Information

- VTK SmartPointer:
  http://www.itk.org/Wiki/VTK/Tutorials/SmartPointers