# A Full Example – Smoothing an image (bis_smoothimage.tcl)

July 10, 2009

## 1 Initial setup for each script

```
#!/bin/sh
# the next line restarts using wish \
    exec vtk "$0" "$@"


lappend auto_path [ file dirname [ info script ]]
lappend auto_path [file join [file join [ file dirname [ info script ]] ".." ] base]
lappend auto_path [file join [file join [ file dirname [ info script ]] ".." ] apps]
```

## 2 Class Definition

Each class needs at least three methods (in addition to the constructor). The Initialize methods is used to define the lists of inputs, outputs and options. This ends by calling the initialize method of its parent class which will append to these lists and then go on to initialize everything. The *GetGUIName* method simply gives the "English" name for the class. The *Execute* method is where the actual execution happens and where the algorithm methods are invoked.

```
package provide bis_smoothimage 1.0
package require bis_imagetoimagealgorithm 1.0

itcl::class bis_smoothimage {

    inherit bis_imagetoimagealgorithm

     constructor { } {      $this Initialize  }

    public method Initialize { }
    public method Execute { }
    public method GetGUIName    { } { return "Smooth Image" }
}
```

## 3 The Initialize Method

```
itcl::body bis_smoothimage::Initialize { } {
  #commandswitch,description,shortdescription,optiontype,defaultvalue,valuerange,priority
  set options {
   { blursigma "kernel size [mm/voxel] of FWHM filter size"
         "Filter Size"  { real triplescale 100 }    2.0 { 0.0 20.0 }  0 }
   { unit      "kernel size unit mm or voxels " "Units"
         { listofvalues radiobuttons }  mm { mm voxels }   1}
   { radius    "radius factor of the gaussian in voxel units "
         "Filter Radius" real  1.5   { 0.0 5.0 }  -1 }
   { dimension  "2 or 3 to to do smoothing in 2D or 3D"
          "Dimensionality"  { listofvalues radiobuttons }    3 { 2  3 }  -999 }
  }
  set defaultsuffix { "_sm" }
  set scriptname bis_smoothimage
  set completionstatus "Done"
  #
  # Documentation
  #
  set description "Smoothes an image with a specific gaussian kernel."
  set description2 "Smoothing kernel size blursigma (in mm) represents the FWHM filter size."
  set backwardcompatibility "Reimplemented from pxmat_smoothimage.tcl. unit and radius \
      options are added.Multiple image processing eliminated,which will be recovered \
      upon request. "
  $this InitializeImageToImageAlgorithm
}
```

## 4 The Execute Method

This is where the actual code is.

```
itcl::body bis_smoothimage::Execute {  } {
```

**Part 1 — get the parameters and inputs**

```
    set blursigma   [ $OptionsArray(blursigma) GetValue ]
    set unit        [ $OptionsArray(unit)      GetValue ]
    set radius      [ $OptionsArray(radius)    GetValue ]
    set dimension   [ $OptionsArray(dimension)    GetValue ]
```

**Next get the actual input image. This is of type pxitclimage**

```
    set image_in    [ $this GetInput ]
# To get the spacing first we need a pointer to the
# encapsulated vtkImageData obtained using the
# GetImage method of pxitclimage
    set spacing [[ $image_in GetImage ] GetSpacing ]
# Stuff to compute proper smoothness kernels if unit is voxels or mm
    if { $unit == "voxels"} {
     for { set j 0 } { $j <=2 } { incr j } {
         set sigma($j) [ expr $blursigma * 0.4247 / [ lindex $spacing $j ]]
     }
    } else {
     for { set j 0 } { $j <=2 } { incr j } {
         set sigma($j) [ expr $blursigma * 0.4247 ]
     }
    }
    set radiusz $radius
    if { $dimension == 2 } {
     set radiusz 0
     set sigma(2) 0.0
    }
```

**This is the actual VTK pipeline code:**

```
# Actual vtk code
    set smooth  [ vtkImageGaussianSmooth [ pxvtable::vnewobj ]  ]
    $smooth SetStandardDeviations $sigma(0) $sigma(1) $sigma(2)
    $smooth SetRadiusFactors $radius $radius $radiusz
    $smooth SetInput [ $image_in GetObject ]
    $this SetFilterCallbacks $smooth "Smoothing Image"
    $smooth Update
```

**Next we store the output.**

```
# When done store the output of the vtk pipeline in the Output Object
    set outimage [ $OutputsArray(output_image) GetObject ]
    $outimage ShallowCopyImage [ $smooth GetOutput ]
    $outimage CopyImageHeader [ $image_in GetImageHeader ]
# Add a comment to the image header (if NIFTI!) for later reference
    set comment [ format " [ $this GetCommandLine full ]" ]
    [ $outimage GetImageHeader ] AddComment "$comment $Log" 0
# Clean up
    $smooth Delete
    return 1
}
```

This checks if executable is called (in this case bis_smoothimage.tcl) if it is, then execute

```
if { [ file rootname $argv0 ] == [ file rootname [ info script ] ] } {
    # this is essentially the main function
    set alg [bis_smoothimage [pxvtable::vnewobj]]
    $alg MainFunction
}
```