Krita: Painting your Dreams

An overview of Krita 1.5

Bart Coppens

FOSDEM 2006

What is Krita

- Some new or enhanced features

A closer look

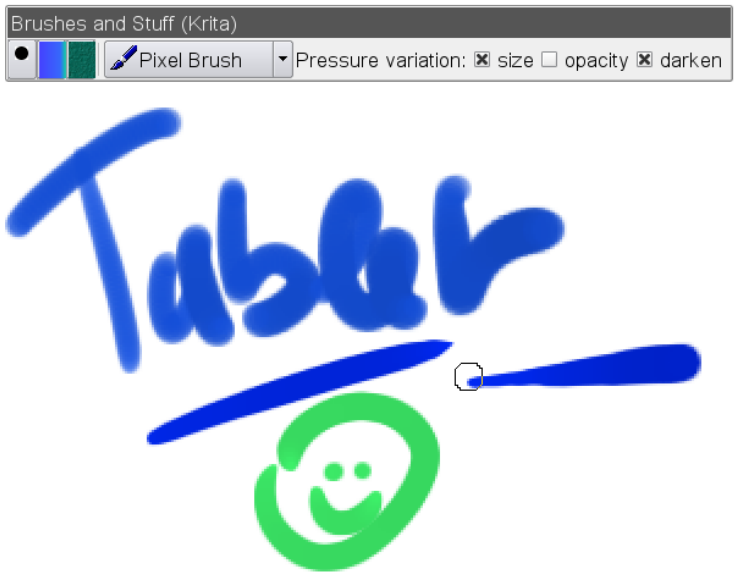
- Colorspaces
- Layers
- Plugins

What is Krita

- The KOffice painting and image editing application
- Started in 1999
- Previous release was with KOffice 1.4 last year
- KOffice 1.5, and thus the new Krita, are scheduled for a March release
- In Swedish, Krita means ‘chalk’ or ‘pencil’.
- `http://www.koffice.org/krita/`

Tablet support

- Supported size scaling for a long time
- For 1.5, you can also let color darkness and opacity depend on pressure
- We don't yet use advanced features like tilt, but it is supported for plugins in case they would use it.
- Make sure your Qt is configured with `-tablet`!



Colorspaces

A colorspace in Krita is a definition on what kind of data your painting contains. Is it a simple greyscale image, or has it millions of colors.
Two common ingredients: what do the channels represent, and how big is each channel?

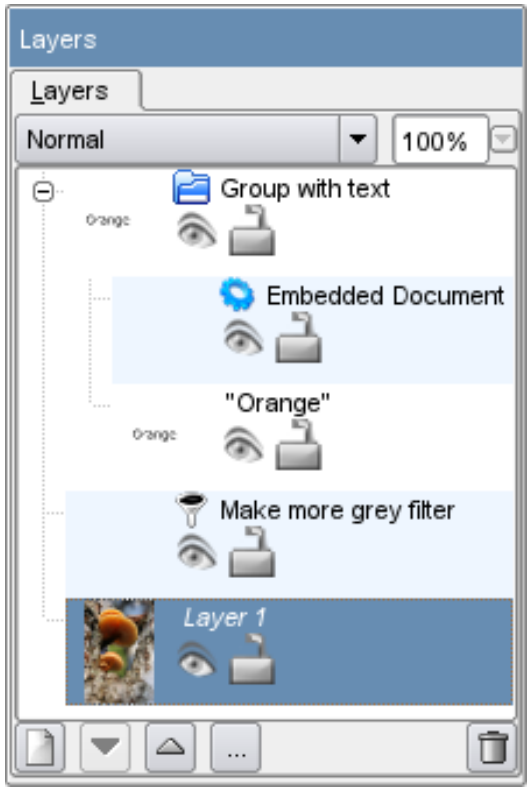
- Grey → 8 and 16 bits per channel
- RGB → 8 and 16 'regular', 16 and 32 bit floats
- CMYK → 8 and 16 bits per channel
- L*a*b* → 16 bits per channel
- Watercolors has fields like 'wetness'
- The Krita core makes it easy to extend it with new and exotic colorspaces

Profiles

- Different monitors produce the ‘same’ color differently
- Profiles try to minimize the difference you see
- Or do more exotic things, like swap channels
- Krita can load and save profiles embedded in images (like PNG and JPEG)
- Uses LittleCMS for the ‘regular’ colorspaces and profiles

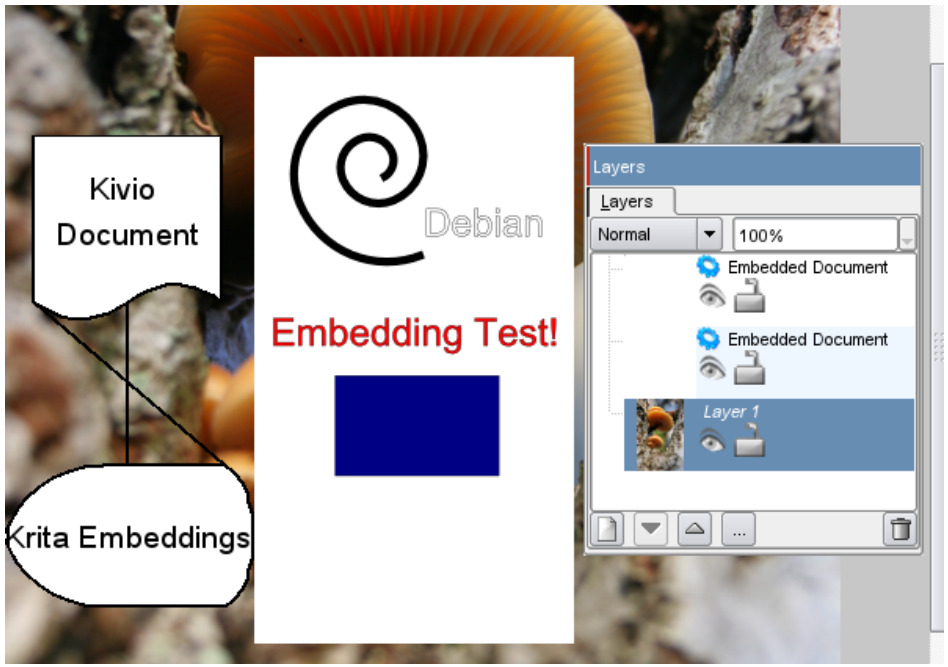
The layerbox

- Was completely rewritten, so less weird behaviour
- That caused some internal Krita changes, so more flexible layers system than before
- Now we have more eye candy (layer previews, informative tooltips, ...)



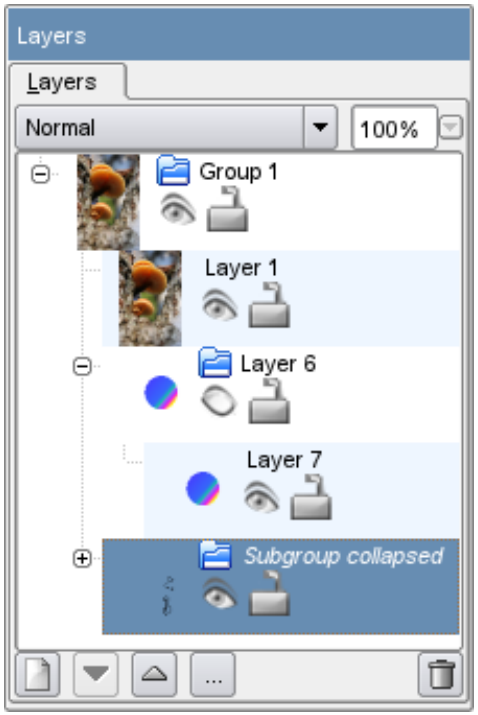
Parts layers

- Embed other KOffice parts into a Krita layer
- Typesetting a formula with KFormula
- A chart with KChart
- Some fancy text with KWord
- The possibilities are almost endless



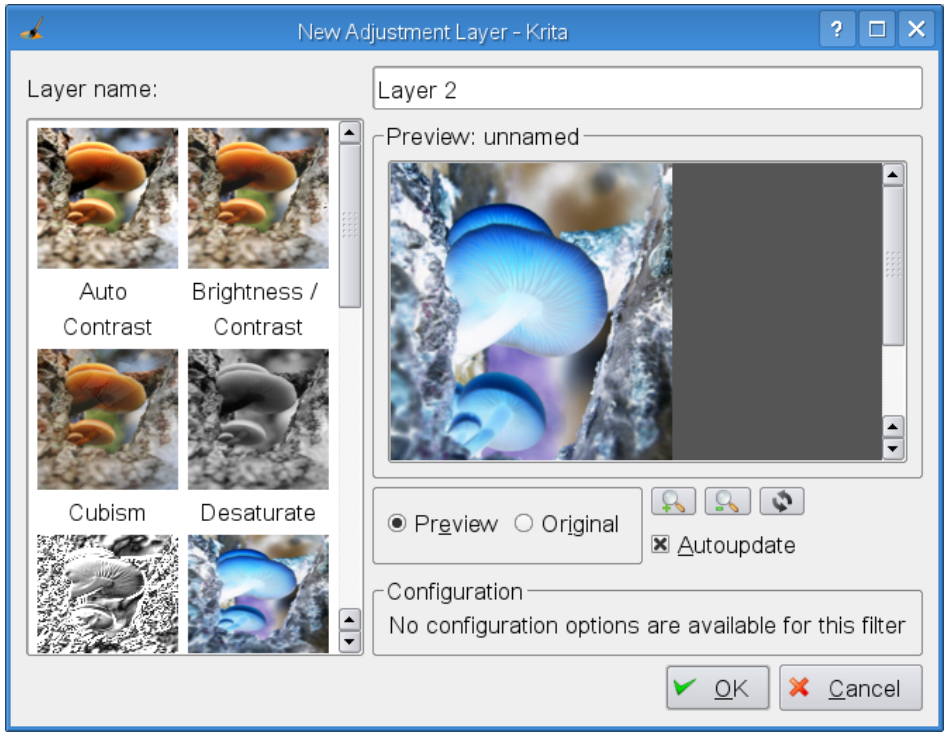
Group layers

- Groups layers into seperate logical unions
- They are easily minimized, so less intrusive
- Each has its own internal rendering, so it's better optimizable



Adjustment layers

- Apply filters to a layer, without losing your original picture
- Krita's implementation filters everything underneath the current filter layer
- To filter a single layer, put the target and the filter in a group



(Almost) everything is a plugin

In Krita, pretty much everything is a plugin. Most of the functionality you actually get to see in Krita, is being implemented in a plugin. For example

- Tools: rectangle, selection, crop image, color picker
- Paint Operations: brush, eraser, airbrush
- Filters: blur, color adjustment, brightness/contrast
- View Extensions: histogram docker, color chooser, scripting
- Colorspaces: every one except alpha and L*a*b*
- File Input/Output: every supported fileformat (except the native one)

Prototyping with scripts

- Krita has *experimental* scripting support
- Both Python and Ruby are supported right now, later perhaps KJSEmbed
- Uses a KOffice library shared with Kexi, called Kross
- It's not a complete representation of the Krita API, but it can be used to test things very rapidly
- It's experimental: there are no guarantees at all it'll work at all, or continue to have a stable API
- Yet we are fairly confident it will fit most purposes for testing and playing around

Python example: desaturate.py

```
class Desaturate:
 def __init__(self):
  try:
   import krosskritacore
  except:
   raise "Import of the KritaCore module failed."
  doc = krosskritacore.get("KritaDocument")
  script = krosskritacore.get("KritaScript")
  image = doc.getImage()
  layer = image.getActivePaintLayer()
  layer.beginPainting("Desaturate")
```

Python example: desaturate.py - continued

```
    it = layer.createRectIterator(
        0, 0, image.getWidth(), image.getHeight())
    while (not it.isDone()):
     p = it.getRGBA()
     mix = p[0] * 0.3 + p[1] * 0.59 + p[2] * 0.11
     p[0] = p[1] = p[2] = mix
     it.setRGBA(p)
     it.next()
    layer.endPainting()
Desaturate()
```

Extra plugins

- You can provide and maintain your own Krita plugins very easily outside the main repository
- The main example are Cyrille Berger's Krita plugins (color to alpha, blur and unsharp mask at the moment)
- They will be officially released seperately
- If you're not afraid of using bleeding edge software, you can (given you have the needed development software, headers, etc.) try them out yourself
- Beware, it's an unreleased SVN version (bugreports are appreciated, though)

Extra plugins - get the examples

Check out the plugins themselves, and go to the directory. The admin directory is needed as well, and then just do the usual routine to make it (srcdir != builddir):

```
$ svn co svn://anonsvn.kde.org/home/kde/trunk/
playground/graphics/krita-plugins/
$ cd krita-plugins
$ svn co svn://anonsvn.kde.org/home/kde/branches/
KDE/3.5/kde-common/admin
$ make -f Makefile.cvs
$ mkdir build
$ cd build
$ ../configure --enable-debug
$ make
$ su -c "make install"
$ krita
```

'The road goes ever on and on'

- The next release, 1.6, will be still Qt3-based
- In the meantime, Qt4-based KOffice 2.0 development will take place
- Krita 1.6 will probably focus on new plugins, rather than core changes
- Krita 2.0 will (probably) have more changes to the core and the architecture

And some actual Krita