Liven up your KDE wallpaper with SuperKaramba

Karamba on the Desktop

The desktop background – is it simply a place to display pretty pictures? No way! SuperKaramba allows you to re-vamp your KDE root window to provide a multifunctional tool. BY HAGEN HÖPFNER

If the desktop background didn't occupy a major part of the screen, there would be no need to worry about the amount of space wasted. Fortunately, there are some tools, such as *superkaramba* [12], that help remedy this unbalance.

This small K Desktop tool is based on an application called *karamba* [11], which it leverages to display system information on the desktop, and to remotely control media players such as XMMS [8] or *noatun* [6]. Karamba uses well-known pseudo-transparency effects in a similar way to applications like *konsole*. But you can use Python scripts to control SuperKaramba, in contrast to its predecessor.

Acting directly on the desktop, SuperKaramba addons do not hide any windows – if you have a screen full of applications running, you will need to move them aside to see the addons.

Curious?

The current version of the tool has binary packages for Suse, Mandrake, Red Hat, Debian and Slackware, and is available from [1]; Gentoo users can run the *emerge* command.

At this time of writing, users wanting to install the very latest version (0.33) had to download the sources, and run the usual three card trick: *./configure && make && make install* to compile and install. As this assumes pre-installed Qt and KDE header files, and a Qt 3.x binary compiled with thread support, you might prefer to use a ready-made binary. It was for this reason that we decided to base this article on version 0.32b.

When issued in a KDE command line window (*konsole*), the *superkaramba &* command will call a functional, but unspectacular, main menu (see Figure 1). The *Download...* item suggests that you might be able to download addons for the K Desktop background just by clicking. Unfortunately, the Web page that Konqueror opened when we did so failed to provide any themes at time of writing.

There are some themes available at [2], however. Our recommendation is to use the following two commands to unpack any themes archives that you download from this address in the hidden *~/.superkaramba* directory (if it does not exist, create it by typing *mkdir ~/.superkaramba*):

```
tar -C ~/.superkaramba/ -xvjf archivename.tar.bz2
tar -C ~/.superkaramba/ -xvzf archivenamegz
```

You can then select the *Open...* link in the main menu to open the definition file from the unpacked theme folder. The file will have a *.theme* suffix.

Figure 2 shows the "glassmachine" theme from [2], which includes a clock, the Kmenu, some application and system icons, and an XMMS control panel.

More cautious users might like to opt for the "Liquid Weather Plus" theme (see Figure 3) to display weather information on the background. Although this does require Python to be first installed. The dropdown menu available by right-clicking the display takes you to the *Configure Theme / Find location on weather.com* menu item, where you can enter your own location (such as *Liverpool, UK*). You can then click on *Reload Theme*, and a quick glance at the desktop will tell you if you need your umbrella today.

Figure 1: Any color if you want black – the SuperKaramba main menu.

Creating Your Own Theme

If you intend to create your own theme, ready-made themes are an endless source of ideas and information. Each of them has its own top-level directory with a control file (which ends in *.theme*). You can refer to [3,4,5] for details of what this directory should contain.

Figure 2: The glassmachine is simply another toolbox on the desktop background.

"Glassmachine" merely has the control file and a few icons. In contrast, "Liquid Weather Plus" adds another program file. As this theme needs to download weather data off the Internet, it needs more than just the *.theme* file. A Python script provides the added functionality. Don't worry, you can do without programming skills for your initial steps as SuperKaramba offers such a wide range of options for displaying system information and the like.

Figure 3: The weather, straight from your desktop.

Figure 4: Time, date and CPU load.

The *.theme* file content can be divided into two categories; the first category being commands that influence the general appearance of the theme. The other thing SuperKaramba can do is evaluate monitoring parameters.

Our first theme (see Listing 1) will display a simple digital clock, the current date, and the CPU load (see Figure 4). This theme is based on the glassmachine (see Figure 2).

The first thing we need is a main window – this is indicated by the *KARAMBA* keyword. The *X* and *Y* co-ordinates specify the number of points from the top left corner of the screen the top left corner of the window will be. *W* specifies the width, and *H* the height of the window. To allow users to move or scale the window on the desktop, you need to stipulate *LOCKED = FALSE*. The final option, *INTERVAL*, specifies the interval between window updates in milliseconds. As our clock has a seconds display, 1000 milliseconds seems like a sensible choice of update interval. Incidentally, capitalization is not significant for *.theme* files.

The second line in our example uses the *DEFAULTFONT* keyword to specify which font the theme should normally use. It is up to you to decide which of the optional parameters you will use here. In Listing 1, we opted for a Sans Serif type font (*FONT = "Sans"*) in white (*COLOR*) with a two-pixel shadow width (*SHADOW = 2*). You can overwrite any font parameters that you define here at a later stage.

SuperKaramba uses RGB values to define colors. The first value is red, the second green, and the third the blue channel of the desired color. If you set all three values to the maximum value, *255*, you end up with white. At the opposite end of the scale, *COLOR = 0,0,0* specifies black.

Now let's divide the display up into groups. The first group will display the time and date, and the second the CPU load. The *format* parameter specifies the layout of the date and time display. In our example, *"ddd MM.dd.yyyy"*, means that the abbreviated weekday (e. g. *Mon* for Monday) will be displayed, followed by the date in *month.day.year* format. If you stipulate *dddd* for the weekday, the full name of the weekday is displayed.

In the last line of the clock group, *CLICKAREA* defines an interactive area, where the parameters *X* and *Y* define the top left corner of the group relative to the main window, or to the parent group. *W* and *H* again specify the width and height, while the *ONCLICK = "..."* parameter specifies the command line (in double quotes) that a mouse click in this area will launch. In *LOCKED = FALSE* mode, this can be a simple left click; as we are using *LOCKED = TRUE* in this case (see line 1), we need to double-click to call the KDE module and configure the date, time and timezone.

Accessing System Parameters

We need two kinds of sensor to query the time, date, and CPU load: *SENSOR = TIME* provides the first two values, and *SENSOR = CPU* the CPU load. SuperKaramba has more parameters:

- *DISK* shows details of mounted drives, including **NFS** and **Samba** mounts.
- *MEMORY* displays the status of the main memory and **swap partition**.
- *NETWORK* provide details of network traffic.
- *NOATUN* tells us the audio track *noatun* [6] is currently playing.

Listing 1: Code for Figure 4

```
01 KARAMBA  x=0 y=0 w=140 h=65 locked=false interval=1000
02 DEFAULTFONT font="Sans" shadow=2 color=255,255,255
03
04 <GROUP> x=10 y=10
05         TEXT x=12 y=0 sensor=time fontsize=22 format="hh:mm:ss"
   interval=500
06         TEXT x=12 y=25 sensor=time fontsize=10 format="ddd
   MM.dd.yyyy"
07         CLICKAREA x=0 y=0 w=120 h=35 onclick="kdesu kcmshell clock"
08 </GROUP>
09
10 <GROUP> x=10 y=50
11         TEXT x=12 y=0 value="CPU"
12         TEXT x=42 y=0 sensor=cpu
13 </GROUP>
```

GLOSSARY

kdesu: *A KDE dialog box that prompts for the root password, and then runs the program specified in the argument passed to it (in this case: kcmshell clock) with superuser privileges.*

- *PROGRAM* displays the standard output of a program. For example

```
TEXT ... sensor=program program
="who | cut -d\  -f 1 |
sort | uniq"
```

would display logged on users.

- *SENSOR* uses the external *sensors* [7] tool to collect statistics.
- *TEXTFILE* continually reads a text file and outputs its content. For example:

```
TEXT ... sensor=textfile path
="/proc/acpi/thermal_zone/THRM
/temperature"
```

would display the current system temperature – assuming that the computer can provide this information.

- *UPTIME* displays the time since the machine was last booted.
- *XMMS* displays the audio track currently being played by *xmms* [8].

A full overview of these parameters is available from [4].

Eye-Candy

Displaying statistics in text-only formats will not win you any prizes for elegance. A level indicator would be useful for the CPU load. To enable this display, let's replace the last line for the second group in Listing 1 with the following code:

Figure 5: CPU load as a bar graphic...

Figure 6: ... and as a line graph.

Figure 7: XMMS remote control.

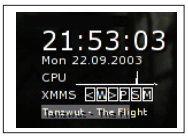

Figure 8: The complete example.

Table 1: General commands in .theme files

Command	Meaning	important parameters
KARAMBA	defines the main window	*X*: horizontal position (default: *x=0*)
		Y: vertical position (default: *y=0*)
		W: width (default: *w=300*)
		H: height (default: *h=300*)
		LOCKED: Disable (*true*) or enable (*false*) scaling/moving (default: *locked=false*)
		INTERVAL: Refresh interval in milliseconds (default: *interval=5000*)
CLICKAREA	defines an area which launches a program when clicked	*X*: horizontal position (relative to main window or parent group)
		Y: vertical position (relative to main window or parent group)
		W: width
		H: height
		PREVIEW: Should the user see the boundaries of the click area? If so, *preview=true* will draw a frame round it.
		ONCLICK: the program to launch
DEFAULTFONT	specifies the default font	*FONT*: Font family (example: *font="times"*). Use the *xfontsel* in *fmly* menu to discover the fonts installed on your system.
		COLOR: RGB color values for font (example: *color=255,0,0* for red)
		SHADOW: shadow width in points (example: *shadow=2*)
<GROUP>, *</GROUP>*	groups multiple elements	

```
BAR x=42 y=0 sensor=CPU
 path="bar.png"
```

We can use The Gimp [9], or another graphics package to create the *bar.png* graphic – in Figure 5 this is a white rectangle that measures 70x12 pixels. SuperKaramba displays a section of the graphic to reflect the current load status. The display starts in the top left corner of the image, and one pixel corresponds to one point on the level indicator.

If the graphic is bigger than the main window, or if you restricted the display with regard to width *w* or height *h*, SuperKaramba will automatically crop the graphic. The image file can use format-specific transparency effects.

If you do not specify a specific folder for *path*, SuperKaramba will assume the folder that contains the *.theme* file. Instead of an image file on disk, you can supply a *http* URL as the image path. To allow SuperKaramba to apply the changes to a running theme, you can right-click the display and select *Reload theme* in the dropdown.

If we now replace the line we just changed with the following:

```
GRAPH x=42 y=0 w=70 h=12 color
=255,255,255 points=100
sensor=cpu
```

SuperKaramba will draw a line graph of the CPU load as shown in Figure 6. *x*, *y*, *w* and *h* define the relative position of the display, and its width and height. You can use the *COLOR* keyword to define the color. The new *POINTS* parameter specifies the number of measuring points; this is 100 in our case. The tool will normalize this against the width of the graphic (*70* in our example).

It can interpret the return values as an image file, and display this file at a specified position, along the following lines:

GLOSSARY

Samba: *A software package that provides compatibility to Microsoft file and printer sharing functions, and allows Linux to access resources of this type. Linux with Samba can also provide file and/or print server functionality for Windows based network clients.*

Swap-Partition: *An area on the hard disk, that (particularly) computers with little physical memory (RAM) use as an additional, virtual memory area to handle memory intensive applications. Swapping main memory out to disk, is a slow process.*

who | cut -d\ -f1 | sort | uniq: *Shows use of the traditional Unix pipeline (|). The who command tells you who is logged on. The cut command cuts the first field with the username out of this output (-f1), using the space character to delimit the columns (-d). The space character needs to be escaped with a backslash \. The sort command sorts the output alphabetically, and uniq removes any duplicates, ensuring that each user appears only once.*

NFS: *The "Network File System" allows transparent mounting of directories on remote Unix/Linux servers within the local filesystem tree.*

Table 2: Important parameters for the BAR level display

Parameter	Meaning
x	horizontal position of top left corner (relative to main window or parent group)
y	vertical position of bottom left corner (relative to main window or parent group)
w	width (optional; if not supplied, SuperKaramba will default to the width of the image)
h	height (optional; height of image by default)
PATH	path to image file
VERTICAL	display bottom/up instead of left/right (default: *vertical=false*)

```
IMAGE x=numeric y=numeric w=width h=height path="path/to/or/URL/of/imagefile"
```

This function is useful for adding webcam images to a theme. Refer to [5] for more good ideas.

More Interaction

As previously mentioned, SuperKaramba is not only capable of displaying system information, but can also remotely control your applications – for example XMMS [8], as shown in Figure 7. To allow this to happen, let's add the code from Listing 2 to Listing 1.

In addition to adding the new group, which will contain buttons for remotely controlling XMMS that use labels and highlighted areas (*preview = true*), we need to adjust the height of the main window specified in the *KARAMBA* line of Listing 1 (*H = 100*).

A click on the < button jumps back to the previous track in the playlist, as specified by the *ONCLICK = "xmms --rew"* command. *W* will start to play (*xmms --play*), > will fast forward to the next track (*--fwd*), and the pause button, *P*, will temporarily pause the current track (*--play-pause*). *S* stops the player (*--stop*), and *M* launches the KDE mixer *kmix* to allow you to adjust the volume control. This uses the *dcop kmix kmix-mainwindow#1 show* [13,14] command.

If you take a closer look at the "glassmachine" theme, you will note that the position within the current track is indicated by a status bar. Also, the title and artist are read and displayed. We can do that, too, as Figure 8 shows! Let's create an image file called *bar1.png* for the status display. The file will contain a gray rectangle that measures 96x12 pixels. We also need to add the following line to extend the group defined in Listing 2:

```
BAR  X=12 Y=15 SENSOR=XMMS format="%ms" path="bar1.png" interval=500
```

Listing 2: Adding XMMS controls

```
01 <GROUP> X=10 Y=65
02         text x=12  y=0 value="XMMS"
03         text x=50  y=0 value="<"
04         text x=60  y=0 value="W"
05         text x=70  y=0 value=">"
06         text x=80  y=0 value="P"
07         text x=90  y=0 value="S"
08         text x=100 y=0 value="M"
09
10         CLICKAREA x=50  y=0 w=9 h=12 preview=true SENSOR=PROGRAM
   ONCLICK="xmms --rew"
11         CLICKAREA x=60  y=0 w=9 h=12 preview=true SENSOR=PROGRAM
   ONCLICK="xmms --play"
12         CLICKAREA x=70  y=0 w=9 H=12 preview=true SENSOR=PROGRAM
   ONCLICK="xmms --fwd"
13         CLICKAREA x=80  y=0 w=9 H=12 preview=true SENSOR=PROGRAM
   ONCLICK="xmms --play-pause"
14         CLICKAREA x=90  y=0 w=9 H=12 preview=true SENSOR=PROGRAM
   ONCLICK="xmms --stop"
15         CLICKAREA x=100 y=0 w=9 H=12 preview=true SENSOR=PROGRAM
   ONCLICK="dcop kmix kmix-mainwindow#1 show"
16 </GROUP>
```

The *%ms* format parameter, which unfortunately is not officially documented, causes the bar display for the XMMS sensor to grow while the track is playing, thus indicating the current position within the track.

So, now we are just missing the title and artist display. Again, we only need to add a single line to retrieve the appropriate data from the XMMS sensor:

```
TEXT x=13 y=16 w=94 H=10 SENSOR=XMMS fontsize=8 font="Sans" color=255,255,255 format="%title"
```

format = "%title" returns both the artist and the title in this case. The complete theme, *small_text_xmms.theme* is available as a packed tar archive, *small_text_xmms.tar.bz2*, from [10].

If you are raring to flaunt your Python skills to define addons for themes, don't worry. I'll be back soon. ■

INFO

[1] SuperKaramba binary packages: *http://netdragon.sourceforge.net/?page=Download+SuperKaramba*

[2] SuperKaramba themes: *http://www.kde-look.org/index.php?xcontentmode=karamba*

[3] General *.theme* parameters: *http://netdragon.sourceforge.net/?page=General+Commands*

[4] *.theme* sensors: *http://netdragon.sourceforge.net/?page=Sensors*

[5] *.theme* meters: *http://netdragon.sourceforge.net/?page=Meters*

[6] The KDE media player, noatun: *http://noatun.kde.org/*

[7] Linux system hardware monitoring: *http://secure.netroedge.com/~lm78/*

[8] X Multimedia System: *http://www.xmms.org/*

[9] The GIMP: *http://www.gimp.org/*

[10] Sample theme: *http://wwwiti.cs.uni-magdeburg.de/~hoepfner/download.html*

[11] Karamba, the original: *http://www.efd.lth.se/~d98hk/karamba/*

[12] SuperKaramba: *http://netdragon.sourceforge.net/*

[13] Patricia Jung: "Hidden but sooo useful!", Linux Magazine 9/2003, Issue 34 p40, *http://www.linux-magazine.com/issue/34/KDETricks.pdf*

[14] Scott Wheeler: "Boost your efficiency", Linux Magazine 11/2003, Issue 36 p46, *http://www.linux-magazine.com/issue/36/KDE_Scripting_DCOP.pdf*