Inhaltsverzeichnis

Topflappen häkeln mit Unix

Wie kriegt man sein Lieblingslogo in die Küche? Als Topflappen natürlich, und die Vorlage macht man natürlich mit Unix:

So'n Lappen hat normalerweise ungefähr 50×50 Felder, also rechnet man sich das Bild erst auf 50 Pixel klein, und dann wieder groß mit etwas Rand. Dabei reduziert man noch die Anzahl der Farben

```
 convert oobd_logo_simple_colors.png  -resize 50   -filter point -flatten -
alpha off -resize 500 -blur 0 -gravity center -extent 600x600 -colors 4
oobd_new.png
```

bzw. man dreht es um 45°

```
  convert oobd_logo_simple_colors.png  -rotate "45" -resize 50   -filter
point -flatten -alpha off -resize 500 -blur 0 -gravity center -extent
600x600 -colors 4 oobd_new.png
```

Dann bedient man sich noch des großartigen Scripts von Fred Weinhaus, um ein Rasternetz zum besseren Abzählen darüber zu bekommen, und fertig. Jetzt muß die Vorlage nur noch zu

Grossmuttern in die Produktion 🙂

grid.sh

```
#!/bin/bash
#
# Developed by Fred Weinhaus 10/22/2007 .......... revised 1/1/2008
#
# USAGE: grid [-s spacing] [-c color] [-t thickness] [-o opacity]
infile outfile
# USAGE: grid [-h or -help]
#
# OPTIONS:
#
# -s        spacing       x,y spacing between grid lines; default=16,16
or 16;
#                         second number is defaulted to the first
# -c        color         color of grid lines; default="black"
# -t        thickness     thickness of grid lines; default=1
# -o        opacity       opacity of grid lines opacity between 0.0 and
1.0;
#                         opacity=0 is transparent; opacity=1 is
opaque;
#                         default=1
#
###
#
# NAME: GRID
```

```
#
# PURPOSE: To superimpose a set of horizontal and/or vertical grid
lines
# on an image.
#
# DESCRIPTION: GRID superimposes a set of horizontal and/or vertical
grid
# lines on an image. Parameters are available to select the grid line
color,
# thickness and opacity.
#
#
# OPTIONS:
#
# -s spacing ... SPACING specifies the horizontal (x) and vertical (y)
grid
# offset between lines. Spacing must be provided as integer values. If
the
# second value is left off, then it will be set identical to the first.
If
# only vertical lines are desired, set the horizontal (x) spacing
larger
# than the width of the image. If only horizontal lines are desired,
set
# the vertical (y) spacing larger than the height of the image. The
default=8.
# Note: if you want the bottom and/or right grid line to show, then the
image
# dimension(s) must be a multiple of the grid spacing plus 1.
#
# -c color ... COLOR is the color of the grid lines. Any valid IM color
# specification is allowed. Be sure to color values in double quotes.
# The default="black".
#
# -t thickness ... THICKNESS is the grid line thickness. Values are
positive
# integers. The default=1
#
# -o opacity ... OPACITY is the grid line opacity. Values are non-
negative
# floats between 0.0 and 1.0. The default=1
#
# CAVEAT: No guarantee that this script will work on all platforms,
# nor that trapping of inconsistent parameters is complete and
# foolproof. Use At Your Own Risk.
#
######
#

# set default values
spacing=16
```

```
color="black"
thickness=1
opacity=1


# set directory for temporary files
dir="."    # suggestions are dir="." or dir="/tmp"


# set up functions to report Usage and Usage with Description
PROGNAME=`type $0 | awk '{print $3}'`  # search for executable on path
PROGDIR=`dirname $PROGNAME`            # extract directory of program
PROGNAME=`basename $PROGNAME`          # base name of program
usage1()
    {
    echo >&2 ""
    echo >&2 "$PROGNAME:" "$@"
    sed >&2 -n '/^###/q;  /^#/!q;  s/^#//;  s/^ //;  4,$p'
"$PROGDIR/$PROGNAME"
    }
usage2()
    {
    echo >&2 ""
    echo >&2 "$PROGNAME:" "$@"
    sed >&2 -n '/^######/q;  /^#/!q;  s/^#*//;  s/^ //;  4,$p'
"$PROGDIR/$PROGNAME"
    }


# function to report error messages
errMsg()
    {
    echo ""
    echo $1
    echo ""
    usage1
    exit 1
    }


# function to test for minus at start of value of second part of option
1 or 2
checkMinus()
    {
    test=`echo "$1" | grep -c '^-.*$'`   # returns 1 if match; 0
otherwise
    [ $test -eq 1 ] && errMsg "$errorMsg"
    }

# test for correct number of arguments and get values
if [ $# -eq 0 ]
```

```
    then
    # help information
   echo ""
   usage2
   exit 0
elif [ $# -gt 10 ]
    then
    errMsg "--- TOO MANY ARGUMENTS WERE PROVIDED ---"
else
    while [ $# -gt 0 ]
        do
            # get parameter values
            case "$1" in
          -h|-help)    # help information
                       echo ""
                       usage2
                       exit 0
                       ;;
              -s)      # get spacing
                       shift  # to get the next parameter - spacing
                       # test if parameter starts with minus sign
                       errorMsg="--- INVALID SPACING SPECIFICATION ---"
                       checkMinus "$1"
                       spacing="$1,"
                       ;;
              -c)      # get color
                       shift  # to get the next parameter - lineval
                       # test if parameter starts with minus sign
                       errorMsg="--- INVALID COLOR SPECIFICATION ---"
                       checkMinus "$1"
                       # test lineval values
                       color="$1"
                       ;;
              -t)      # get thickness
                       shift  # to get the next parameter - thickness
                       # test if parameter starts with minus sign
                       errorMsg="--- INVALID THICKNESS SPECIFICATION --
-"
                       checkMinus "$1"
                       # test width values
                       thickness=`expr "$1" : '\([0-9]*\)'`
                       [ "$thickness" = "" -o $thickness -eq 0 ] &&
errMsg "--- THICKNESS=$thickness MUST BE A POSITIVE INTEGER ---"
                       ;;
              -o)      # get opacity
                       shift  # to get the next parameter - opacity
                       # test if parameter starts with minus sign
                       errorMsg="--- INVALID OPACITY SPECIFICATION ---"
                       checkMinus "$1"
                       # test width values
                       opacity=`expr "$1" : '\([.0-9]*\)'`
```

```
                        [ "$opacity" = "" ] && errMsg "OPACITY=$opacity
IS NOT A NON-NEGATIVE FLOATING POINT NUMBER"
                        opacitytest=`echo "$opacity > 1" | bc`
                        [ $opacitytest -eq 1 ] && errMsg
"OPACITY=$opacity MUST BE BETWEEN 0.0 AND 1.0"
                        ;;
                    -)  # STDIN and end of arguments
                        break
                        ;;
                   -*)  # any other - argument
                        errMsg "--- UNKNOWN OPTION ---"
                        ;;
                    *)  # end of arguments
                        break
                        ;;
            esac
            shift   # next option
    done
    #
    # get infile and outfile
    infile=$1
    outfile=$2
fi


# test that infile provided
[ "$infile" = "" ] && errMsg "NO INPUT FILE SPECIFIED"

# test that outfile provided
[ "$outfile" = "" ] && errMsg "NO OUTPUT FILE SPECIFIED"

# setup temporary images and auto delete upon exit
# use mpc/cache to hold input image temporarily in memory
tmpA="$dir/profile_$$.mpc"
tmpB="$dir/profile_$$.cache"
trap "rm -f $tmpA $tmpB; exit 0" 0
trap "rm -f $tmpA $tmpB; exit 1" 1 2 3 15


#
if convert -quiet -regard-warnings "$infile" +repage "$tmpA"
    then
    : 'do nothing - continue processing below'
    else
        errMsg "--- FILE $infile DOES NOT EXIST OR IS NOT AN ORDINARY
FILE, NOT READABLE OR HAS ZERO SIZE ---"
fi


# get image dimensions
width=`identify -format %w $tmpA`
```

```
height=`identify -format %h $tmpA`

width1=`expr $width - 1`
height1=`expr $height - 1`

xinc=`echo "$spacing" | cut -d, -f1`
yinc=`echo "$spacing" | cut -d, -f2`
[ "$yinc" = "" ] && yinc=$xinc
testx=`expr $xinc : '[0-9]*'`
testy=`expr $yinc : '[0-9]*'`
[ $testx -eq 0 -o $testy -eq 0 ] && errMsg "--- SPACING MUST BE AN
INTEGER ---"

# get string for drawing grid lines
drawstr=""

if [ $yinc -le $height ]
    then
    i=0
    while [ $i -le $height ]
        do
        drawstr="$drawstr M 0,$i L $width1,$i"
        i=`expr $i + $yinc`
    done
fi

if [ $xinc -le $width ]
    then
    i=0
    while [ $i -le $width ]
        do
        drawstr="$drawstr M $i,0 L $i,$height1"
        i=`expr $i + $xinc`
    done
fi

# process image
convert $tmpA -fill none -stroke $color -strokewidth $thickness -draw
"stroke-opacity $opacity path '$drawstr'" $outfile
exit 0
```