Netpbm Tools and Shell Scripts

Animation on Demand

Editing images does not mean you will automatically need a mouse. The filters included in the Netpbm package and similar tools can be used in shell scripts to automate various steps. BY CHRISTIAN PERLE

Pixel based graphic formats have been around for some years. Consequently, there has been a similar demand for suitable conversion programs. If you want tackle this problem Unix style, that is using individual filter programs for the command line, you will need to write exactly *(n-1)*n* filters for *n* graphic formats. If you decide to use an interim format instead, you will only need *n* filters to convert various graphic formats into the interim format and another *n* to convert the interim format back to an original graphic format.

Jef Poskanzer started working on *pbmtools* in 1989 with this in mind. Up until 1994 new formats and effect filters for the interim format were added step by step to the collection now known as *netpbm*. Since 2000 the Netpbm project has seen a return to more active development and this is now hosted by the **Sourceforge** project.

GLOSSARY

Sourceforge: *A Web service for Open Source projects including developer forums, version control, download areas, and various other resources at http://sourceforge.net/.*

Povray: *A freeware raytracing (3D graphics) program that runs on various operating systems – such as Linux. The Povray homepage is available at http://www.povray.org/; the subscription CD includes a tutorial in HTML format.*

Anti-aliasing: *Automatic smoothing of lines in high contrast images. Prevents an image from appearing "over-pixeled" and simulates a higher resolution.*

Bit, Grey or Pix?

To be more exact, Netpbm does not offer a single interim format but three: "*P*ortable *Bitm*ap" (PBM), "*P*ortable *Greym*ap" (PGM) and "*P*ortable *Pixm*ap" (PPM). The PBM format recognizes only occupied (black) and vacant (white) pixels and thus requires one bit per pixel. The PGM format can store only greyscales and will normally require eight bits per pixel (256 greyscales). The PPM format requires 24 bits per pixel (eight bits each for the base colors red, green, and blue), allowing 16.7 million colors ("true color"). "*P*ortable *Anym*ap" (PNM) refers to any interim format.

tgatoppm, *giftopnm*, or *g3topbm* are examples of the filters used to convert external formats to the interim format. *ppmtogif*, *pnmtotiff*, or *pnmtops* are examples of formats for the opposite direction. Additionally, there are some formats that are applied only to the interim formats. *ppmtopgm* converts images to greyscales, *pnmsmooth* applies a soft focus effect and *pgmnorm* is used for normalizing greyscales.

Source Material

Of course, raw material is required for command-line based image processing. But your computer can also take care of this task using the freeware raytracer, **Povray**. The scene description file *glass.pov* from Listing 1 causes the program to trace an image with five glass balls on a checkered background. You can use the *clock* variable to move the background and in turn produce a simple animation.

Now let's feed this to Povray version 3.0 or 3.1. We want the program to create a 320 by 240 pixel image using **anti-aliasing**, and creating output in PPM format:

```
povray +i glass.pov +w320 +h240 ➥
+a0.1 +fp +v
```

Figure 1 shows the results as stored in the *glass.ppm* file.

Filters in Chains

The following steps show how to use filter commands to modify an image. The following command converts the image to greyscale:

```
ppmtopgm glass.ppm > glass.pgm
```

Just like the filters in the Netpbm

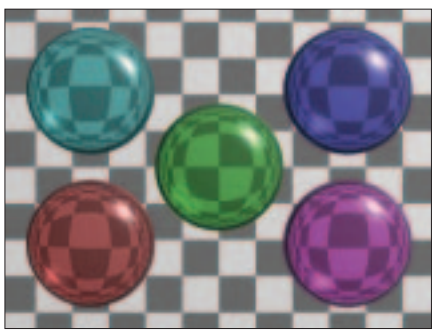

Figure 1: Glass balls as a test image

Listing 1: glass.pov

```
01 // Glass ball animation
02 // (C) 11/2002 Christian Perle
   (POVaddict) / Linux Magazine
03
04 // Camera
05 camera {
06   location <0, 0, -10>
07   direction <0, 0, 4>
08   look_at <0, 0, 0>
09 }
10
11 // Lighting
12 light_source { <10, 10,
   -10> color rgb<1, 1, 1> }
13
14 // Declaration of glass ball
15 #declare GBall = sphere {
16   <0, 0, 0>, 0.5
17   scale <1, 1, 0.5>
18   finish { phong 0.7
   reflection 0.1 refraction
   1 ior 1.33 }
19 }
20
21 // Five colored glass balls
22 object {
23   GBall
24   translate <-1, -0.6, 0>
25   pigment { rgbf<1, 0.7,
   0.7, 0.7> }
26 }
27 object {
28   GBall
29   translate <0, 0, 0>
30   pigment { rgbf<.7, 1,
   .7, .7> }
31 }
32 object {
33   GBall
34   translate <1, 0.6, 0>
35   pigment { rgbf<0.7, 0.7,
   1, 0.7> }
36 }
37 object {
38   GBall
39   translate <1, -0.6, 0>
40   pigment { rgbf<1, 0.7,
   1, 0.7> }
41 }
42 object {
43   GBall
44   translate <-1, 0.6, 0>
45   pigment { rgbf<0.7, 1,
   1, 0.7> }
46 }
47
48 // Checkered pattern in
   background
49 plane {
50   <0, 0, -1>, -4
51   pigment {
52     checker color rgb<0.5,
   0.5, 0.5>, rgb<1, 1, 1>
53     translate <-clock, -
   clock, 0>
54     scale 0.4
55   }
56   finish { ambient 0.4 }
57 }
```

package, *ppmtopgm* sends the results to your **standard output**, which can be redirected to an new file called *glass.pgm* using a greater than sign " > ". Figure 2 shows the results.

In order to avoid creating additional temporary files for the following filtering steps, we now make use of the fact that the Netpbm tools can read from **standard input**. This allows us to link a series of filters using **pipes** in the shell:

```
ppmtopgm glass.ppm | pgmoil ➥
-n 5 | pgmtoppm Blue-White > ➥
oil.pgm
```

The output from *ppmtopgm* is sent directly to *pgmoil*. This filter adds an effect to the image, making the contures appear to melt just like in an oil painting. The *pgmoil* option *-n 5* tells the filter to apply the oil effect to fields measuring 5 by 5 pixels.

To bring back some color to the image, we now add another filter to the chain. *pgmtoppm* converts the greyscales to blue and white, as is shown in figure 3.

Learning to Run

Of course, there is nothing wrong with calling individual filters in the command-line, but you will probably need a **shell script** to make use of the power of most command-line tools. Shell scripts allow you to send a whole collection of image files through the same chain of filters, or simply convert them to a different format.

So now all we need is a whole bunch of images to experiment on. Let's ask our old friend Povray to help us out with that animation feature we just talked about. You can use the following command:

```
povray +i glass.pov +w240 +h180➥
+fp +a0.1 +kfi00 +kff49 +kc
```

to have the raytracer produce an animation sequence with 50 images. The images are stored as *glass00.ppm*, *glass01.ppm*, through *glass49.ppm*. Correspondingly, the options *+kfi* and *+kff* refer to the numbers of the first and last images. The option *+kc* shows that this is a cyclical animation.

The next task is to create an animated GIF image from the individual images

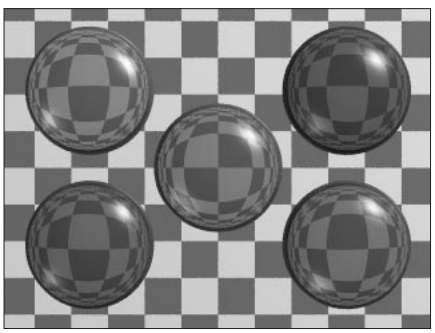

Figure 2: Everything turned grey all of a sudden

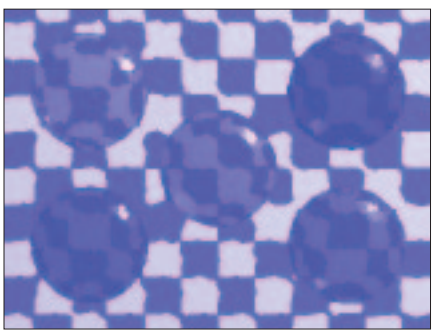

Figure 3: Blue and white colouring with oil effect

GLOSSARY

Standard input, standard output: *Many command-line programs allow you to omit the name of the input file. In this case the program reads from standard input, which will normally mean the keyboard. If you omit the name of the output file, most programs will use standard output, that is display the results on your terminal*

Pipe: *The pipe character "|" (representing a stylized pipeline) connects the standard output of a program to the standard output of another program. This allows you to use multiple programs in a single processing step.*

Shell script: *A file containing shell commands that are processed automatically. Repetitive tasks are often best accomplished using automated shell scripts.*

Listing 2: mkgifanim

```
#!/bin/bash
for f in glass??.ppm
do
 pnmscale -w 100 $f | ppmquant ➥
256 | ppmtogif > ${f%.ppm}.gif
done
whirlgif -o g_anim.gif -loop ➥
-time 8 glass??.gif
```

Listing 3: mkedge

```
#!/bin/bash
for f in glass??.ppm
do
  ppmtopgm $i | pgmedge | ➥
pgmnorm > ${i%.ppm}.pgm
done
ls glass??.pgm > frames.list
ppm2fli -g240x180 frames.list ➥
edge_anim.fli
```

and incorporate the GIF in a website. To prevent the GIF image from becoming too large you might decide to scale down the image to 100 by 75 pixels. The shell script *mkgifanim* (Listing 2) takes care of this task and goes on to call the *whirlgif* tool that assembles the individual GIF images to an animated GIF.

In the *for* loop, the variable *f* parses any file names that match the shell expression (see box below) *glass??.ppm*. Within the filter chain *pnmscale* is used to scale down the individual images to a width of 100 pixels. The height is calculated automatically to retain the original proportions. In the next step *ppmquant* reduces the number of colors in the GIF to a maximum of 256. Finally, *ppmtogif* writes the GIF image itself. The script uses the current value of the variable *f* to construct a file name, removing the *.ppm* suffix and adding *.gif* as the new suffix.

The following *whirlgif* call will run the animation, *g_anim.gif*, in an infinite loop (using the *-loop* option), with an interval of 8 milliseconds between the images *-time 8*. Figure 4 shows the animation – of course you can only run the animation, if you purchased the flip-book plug-in for this issue. But seriously folks, check out the subscription CD for the file, which you can view in your web browser or *xanim*.

Animated Effects

In addition to GIF there are a few patented animation formats, such as *MPEG* and *FLI*. You can use *xanim* to view the latter. The *mkedge* animation in Listing 3 requires *ppm2fli*, which is not a Netpbm tool.

This script also processes all the individual images in the glass ball sequence, converting them to greyscale (*ppmtopgm*) and creating lines for the edges in the image (*pgmedge*) and normalizing the brightness (*pgmnorm*).

The resulting images are named *glass00.pgm* through *glass49.pgm*. Since *ppm2fli* expects a list of the individual images in a file, you will need to run *ls* to create this list, before you launch *ppm2fli*. You will also need to use the option *-g* to tell the tool the image formats in use.

You can use the *ffmpeg* tool to create a further animation – as the name would suggest, an MPEG. You can apply the filter chain used for *mkedge* first, however, you will need an additional step (*pnmarith*) to add the original image *glass00.ppm* pixel by pixel. This creates an interesting overlay effect. When *ffmpeg* is called in *mkedge* (Listing 4), the expression for the input file (*glass%02d.overlay.ppm*) is expanded by *ffmpeg* itself.

You might like to perform some experiments of your own, and read the man pages, to familiarize yourself with the range of filters provided by the Netpbm tools. The man pages for *pbm*, *pgm*, *ppm*, and *pnm* contain an overview. ■

Listing 4: mkoverlay

```
#!/bin/sh
for i in glass??.ppm
do
  ppmtopgm $i | pgmedge | ➥
pgmnorm > temp.ppm
 pnmarith -add temp.ppm glass➥
00.ppm > ${i%.ppm}.overlay.ppm
done
ffmpeg -an -i
glass%02d.overlay.ppm -b 768 ➥
g_overlay.mpg
```

Shell Patterns

The shell recognizes various expressions for file and directory names, and expands them before running the current command. The most important examples are

- the question mark ?, which represents exactly one character.
- the asterisk *, which is a wildcard for any number of random characters (even zero).
- character counts in square brackets []. Exactly one character of the type in the brackets must occur at this position. There are various notations as is evidenced by the following examples: The expression lx[acE].txt matches the names lxa.txt, lxc.txt, and lxE.txt.

graph[a-z][0-9][0-9].jp? matches the names *graphi50.jpg*, *graph001.jpg*, and *graphx55.jpe* (amongst others). The dash within the brackets indicates that a complete character set is to be matched, for example lower case letters *a* through *z*, or numbers between *0* and *9*.

The expression *[^abc][xyz].b** matches both *wy.b* and *3x.ball* (amongst others), but does not match *cz.bmg* or *rx.img*. The ^ character after the opening bracket indicates a negation, meaning any characters apart from those listed.

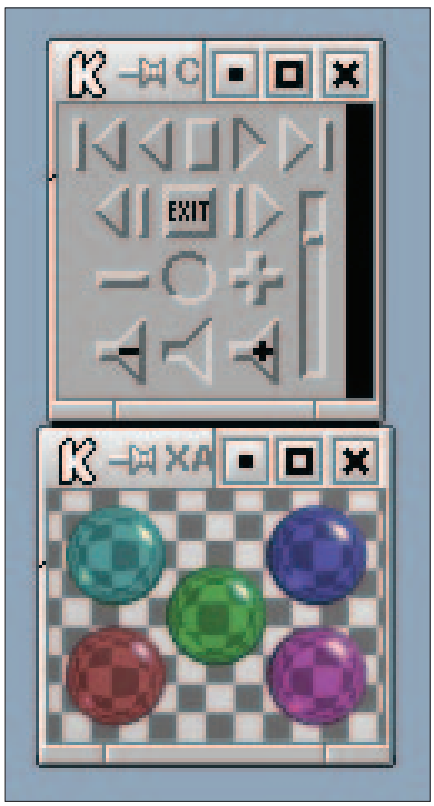

Figure 4: Animation