The POV-Ray raytracing utility

PERSISTENCE OF VISION

Should you have an evening, weekend, or a whole year to fill, then you might just find raytracing a worthy pursuit. Colin Murphy finds out why

POV-Ray (Persistence Of Vision Raytracer) is one of the many utilities that you may have on your system without even realising it was there. As such, you may have never got to play around with it, missing out on the chance to while away the hours with nothing more than idle tinkering, a *bona fide* pursuit if ever there was one.

Raytracing

For the uninitiated, POV-Ray is a raytracing tool. Raytracing is a method that enables you to create stunning graphic images, be they abstract, geometric or photo-realistic. This will take a little time to learn, as it calls upon your imagination and requires a little patience.

The first step in the road to creating your graphical masterpiece is to "describe" what you want to depict in your picture. This description comes in the form of a programming language, or uses an interactive modelling system, like a CAD package. Either way, you'll need to specify what objects are in your imaginary world, what shape they are, where they are, what colour and texture they have and where the light sources are to illuminate them. Having done all of this, you feed it into your ray tracer then sit back and wait, which is a necessary evil, as the rendering can take some time.

Without having some idea of how the images are built up in a raytracing description file, you won't be able to fully appreciate what is going on under the hood, so we will leave the CAD package-like development systems to one side and 'play' in text mode for the moment.

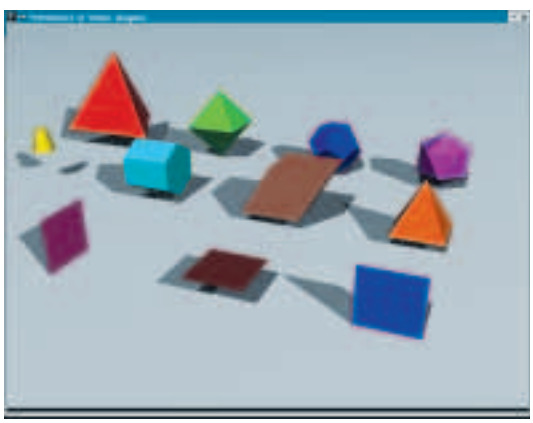

Figure 1: POV-Ray example image shape2. Well, we all have to start somewhere. On a 450MHz machine this took 15 seconds to render

POV-Ray is freely available for download. If you want the complete Linux distribution of POV-Ray 3.1g, including X Window and SVGAlib ELF binaries, documentation, sample scenes, and include files, download the povlinux.tgz file, which is 1.5Mb.

```
cd /usr/local/
tar -xzvf /download directory/povlinux.tgz
```

read the README.linux file and if all is well

```
povray31/install
```

will call up the installation script. Using this method, I found that a 'povray' executable wasn't created, but I did get x-povray and s-povray – you might like to set up a symbolic link to whichever you will use on your system. I will just refer to the executable I use.

There are two documents you should read to make most use out of POV-Ray: the README mentioned above and povuser.txt, which will also be in /usr/ local/ povray31/ if that's where you uncompressed the files to. This second text file contains a beginners' tutorial, a complete reference to the Scene Description Language, which you will use to code your efforts, and other information. This file is also available from the POV-Ray site in other formats, including HTML, which might make it more useful to some.

Create those images

To get you started, we'll run through some basic examples. In the /scenes/ directory you'll find some .pov files, which we'll use as our examples to take a look at. The command

```
x-povray +i
/usr/local/lib/povray31/scenes/incdemo/shapes2
+o /home/LinuxMag/shapes.png+W800 +H600 +D0
```

will produce the output shown in Figure 1. The +i and +o switches define the input and output files;

Figure 2: A more complex picture in only 500 more bytes, which only took another minute to render

the +W and +H define the resolution of the output file. The +D0 switch is a must for beginners, as it allows for some instant gratification that POV-Ray is doing something by printing a version of what it has rendered so far in an X window. This no doubt takes up valuable processor cycles, but while you're playing with POV-Ray it can only help.

Some code

This has come from a file less than a page long and we will run through just some of example lines in that code.

The camera section defines how the objects in the file are viewed:

```
camera {
location <10, 10, -20>
look_at <2, 0, 0>
}
```

Briefly, location <10, 10, -20> places the camera up ten units, ten units to the right and back twenty units from the centre of the raytracing universe (which is at <0,0,0>). By default +z is into the screen and -z is back out of the screen.

Also, look_at <0,0,1.5> rotates the camera to point at those coordinates. The look_at point should be the centre of attention of our image.

```
light_source {<0, 1000, -1000> colour
LightGray}
```

The vector in the light_source statement specifies the location of the light set to some extreme values to make the shadows obvious. The light source is a tiny invisible point that emits light. It has no physical shape, so no texture is needed. There are two light sources referred to in the code listing.

The remaining lines describe the construction of the objects in the image using Constructive Solid Geometry. POV-Ray allows us to construct complex solids by combining primitive shapes in a number of different ways. Full details of how this works can be found in the povuser documentation.

Figure 3: A much more complex example of rendering, which includes reflections

Experimentation really is the order of the day. You should tinker with the parameters in your favourite text editor and render a new image from your code. Code can be built up very quickly – the image shown in Figure 2 is only another 500bytes longer than the original example, which is almost an effort worthy of printing and mounting.

Graphical user interfaces.

Working from the command line might be a little off-putting for some people, particularly for those that like a friendly environment. There are plenty of GUI interfaces for POV-Ray available, and you should be able to find one that works on your desktop.

Povfront is one such front-end. It's designed to run under any flavour of Unix using GTK and glib libraries, it's POSIX compliant and has been successfully tested on Linux Red Hat 5.2/6.0/6.1 and Mandrake 6.0/6.1. It aims to provide an easy way to launch pov rendering with a graphical interface, which provides all the available options – even the script only ones. The only requirement is that you have the GTK+ (1.2 version) or later libraries installed.

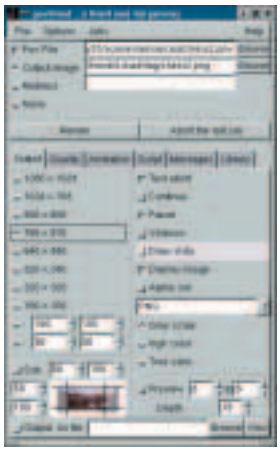

Figure 4: Povfront will allow you easy access to all the switches you can use in PovRay, and then some

Support

There is a very active user base for POV-Ray and raytracing in general. There are plenty of sites on the Web offering tutorials and other documentation, as well as people's own efforts in creating objects and textures for you to use in your work – there are even competitions to enter. Take a look at Figure 5 to see what you can achieve with some open source software and three day to render in!

Info

POV-Ray homepage	*http://www.povray.org/*
Povfront	*http://perso.club-internet.fr/clovis1/*