elow is the original README file from the RTrace distribution. Note that some of the information is not directly applicable to the Macintosh version, since I have made significant changes to the way RTrace interfaces with the user. In particular, I used the Mac Toolbox, rather than SUIT, and MacRTrace saves generated images as PICTs, and animations as Quicktime movies.

Incidentally, in this day of Shareware Madness, it is rare to see such a well-designed program as Antonio's RTrace available from free. Let’s give Antonio a big round of applause for writing this fabulous ray tracer, and for donating it to the world.

```
##############################################


        `rtrace' RAYTRACER


     written by Antonio Costa


         a_costa@inescn.pt


       a_costa@inescn.rccn.pt


         acc@asterix.inescn.pt
##############################################

This represents the 8th formal release of the `rtrace' Raytracer.
It was written to help me understand how raytracing works, to
generate cute images, and generally because I like to program.
Feel free to use it for any purpose, I am releasing it into the
public domain.

NEW: this version can be built with a user-interface from a
toolkit called SUIT from uvacs@cs.virgina.edu. The SUIT user-
interface toolkit exists for UNIX, DOS, MAC, Windows and a DOS
extender called GO32, so theoretically this program could run
unchanged in all those OS...
```

The input format to this ray tracer is called "SFF" or Simple File Format, after using "NFF" or Neutral File Format, which was invented by Eric Haines' for his Standard Procedural Database. The SPD was designed to allow programmers to test their raytracers on databases of varying sizes. While not the end-all to object file formats, it has served me well.

If anyone uses or wants to use NFF, I can send a NFF to SFF converter.

SFF supports the following concepts and primitives:

point lights

directional lights

spot lights with fall-off

extended lights

2 ways of defining surfaces

spheres

axis-aligned parallelipipeds

cylinders

cones

bicubic patches

polygons

polygonal patches (normals are interpolated from corner points)

3D text with high quality

CSG operations

4x4 matrix transformations

textures

depth of field

diffuse distribution

stereoscopic pair creation

The `rtrace' raytracer supports all of these primitives, with the minor
limitation that polygonal patches must be triangles (have only three
vertices).

Procedural textures (with 4x4 matrix transformations) are also supported:

checkerboard

color blotches

marble

bump map

fbm

fbm bump map

wood

gloss

image mapping

waves

(and many others...)

The output from the raytracer is very simple, and not directly tied to any specific device. It consists of a single line, with format in C style of "%c%c%c%c", which gives the resolution of the image (Width LSByte, Width MSByte, Height LSByte, Height MSByte). It is then followed by Width*Height sets of (red green blue) bytes.

I have lots of filters source code for displaying the ".pic" files, as well as interesting objects that I run accross. Filters already exist to display images on Suns, to convert to PostScript, as well as X11 bitmaps for xwud.

I advise you to get a package called the "Fuzzy Bitmap Package" (FBM), that has lots of useful programs for simple image processing, conversion, etc. The author is Michael Mauldin <mlm@nl.cs.cmu.edu>. The Utah Raster Toolkit is also a very good graphics package. Also good is Eric Haines' SPD source code, so you can generate your own fractal spheres, mountains, gears, etc.

Also thanks to the numerous authors whose research into raytracing has seen implementation in this raytracer.

Antonio Costa.