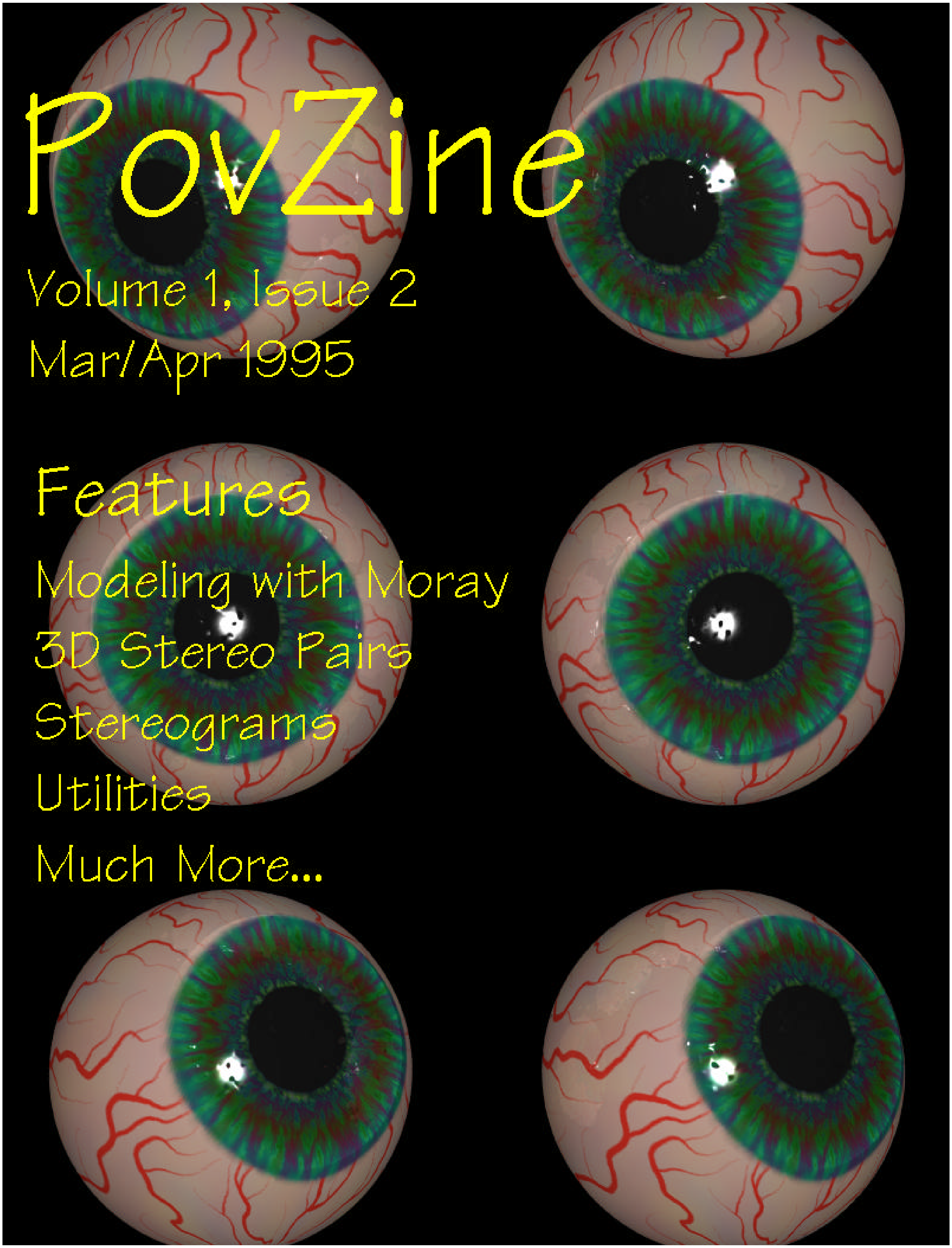


PovZine

Volume 1, Issue 2
Mar/Apr 1995

Features

Modeling with Moray
3D Stereo Pairs
Stereograms
Utilities
Much More...

Editorial

Welcome to the second issue of PovZine.

The first issue has been received well by the PovRay community. I've received over a hundred email messages the last couple of months about PovZine. I've had lots of suggestions and encouragement. In general, folks seems to have really liked the content and the number of images included in the first issue.

The Mar/Apr 95 issue is truly a collaboration of the PovRay community. The cover is by Steve Perrigo. There are articles by Steve, Perrigo, Harry Rowe, Robert A. Mickelsen, and yours truly. It is my hope that this level of contribution will continue in upcoming issues.

However, there is one topic that I've gotten so much email about that I feel I need to address here. That is the distribution format for PovZine.

I've had folks ask why I don't have a Amiga version, a Mac version, HTML version, a MS-Windows help version, a PovHelp version, a PostScript version, a text version, and an Adobe Acrobat version. I've had folks ask for a W4W version, a WordPerfect version, and several other word processor or DTP program for various system. It seems that everyone wants PovZine in their own unique format.

I can hardly blame you for wanting PovZine in a form that is convenient for you. It's a reasonable request that I wish I could accommodate. However, I'm just one person with limited resources. I'm doing this as a service to the PovRay community. I'm not making any money on this, so can't afford to invest much in tools. I selected Common Ground because it's cheap (~$70 for the distiller) and it allows me to target IBM PC, Mac, and Unix systems. It seemed like a good compromise between accessablity across platforms and my development effort.

As PovZine gets more established, I would like to add support for additional formats. But for the moment, I would like to focus on getting PovZine established and developing a high quality for both the content and form of PovZine.

I believe you will find this issue considerably better than the first issue. I'm extremely happy to have had several great articles contributed to this issue. I would also like to remind you that PovZine needs your contribution of articles, tips, and suggestions. Please help me to make this a resource that you look forward to receiving every other month.

Keith Rule

keithr@hevanet.com
■

About the Cover

The cover is by Steve Perrigo. A description of how the image was created is found in his article Creating A Cross-Eyed Stereogram which starts on page 12. Steve can be contacted at sperrigo@nwlink.com or at 70244,2773 on CompuServe.

PovZine

March/April 1995
Volume 1, Issue 2

Mission Statement

PovZine exists to serve the POVRay user community by providing useful POVRay related information in a way that is friendly and [hopefully] entertaining.

This magazine is intended to be freely available to all POVRay users.

Information

PovZine is published 6 times a year by Keith Rule. It is available on the anonymous ftp site "ftp.povray.org" under the directory pub/povray/ezine.

Submissions

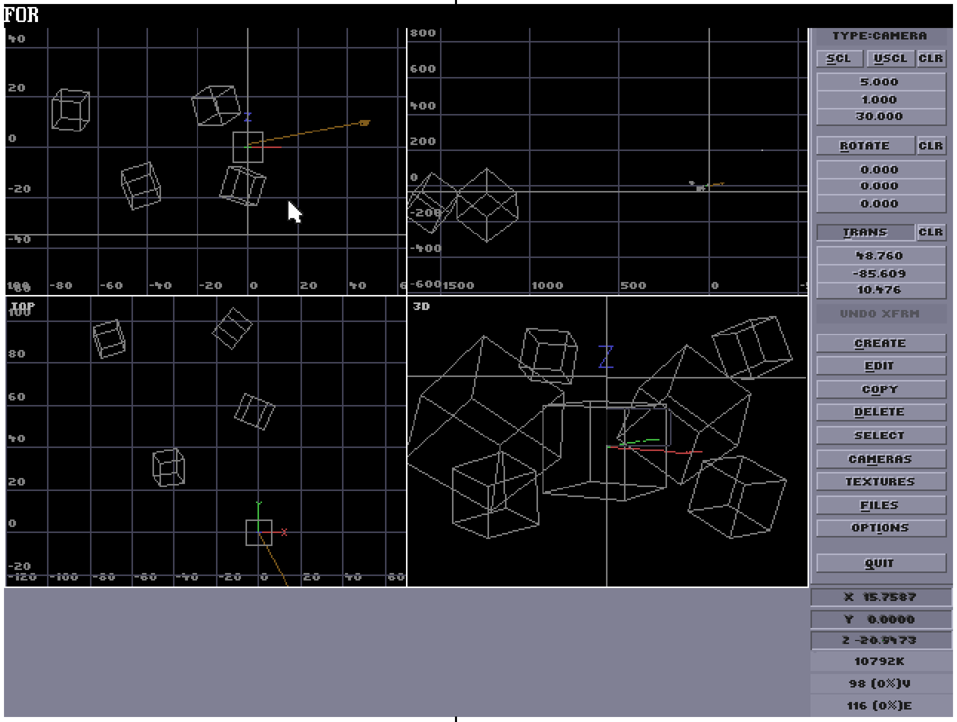
PovZine is looking for unsolicited articles, art, questions, etc. Please email information about your article, art, or your questions, to keithr@hevanet.com. ■

Features

Modeling with MORAY - Tutorial 1
Using 'Dummy' Objects

By Robert A. Mickelsen

When the first raytracers were written, they were nearly all text-based. Image makers had to visualize the scene elements and, through a series of trial and error test traces, fine tune the objects, textures, and lighting. This is fine if you are fluent in the particular language the raytracer uses, but it is extremely intimidating to newcomers faced with the task of learning a language that seems nearly as daunting as C.



Then along came the first modelers. The earliest ones were buggy and lacked many of the features that make programs like POV-Ray and Polyray so powerful. As time passed, subsequent releases cleared up the bugs and added the features. Today, there are several excellent modelers that make scene design, complex object creation, and even texture design much easier by introducing a visual interface. My favorite modeler (and the one about which I will be writing about) is SoftTronic's MORAY written by Lutz Kretzschmar.

Inexpensive shareware, MORAY is still an extremely powerful modeler. It has several features that set it apart from other would-be competitors. You would think that drag-and-drop transformations would be standard in any modeler, but it is a feature that is so far unique to MORAY among shareware and freeware modelers. Scaling, rotating, and translating can all be accomplished either by simply dragging with the mouse, or by numerical entry. Changes to simple objects are represented on screen in real time, while changes to more complex objects are represented by a box in real time. Screen redraws still occur in all active screens but, if redraw time is getting long due to complexity, screens can be turned off to hasten redraws.

The most powerful feature of MORAY is the object browse tree. This feature gives you a graphical representation of the hierarchy of objects present in your scene. It clearly shows what the edit screen cannot, the parent and child relationship between objects that make up groups and CSG objects. Where these object relationships are apparent in the text file, they are invisible in the wireframe view. The browse tree solves this problem in a very efficient manner. Not only can you view your objects, but you can select them for further transformations or other editing simply by clicking on them.

In this installment, I want to talk about using MORAY to make a scene with an object created in another program. Since there is no POV2MDL converter (actually, there is, but it is still in beta and lacks many features), you cannot 'import' an object into MORAY. But that should not prevent you from using the modeler to place the objects in the scene, and to do the other modeling involved. What we will do is to create several 'dummy' objects to take the place of the object we want to import.

Let's make a scene using an object created in Andrew Rowbottom's program called FORM. The data for this object is available on Compuserve in the GRAPHDEV forum in the POV Modelers Library. The filename is ammont.zip. Download this file and unzip it before continuing this tutorial. The unzipped file should be 'ammonit.pov'.

ammonite.pov

I suggest rendering this file before continuing to see what we will be placing in the final scene file. You will see that the object is an ammonite-type thingie with a nice stone10 texture. We like the way this one looks and want it to look just like this in the scene file, but we also want four companion ammonites in other positions and rotations to keep this one company. If we were to try and accomplish this using text only, it would mean numerous trial and error renderings just to get the positions of the ammonites correct relative to the camera as well as the positions of other objects in the scene. MORAY makes this a much easier task.

1. Start MORAY and click on 'files', 'new', and 'delete scene from memory?', 'yes'.
2. Create a camera, select it (shift-drag a rectangle over any portion of the camera), and enter the following coordinates. (These coordinates are taken from the ammonit.pov file.)

   ```
   location: <40, -80, 20>
   // swap z and y coordinates
   // right handed system
   ```

   Accept the defaults for everything else for now.
3. Create a point light source and enter the following coordinates, also taken from ammonit.pov:

   ```
   location : <100, -300, 200>
   // z and y are swapped accept
   // the defaults for everything
   // else.
   ```

4. From the file menu, click on the name bar and name your new file scene01.mdl. Click on 'save', and then 'done'.
5. Create a plane and name it 'surface1'. Click on 'new' in the texture list box, and then 'create'. Select 'opaque' and name it 'surfacetex1'. Click on 'done' and then play with the color sliders until you get a kind of sandy beige color... like desert sand. (I used rgb <.7647, .6549, .5451>.) Set phong and phongsize value to '0' and click on done. Back in the texture list

box, click on 'surfacetex1', 'done', and again 'done' to return to the main edit screen. Click on USCL and enter '1000' in any of the boxes to scale the plane up *1000. (We do this because the wireframe representation of a plane is too small in relation to the other objects we plan to put in the scene. We will delete the scale factor from the plane later.)

6. Create a cube and name it 'dummy1'. Click on 'new' in the texture list box and, using the same procedure as before, make an opaque texture named 'dummytex' and accept the defaults. Click on 'done' and then on 'dummytex' in the texture list box before returning to the main screen. (The only reason to select a texture is because MORAY will not export to POV unless every object has a texture.) Universal Scale (USCL) the cube *6. This cube will be our dummy object and will represent the ammonite we will be substituting later. Except for the scaling, we want to leave this cube untransformed because we want the first ammonite that we will place in the scene to look like the ammonite in ammonit.pov.

7. With the first dummy cube still selected, copy it. Don't bother with the transformations in the copy object dialogue box. If we could figure out this stuff in our heads we wouldn't need a modeler would we? Click on 'OK'. From the main screen, click on 'select'. From the browse tree, click on 'dummy2', and right mouse-click to get back to the main screen. Zoom out (alt-drag toward you) in the TOP view so you can see the objects and camera. Make sure the translate button is pushed, and drag the cube to the upper left. I prefer to watch the 3D screen while I do this, even though I am working in the TOP view. Then, using the same method in the other views, adjust the position until the cube is right where you want it to be. Fiddle around with the rotation to get a slightly different position than the original. These are the transformations I used:

```
   rotate <-340, -4.519, -16.562>
   translate <-70.842, 91.620, 7.122>
```

8. Reselect the original cube and, using the same method as above, make three more copies and scatter them around the screen as you want, varying their rotations and distance from the camera. These are the positions and rotations I used:

```
dummy3: rotate <-18.434, 0, -67.19>
   translate <-1.954, 57.029, -15.476>
dummy4: rotate <337.329, 0, 40.643>
   translate <-12.623, 96.294, 8.216>
dummy5: rotate <19.778, -25.938, -89.972>
   translate <-42,759, 30.655, -15.122>
```

9. Now we notice that the plane is intersecting a couple of the cubes. This is no good, so select the plane and translate it -35 in the z axis. This should place it below all of the cubes.

10. We have now placed all of our dummy objects. Just for fun, let's make a couple of pyramids to go in our desert scene. Create a new cube, name it 'Pyramid1', assign it 'surfacetex1' as a texture, and scale it up (USCL) *100. Then, rotate it so that one corner sticks upward above the plane from the corners to the point. It should look like a pyramid above the plane. Then, drag your pyramid +y until it is an appropriate distance away. I used the following values:

```
rotate <-317.549, 36.264, -98.848>
translate <-666.859, 1551.332, -98.271>
```

Copy this cube and apply the following transformations:

```
scale (USCL) *120
rotate <-315.762, 36.264, -0.081>
translate <-1017.13, 1241.683, -
113.427>
```

11. At this point, we decide we are not happy with the camera position so we select it and drag it down and away a little bit. The resulting values are: <49, -86, 10.5>. Now dummy2 and dummy4 are a little too low so select them and change their translation z value as follows: dummy1 z=14.881; dummy2 z=16.666. Click on 'file', 'save', 'export to POV', and then render your file. You should have a beige plane, two pyramids, and five cubes in various positions. Boring, eh? Not for long! <G>

12. Copy ammonit.pov into your scene directory if it is not already there. You will have to edit the file to comment out the camera, light, and the standard includes at the beginning. Go to the end of the file and add the line: '#declare Ammonite=' immediately before the final FormObject block. In the FormObject block, reduce scale from 100 to 50. Immediately after that line, add the following transformations:

    ```
    rotate <90, 0, 0>
    scale <1, -1, 1>
    ```

    These transformations effectively 'convert' the ammonite object from left-handed coordinates to right-handed coordinates. (See POVFAQ.DOC for details)

13. Now, make the following changes to scene01.pov:

    a. Add '#include "stones.inc" ' to the standard includes at the beginning of the file.
    b. Add '#include "ammonit.pov" ' as the first line under the OBJECTS header.
    c. Delete the first six lines of each 'dummy' block and replace it with the Ammonite like this:

Before	After
`box{ //dummy*` `<-1, -1, -1>, <1, 1, 1>` `texture {` `dummytex` `}` `scale <6.000, 6.000, 6.000>` `rotate <x, y, z>` `translate <x, y, z>` `}`	`object {Ammonite` `rotate <x, y, z>` `translate <x, y, z>` `}`

Now try and render this scene again. You will notice that the ammonites have replaced the cubes, but have retained the transformations you modeled in MORAY using the cubes. This is much more interesting, isn't it? (If you haven't figured it out by now, we are not going back to MORAY. Everything else we need to do is most easily done by editing the .inc and .pov files by hand. Modelers are good for modeling, but are not well suited to the fine-tuning stage of scene development.)

But the scene still needs something. The plane is too smooth and featureless so let's add some ripples for effect. In the scene01.inc file, add the following normal statement to the 'surfacetex1' texture block:

```
normal {
      ripples 0.35
      frequency 100
      turbulence .25
      scale 1000
      // this is needed only if you
      // remove the scale *1000 of
      // the plane
} // created in MORAY
```

The Pyramids could use some surface texture as well. They look too flat. But we gave the pyramids the same texture as the plane, 'surfacetex1'! No problem. Just use your text editor to copy surfacetex1 and rename it 'pyramidtex'. Replace the ripples with the following bumps normal: normal {bumps 1 scale .01}. Now, return to scene01.pov and change the texture name in the pyramid blocks to 'pyramidtex.'

```
   normal { bumps 1 scale .01 }
```

We also need some sort of sky. We don't want to use any kind of standard 'cloud' texture. I think a simple gradient would be best, but how do we do that? We also want to create the illusion of distance for the pyramids and fog is best for that. Maybe we can combine the two effects to get what we want for both features. Add the following to the end of the scene01.pov file:

```
plane {z,150  //sky plane
   pigment {color Blue}
   finish {ambient 1 diffuse 0}
}

fog {
   color rgb <.765, .655, .545>
   //the 'sand' color
   distance 1000
}
```

There is one last change to be made. With the sky plane set at z=150, the light source must be lowered from 200 to 145.

Now render your scene. Ahhhh... that's better! It is not so boring anymore is it?

That's it for today, campers! I hope you had fun and maybe learned something about what modelers can and cannot do. In the next issue of PovZine, I will be talking about the dreaded bezier patch and how to do the impossible with it!

Until then... keep tracing!

■

Did You Know?

Did you know that you can use files from version 1.0 of PovRay directly in PovRay V2.x without modification?

All you need to do is use the +MV1.0 flag on the command line when invoking PovRay.
■

Classifieds

Classifieds are available to anyone who would like to advertise a PovRay related product or service. Please send text of your classified to keithr@hevanet.com.

Raytech BBS is the UK's specialist independent raytracing support service and offers free public access. Set your comms to 8N1 and your terminal emulation to ANSI and dial +44 1862 83 2020 and see why we have 4,500 registered users making 1,000 calls each week. Email facilities for all users. PCGnet. Dgnet. 24net. Internet News. Moray and Vivid registrations. 3D Artist magazine subscriptions. Waite Group Books held in stock.

Fargo Primera Dye Sublimation Printer for IBM PC. Used for about one year. In great shape. Includes dye sublimation drivers, standard 3 color ribbon and some paper. Price $500.00 plus shipping. Contact: keithr@hevanet.com.

Micro System Options: 3d Graphic Tools 3.1.1 Library and source - Professional C/C++ developers edition for MS-Windows. It's fast and includes source code.
Paid $300, will sell for $150. Contact: keithr@hevanet.com.

Windows World BBS
(513) 866 - 8181   V.34
Dayton, OH, USA
Profiled in the January 1995 issue of
Boardwatch Magazine
Specializing in DOS and Windows based ray tracing files and 1 gig of Windows CICA programs

Your host, Harry L. Rowe, cordially invites you to call. Free access of 40 minutes/day and 3 meg downloading on your first call. People who contribute files get greater access. The BBS requires an Excalibur ™ Windows client program which can be freely downloaded.

email:
harry.rowe@wedowind.lexis-nexis.com or
ac401@dayton.wright.edu

3D Stereo Pairs with PovRay

By Keith Rule

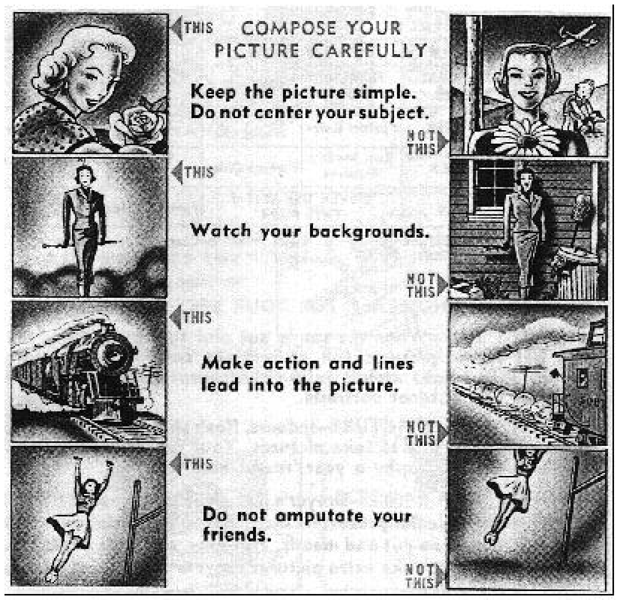
I discovered PovRay and 3D cameras both about 2 years ago. I was surprised at how many 3D cameras existed along with the accessories required to mount and view stereo images. I was even more surprised to discover how well these two hobbies go together.

After gaining experience with both stereo cameras and PovRay, I've discovered that there were three key areas that I needed to learn about to create raytraced 3D images. These areas were:

- Composition
- 3D Camera Techniques
- Mounting & Viewing

Composition

There are only a few fixed rules in 3D image composition. Here are some rules from a 1950's era ViewMaster personal camera users


THIS
COMPOSE YOUR PICTURE CAREFULLY
Keep the picture simple. Do not center your subject.
NOT THIS
THIS
Watch your backgrounds.
NOT THIS
THIS
Make action and lines lead into the picture.
NOT THIS
THIS
Do not amputate your friends.
NOT THIS

manual. Following these rules is generally good advice even for raytraced images. However, there are no fixed rules where composition and art are concerned.

3D Camera Techniques

Precise camera alignment is key to a 3D stereo image that is comfortable to view. The following items are key to making an easy viewing 3D image.

- Vertical Alignment - The same points in the two stereo images must be on the same horizontal line. If there is a perceptible difference there will be some very unpleasant eye strain for the viewer. For example, if you have a visible horizon they should line up on both images.

- Camera Separation - This is key to getting a natural feeling of "depth" into an image. There is no fixed distance between human eyes, but you can assume that it is around 2". If you're are creating a 3D image to scale, then your camera separation should be close to that of the human eye. If you have a larger separation than that you get a hyperstereo effect.

  Hyperstereo in 3D stereo photographs can be used to give addition depth to landscapes (such as the Grand Canyon). ViewMaster slides of landscapes are typically a little bit hyperstereo. However, if there are other objects that are familiar in the scene, such as a house or a car, they may look like a doll house or toy car. This effect of hyperstereo needs to be considered in raytraced images too. Try to keep the camera separation close to the scale you'd expect in the real world.

For raytracing applications where you don't have any specific scale, using a 30:1 ratio of your main subject to camera separation usually give very satisfactory results.

Let's use these rules to modify the camera in the PovRay scene file `basicvue.pov`. We will assume that the original view is the left image of the stereo pair. This scene's subject is a sphere located at <0, 3, 0> with a radius of 3. The camera definition for this scene is:

```
camera {
   location  <0, 3,-10>
   direction <0, 0,  1>
   up        <0, 1,  0>
   right   <4/3, 0,  0>
   look_at   <0, 2,  0>
}
```

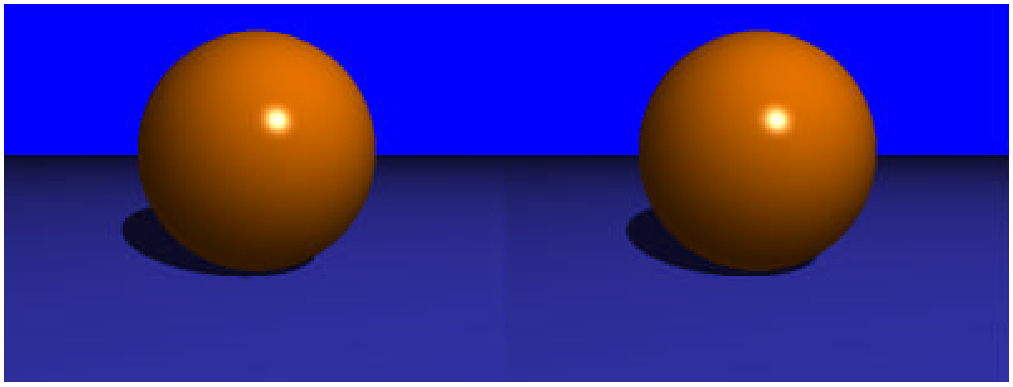
Notice that Y is the vertical axis, X is the horizontal axis, and Z gives the depth. To create the right image we need to move the camera to the right using the 30:1 ratio. The nearest point of the sphere is -3 on the Z axis. This means that its distance is 7 units away from our camera. This 30:1 ratio means that we need to move our camera to the right 1 unit for every 30 units of distance the nearest edge of the object is to our camera. So in this case the formula `old_x + (distance/30)` will calculate the new position for the camera along the x axis. So when we substitute this formula into the x position of camera location statement we get the following:

```
location < 0+7/30, 3, -10>
```

or

```
location <0.23, 3, -10>.
```

The following image is the result of combining these two images into one stereo image.


basicvue.pov in stereo

The two PovRay files were rendered and then pasted together using a paint program (Fauve Matisse).

It's easy to see that creating 3D stereo images is very straight forward. The technique used for creating a stereo pair for a simple image or a complicated image is exactly the same.

Mounting & Viewing 3D Images

Viewing an image is a bit more challenging. There are several possibilities for viewing images. I will go through three of the many possibilities. These are:

- Free-viewing - which similar to viewing a stereogram.
- Mounting & Viewing Slides - This uses standard, readily available Realist viewers and slide mounts.
- Mounting & Viewing Printed Images - This uses viewers such as the one available with the Loreo camera to view stereo pairs.

Free-viewing

Free viewing has two advantages over the other viewing methods.

1. It doesn't require any special equipment.
2. It's always available.

The downside is that some people have trouble learning to free-view. Many who can free view initially complain of eye-fatigue and headaches. My experience is that it is easy to learn, and I've had little or no problems with eye-fatigue and headaches. Of course, your mileage may vary.

I free-view by looking at the stereopair. I then relax my eyes so that they go out of focus. I slowly refocus while trying to keep 3 images visible. The middle image is the one which will pop out into a 3D image. This may take some practice. I remember sitting and trying this for 10-20 minutes before I could do it the first time. So relax and give it a few tries. This technique works well on a computer screen, or printed page.

Mounting & Viewing Slides

Mounting slides requires considerably more work than just free viewing an image on the screen. You will need to photograph your images from your computer screen while maintaining careful vertical alignment. Alternately, you could take your images to a service bureau and have them transferred to slide. The same mounting techniques apply regardless of how you get the slides.

The most popular format for 3D slides is called the Realist format. This is the format that the 1940-1950's Realist camera used. Slide mounts and viewers for this format are available from Reel 3-D Enterprises, Inc. [(213) 837-2368].

.

A slide cutter, Realist Viewer, and. half-frame slide mounts.

This simplest way to create a Realist compatible slide is to place the two images side-by-side on your computer screen and photograph the screen in a darkened room with a 35mm camera on a tripod. You want to make sure your shutter speed equal to or slower than the time it takes to update your monitor screen. Typically a 30th of a second or slower is adequate. Make sure your stereoimages are square and fill your viewfinder.

Carefully remove the slide from the mount. Cut the slide in half. Mount these slide halves in a half-frame cardboard slip-in mounts (Reel 3-D stock No. 4716).

Realist slide viewers are still available, consult Reel 3-D for current available models.

Mounting & Viewing Printed Images

I own a Loreo stereo camera. This camera takes two half frame images on a single 35mm frame. When these images are printed, both images are placed on a single print. This single photograph is placed into the Loreo viewer for viewing.

Using the half-frame technique defined for the Realist slides above also works for prints. Simply use negative film rather than slide file. Print the image normally, and then place the image in your Loreo (or compatible) viewer. This works quite nicely. ■

Creating A Cross-Eyed Stereogram

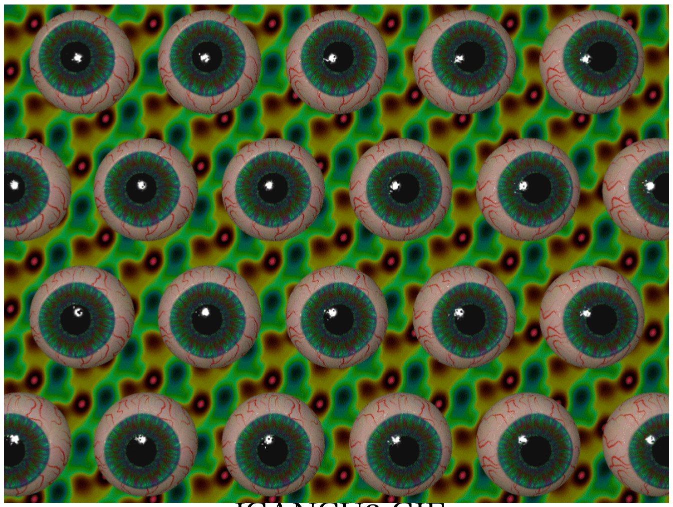
By Steve Perrigo

Cross-eyed stereograms differ from the "mall stereo images" in that there isn't any hidden image in them (they're kind of WYSIWYG), the textures take on additional sharpness when viewed in 3D, and the objects can have a real feel of depth when properly viewed. They're actually quite easy to view. All you need to do is cross your eyes. If you think you can't do that - think again. You're crossing your eyes now if you're reading this text with two eyes. The secret of cross-eyed viewing is to converge your eyes in front of the focal plane while focusing behind the convergence point. This allows adjacent images to overlap, giving the illusion of stereoscopic (3D) sight.

Creating cross-eyed stereograms in POV-Ray is actually quite easy. A cross-eyed stereogram can consist of a pair of objects or a whole series of objects lined up (as in ICANCU2.GIF). Creating these objects with POV-Ray doesn't even require placing objects at meticulously calculated angles either. All of the work is in the placement of the camera, light and the objects. Start with a simple scene:

1. Define an object, such a sphere or a box of two units in size located at the origin and apply a texture to it.
2. Make a copies of the object of it at 2.5 units in the +X and -X directions.
3. Place a camera about 15 units away along the -Z axis, pointing at the origin.
4. Place a light source AT THE CAMERA LOCATION.
5. Render away, and voila - your first cross-eyed 3D image

```
#include "colors.inc"
```


ICANCU2.GIF

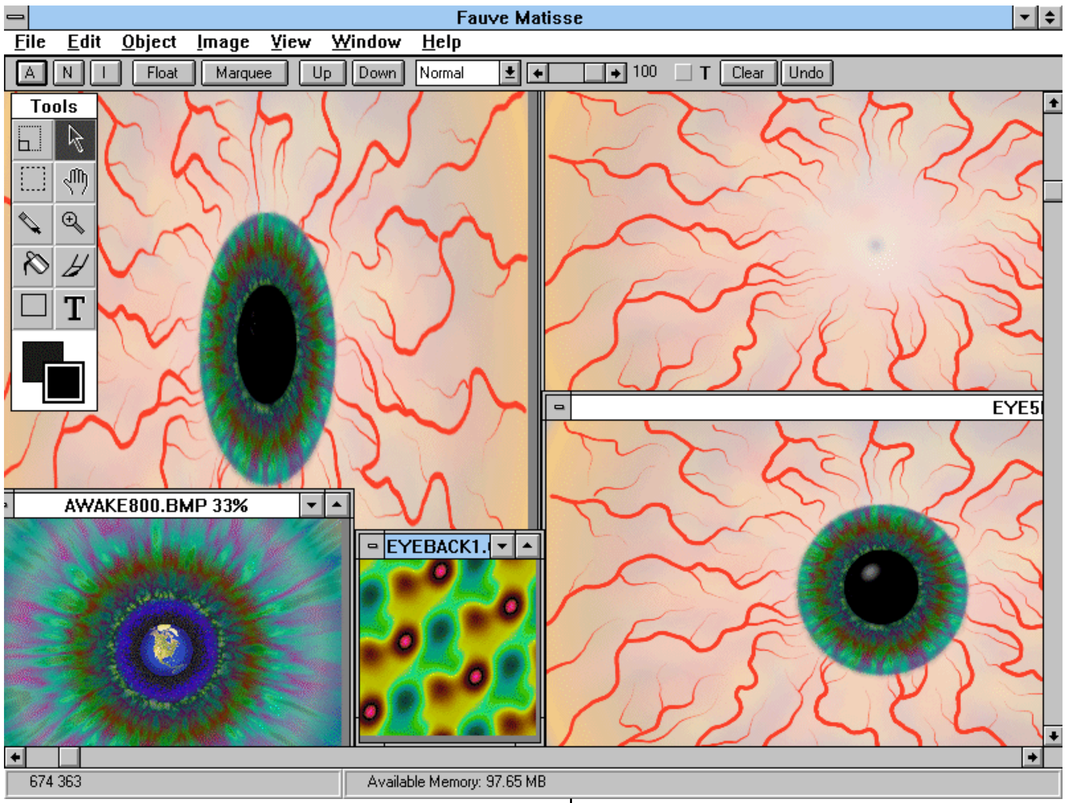
```
#include "textures.inc"
#declare RedBall = {
      sphere <0,0,0>,2
      texture <Red_Marble>
}

camera {
      location <0,0,-15>
      look_at <0,0,0>
}

light_source { <0,0,-15>
      color White}

object { RedBall
      translate <2.5,0,0> }
object { RedBall
      translate <-2.5,0,0> }
```

The scene in ICANCU2.GIF was created by placing rows of eyeballs with a diameter of about 0.8 unit at a distance normal to and about 20 units away from the camera. The tiled plane in the background is just behind the eyes. The "apparent" location of the tiled plane can be placed at any distance in front or behind the rows of eyes by varying the distance between the repetition of the seamless tiles. The final, and most important secret for creating these images is the placement of the lights. Shadows give visual cues regarding the depth and placement of objects. Shadows in this type of cross-eyed stereogram give conflicting information to your eye. To eliminate shadows, just place the single light source at the precise position of the camera.. Cross-


Fauve Matisse
File Edit Object Image View Window Help
A N I Float Marquee Up Down Normal 100 T Clear Undo
Tools
AWAKE800.BMP 33%
EYEBACK1.
EYE5
674 363
Available Memory: 97.65 MB

eyed stereograms can be created with multiple light sources and shadows - but that requires a different technique.

For a real strange effect - view this image using the divergent viewing technique common to RDS images. The 3D effect will be reversed, with the green plane in the foreground and the concave eye shells pointing away from you.

The IRIS and the Eyeball

Creating the iris of this eyeball was especially interesting. The iris is actually a raytraced image that was combined with hand-draw artwork and then used as an imagemap. The green iris part of the eyeball was created by rendering an image of looking down a long cone (on the inside -- looking toward the apex) with several light sources situated along the line of sight - in front of the camera. The cone was textured with a heavily modified Cork texture. The cone image was then post-processed with a special gradient transformation in Fauve Matisse using a plug-in from Kai's Power Tools.

The imagemap for the eyeball was created in Fauve Matisse. I started with a yellowish-white background and then hand-painted the blood vessels. The original imagemap has many more veins in it -- but they tended to disappear with antialiasing. Once I had a nice bloody eye <ggg>, I clipped the iris from my cone render, pasted it in, and smudged the edges to the white of the eye. Then, the center of the eye was filled in with a black circle. The final trick to creating the imagemap of the eye was to stretch it vertically, so that when applied to a sphere, the features of the eye would be round once again.

Finally, the creation of the eyeball was completed by placing the imagemap onto a sphere with one side truncated where the iris would lie. Then, the whole eyeball was “encased” in another larger sphere of glass with a slightly bumpy and shiny surface. ■

You can contact Steve at sperrigo@nwlink.com or at 70244,2773 on CompuServe.

Creating Random Dot Stereograms with PovRay

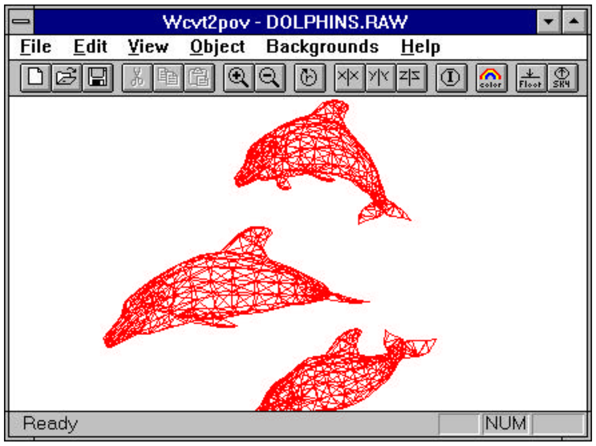
By Keith Rule

It's hard to walk through a mall or bookstore these days without seeing a book or poster of 3D stereograms. These images look very impressive, but are actually very easy to create.

There are virtually dozens of programs available for the IBM PC and other platforms that can create Stereograms. All of these programs have on thing in common, they need a depthmap to create a 3D image.

A depthmap is simply a grayscale image where the nearest points of the scene are lighter than the furthest points. Creating a depthmap of an image is very easy with PovRay.

For this article I will use Wcvt2pov to create a PovRay image, then replace the default textures in the PovRay file with one that will create a depthmap. Then I will use that depthmap image to create a random dot stereogram using a program distributed with Kai Power Tools called "3D Stereo Noise".

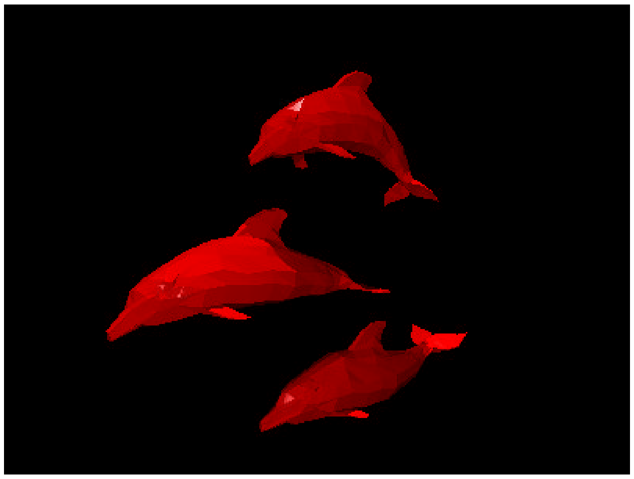


First select a 3D model with Wcvt2pov. I like dolphins, so I will import a dolphin model.

That image is saved as a PovRay file using the File | Save As | PovRay V2.2 (*.pov) menu selection. Name the file dolp.pov. Now render the file with PovRay using the command:

```
povray +H300 +W400 +V +Idolp.pov +Odolp.tga
```

Here is what the rendered scene looks like.



This scene looks good to me. Now we need to modify the PovRay file to allow us to create our depthmap image.

PovRay files are just ASCII text files that contain a description of the scene being rendered. The following text is from the dolp.pov file:

```
light_source {
        <0, 0, -967.938171> color White}
light_source {
        <0, 2419.845428, 0> color White}

// Object converted to POVRay V2.2 by
// WCVT2POV V2.5
#declare _Dolph01 = texture {pigment
        {color Red} finish {Shiny}}
#declare _dolph02 = texture {pigment
        {color Red} finish {Shiny}}
#declare _dolph03 = texture {pigment
        {color Red} finish {Shiny}}


// This object has the following minimum
// and maximum values:
// xmin=-318.371287, xmax=372.306823
// ymin=-373.515331, ymax=373.515331
// zmin=-268.065488, zmax=332.034750
```

The key thing to note with this PovRay file is that the Y axis is the vertical axis, the X axis is the horizontal axis, and the Z axis gives the the depth. To make our depthmap we must delete the lights and existing textures and replace them with a texture that moves from White to Black along the Z axis.

To begin, delete the light sources, and the textures `_Dolph01`, `_dolph02`, and `_dolph03`. Then remove all occurrences `texture{_Dolph01}`, `texture{_dolph02}`, and `texture{_dolph03}` in each of the objects [the occurance of each texture will be found at the end of each object definition].

Now add the following texture near the top of the file:

```
#declare DepthMapTexture = texture {
      pigment {
            gradient z
            color_map {
                  [0.0 color White]
                  [1.0 color Black]
            }
      }
      finish {ambient 1.0 diffuse 0.0}
}
```

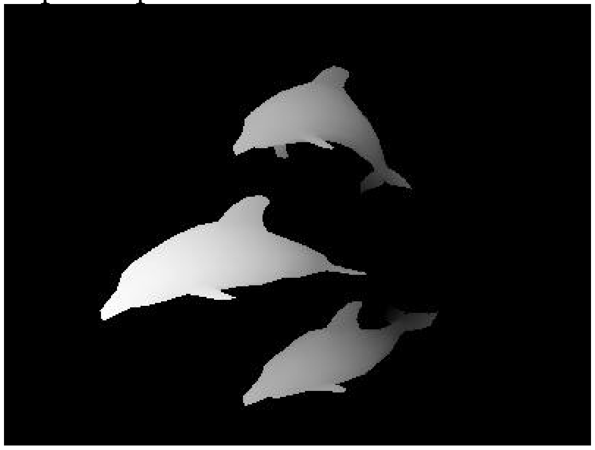
This texture simply varies the color from White to Black along the z axis. The color gradient by default starts at 0.0 on the z axis with the color White and transitions to black by 1.0 on the z axis.

Notice that the file contains the minimum and maximum values of each of the axis of the dolphin model. This will be very helpful to us now. Now go to the very end of the file. You will see two closing curly braces: Replace them with:

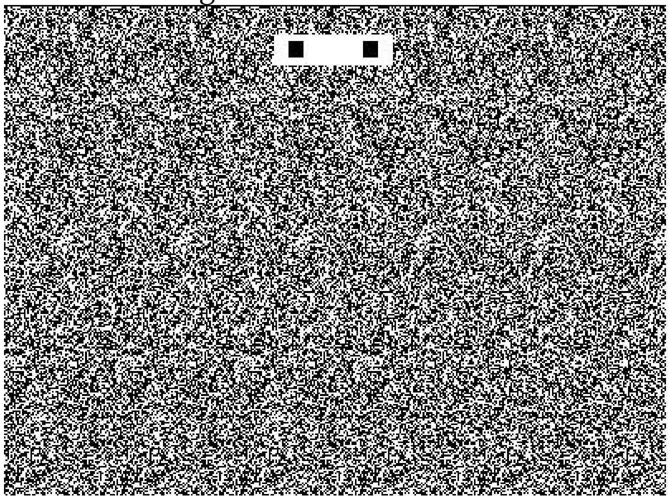
```
      }
      texture {
            DepthMapTexture
            scale <1, 1, 600>
            translate <0, 0, -268.065488>
      }
}
```

The scale value is simply the absolute value of zmin - zmax. This allows the grayscale image to transition from White to Black in the length of the object. The translate value is simply zmin, this moves the starting point of the White color to the beginning of the dolphin object.

When this file is rendered, the following image is the result. This file is our depthmap.



Now all that is required is to put the file through one of the many programs that creates random dot stereograms. For this example I will use the "3D Stereo Noise" filter that came with Kai Power Tools. The resultant image looks like this.



This short example should help you use nearly any stereogram generation software. The principles are the exactly the same.

■

Running POV-Ray under Windows 3.1

By Harry L. Rowe

Yes, I know! Running POV-Ray under Windows is an oxymoron. However, if you are like me, your new system came with Windows 3.1 or Windows For Workgroups 3.11 preinstalled. Perhaps that is the operating system that you use at work? To allow you to use your system while rendering a large scene that could take hours or even days, you can set it up to run in the background in a "DOS box." Does the scene render more slowly? Absolutely! I have seen a 50% decrease in speed on a 486DX33 versus running under straight DOS 6.20. Unless you have a deadline to meet, so what!

I would like to show you the method I use and it works great. You must know that under this method you cannot watch the scene graphically build as it renders. This only makes sense because the video cannot do two things at once.

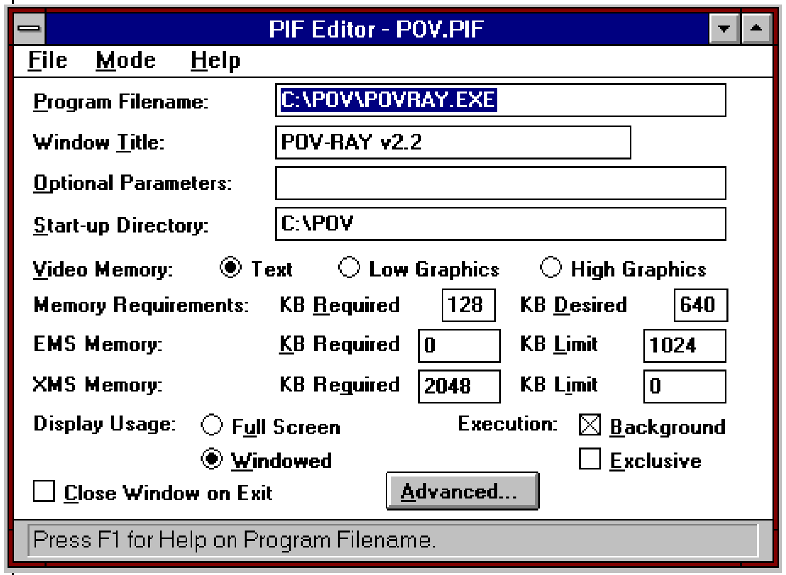
I do not use the OFFICIAL POV-Ray DOS compile. Instead, I use POVFAST which is a Watcom compile done by A. Haritsis (ah@doc.ic.ac.uk). Of course, you could use the official POV-Ray version or other "souped-up" compiles. Look for povfast.zip. My BBS has it along with ftp.povray.org and Compuserve GRAPHDEV. It uses the Windows swap file for virtual memory for those huge scenes we create or run across. Under many scenes I have measured a 23% speed increase.

1. Make sure povray.exe is installed in a directory of your choosing along with the necessary POV-Ray include files. As an example:

```
   c:\pov   and   c:\pov\include.
```

Also make sure dos4gw.exe is in the same directory with povray.exe. This is Relational's 32 bit DOS extender. It comes with POVFAST.

2. Now you need to make a Program Information File (PIF). On your desktop under the Main group you should find the PIF Editor. After you fire it up, enter the necessary information. Here is a copy of mine that you can use.



PIF Editor - POV.PIF

Program Filename: C:\POV\POVRAY.EXE

Window Title: POV-RAY v2.2

Optional Parameters:

Start-up Directory: C:\POV

Video Memory: [✓ Text] Low Graphics High Graphics

Memory Requirements: KB Required 128 KB Desired 640

EMS Memory: KB Required 0 KB Limit 1024

XMS Memory: KB Required 2048 KB Limit 0

Display Usage: Full Screen [✓ Windowed]

Execution: [✓ Background] Exclusive

Close Window on Exit

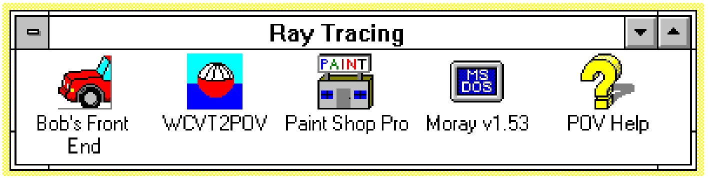
Press F1 for Help on Program Filename.

Notice that Video Memory: text is checked and I told Windows to give POV-Ray at least 2 megabytes of XMS (extended) memory. No need to enter a limit. Windows will only use what is available anyway. Background and Windowed are checked. Close Window on Exit is a personal choice. Do not worry about Advanced settings, the defaults are fine. You must save it as POV.PIF to your Windows directory.

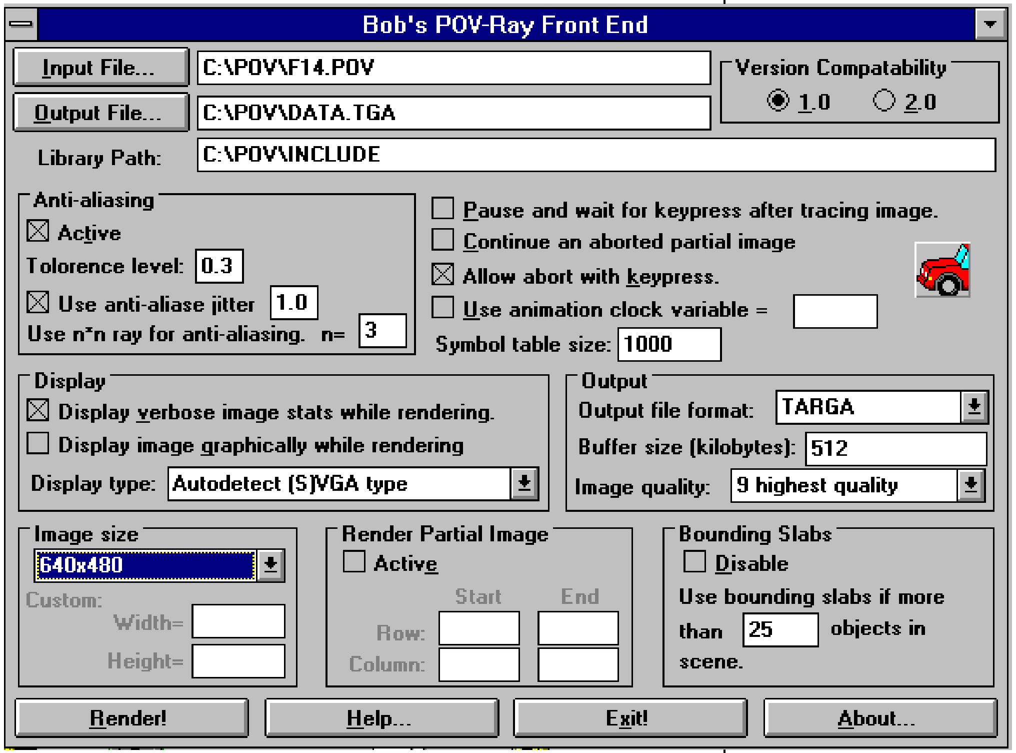
3. We could set up a batch file, which is useful for running several scenes, or we could have entered a ? mark in the Optional Parameters: field. This would result in a dialog box popping up to enter POV-Ray command line parameters when you run povray.exe. This is tedious. What I use is a Windows front end program to enter the switches. There are a few available but I use

Bob's POV-Ray Front End. You can find it on my BBS or Compuserve GRAPHDEV listed as bpfe10.zip. I do not know if it is at ftp.povray.org. Best of all, it is Freeware!

4. Unzip the archive and you can give it it's own directory. I simply dropped mine in the same directory with povray.exe. Next, place it in your appropriate program group by creating a new program item. For Command Line: I entered C:\POV\BPOVFE.EXE and for Working Directory: I entered C:\POV



Here is what the program looks like after you double click the icon:



Bob's POV-Ray Front End

Input File... C:\POV\F14.POV
Output File... C:\POV\DATA.TGA
Library Path: C:\POV\INCLUDE

Version Compatability: (● 1.0) ○ 2.0

Anti-aliasing
☒ Active
Tolorence level: 0.3
☒ Use anti-aliase jitter 1.0
Use n*n ray for anti-aliasing. n= 3

☐ Pause and wait for keypress after tracing image.
☐ Continue an aborted partial image
☒ Allow abort with keypress.
☐ Use animation clock variable =
Symbol table size: 1000

Display
☒ Display verbose image stats while rendering.
☐ Display image graphically while rendering
Display type: Autodetect (S)VGA type

Output
Output file format: TARGA
Buffer size (kilobytes): 512
Image quality: 9 highest quality

Image size: 640x480
Custom: Width= Height=

Render Partial Image
☐ Active
Start End
Row:
Column:

Bounding Slabs
☐ Disable
Use bounding slabs if more than 25 objects in scene.

Render! Help... Exit! About...

5. Enter the appropriate information and once POV-Ray starts you can immediately exit Bob's and minimize povray.exe to an icon. You can then get back to some serious work such as playing The Incredible Toon Machine or Myst :-).

Harry Rowe is the sysop of Windows World BBS (513) 866 - 8181. His email address is harry.rowe@wedowind.lexis-nexis.com (primary checked daily) OR ac401@dayton.wright.edu (FreeNet address checked every few days).
■

Polyray and POV-Ray Resource List For DOS and Windows™ Platforms: Part 1

By Harry Rowe

The Ray Tracers

Polyray

Alexander (XANDER) Enzmann's Ray Tracer. Shareware $35.00. DOS based only and source is not available. It has powerful math functions and great animation support. Harry's Note: I am a registered owner.

Polyray v1.7 Executable (coprocessor)
(ply386.zip 217,074 bytes)

Polyray v1.7 Executable (coprocessor)
plyexe.zip (181,794 bytes)
Polyray v1.7 executable (no coprocessor)

Polyray v1.7 Data files
plydat.zip (280,067 bytes)

Polyray v1.7 Documentation
plydoc.zip (74,483 bytes)

Polyray v1.7 Utility programs
plyutl.zip (78,518 bytes)

POV-Ray v2.2 Documentation
povdoc2.zip (200,565 bytes)

PovRay

Persistance of Vision Raytracer. PovRay is Copyrighted freeware.

POV-Ray v2.2 DOS Executable (Official Distribution)
povibm22.zip (252,830 bytes)

POV-Ray v2.0 Scene Files
povscn2.zip (499,737 bytes)

POV-Ray C Source Code for All Platforms
povscr22.zip (424,153 bytes)
Feel free to explore and hack the source code. Just read the POV-Ray legal documentation about distribution.

Pentium Optimized POV 2.2 Executable (Unofficial)
pentpov.zip (306,757 bytes)
pentpov.exe is a Pentium optimised version of povray 2.2, compiled using the Watcom 9.5 compiler. I've modified the source to only check for keypress at the end of every line, which saves a few seconds - it effectively means there is no difference in speed between the +x and -x options, and gives the luxury of being able to abort while running full speed. Harry's Note: This executable also runs fine on a 486.

Windows NT port of POV-Ray v2.2 Raytracer (Unofficial)
pov22-nt.zip (164,118 bytes)

Windows NT console mode POV 2.2 Executable (Unoffical)
povnt.zip (131,780 bytes)
Windows NT console mode, POV-Ray 2.2 executable, Optimized for Pentium. Built with MS Visual C++ 2.0. P.S. Dan Goldwater did the hard stuff. Derek W. Taylor, taylordw@sage.inel.gov.

POV-Ray v2.2 Executable (Unofficial)
povfast.zip (252,690 bytes)
Watcom COMPILE. Harry's Note: On average, I have found this executable to be ~23% faster than the official POV-

Ray compile. It is the executable I normally use,

POV-Ray 2.2 Executable (Unofficial)
ftpov2.zip (186,707 bytes)
FasterThanPOV.EXE Ver2.0. A modified and speed-up version of POV-Ray 2.2. Typical 2-3x faster, sometimes 20x !!! This archive contains a 32bit executable for MS-DOS. It was compiled using DJGPP 1.11 and needs a 386/387, 486 or Pentium. Harry's Note: If memory serves, this compile has implemented Eric Haine's Light Buffer code. It also uses non-standard command line switches, so it will not work with current front ends.

Converter Programs

Converts 3DS to POV, Vivid or Polyray
3dspov18.zip (131,714 bytes)
3DS2POV v1.8. Converts 3D Studio .3DS files to POV-Ray 1.0/2.0, Vivid, or Polyray raytracer formats. Supports animation. Now converts basic material properties (colour, ambient, diffuse, reflection, and transparency).

Converts USGS DEM files to .TGA
dosdem.zip (2,849,322 bytes)
This is a program to convert U.S. Geological Survey DEM data files to .TGA files for use with POVRAY. Harry's Note: Useful for absolutely realistic height fields to build terrains for use with ray tracers. Do plan on having enough memory. USGS DEM files are 9 meg in size.

V1.06 AutoCAD DXF file to POV File Converter
dxf2pv16.zip (27,723 bytes)

dxf2v -- strip 3d faces out of a .dfx file
dxf2raw2.zip (23,325 bytes)

Moray to Polyray file converter
mryply.zip (40,638 bytes)
Moray -> Polyray converter v1.0 by Alexander Enzmann, 4 August 1994. This program converts the .MDL files created by Moray v1.3 and v1.5 into scene files that can be rendered by Polyray (either v1.6a or v1.7). The program Moray is a Shareware graphical modeller originally created to produce scene files for the POV-Ray raytracer. Future versions of Moray will directly support Polyray, but in the interim this program can be used for building scenes.

Converts OBJ files to 3D-Studio ASCII Files
obj2asc.zip (26,349 bytes)

Polygon to Triangle Converter by Steve Anger
poly2tri.zip (24,280 bytes)
This program is commited to the public domain. Feel free to use all or part of this code in your own programs. This is a utility that breaks polygon shapes down into individual triangles. The polygons are assumed to be planar however the program should be able to tolerate slightly non-planar shapes.

Converts RAW Data File to Polyray Format
raw2ply.zip (13,865 bytes)
This program will take a file of RAW triangle data and convert it into a PolyRay include file.

Raw to 3D Studio Converter v1.1 by Steve Anger
raw3ds11.zip (51,873 bytes)
Raw23DS is a utility to convert lists of triangle coordinates in raw ASCII text format to 3DS ASCII save files which can be loaded directly into 3D Studio.

RAW to POV v1.8 for POV, Vivid, Polyray
rawpov18.zip (121,385 bytes)
RAW2POV v1.8. Converts lists of triangle coords in RAW ascii format to

POV-Ray 1.0/2.0, Vivid, or Polyray raytracer formats. Adjustable levels of smoothing can be applied to the triangle surfaces.

Rob's Triangle Smoother for Windows
rtswin.zip (11,510 bytes)
RTSWIN converts raw data files as output by POVCAD (and other programs) into smooth ray-traceable objects for POV 1, POV 2, and Polyray. RTSWIN is sort of a post-release extension of POVCAD 4. You don't need to have POVCAD 4 to get RTSWIN to work, but you'll need the files.

POV Surface Normal Calculator
smooth50.zip (195,601 bytes)
SMOOTH v5.0 - Raw Triangle Utility. Calculates surface normals for raw triangle data and creates a variety of output formats. Fully supports POV v1.0 & v2.0 with limited support for MTV (NFF), Vivid 1.0/2.0, Polyray 1.6, DKBTrace, POV 0.5, Rayce 2.7, CTDS 3.0, Rtrace 8.30 (SCN) and Rayshade. Includes a center utility that centers triangle data about the point 0,0,0, a scale utility that scales triangle data on any axis, a utility to extract triangle points from scene files, and a preview mode to view triangle files. (Freeware)

Convert TDDD, Imagine or QuickSilver to 3DS
tdd2asc.zip (148,435 bytes)
TDD2ASC will convert a TDDD, Imagine or Quick Silver object to a 3D Studio ASCII file (for importing).

Converts TDDD to Other Formats
2ddd2ray.zip (77,342 bytes)

Turns an Existing .TGA File into a POV Array
tgadot.zip (16,051 bytes)
TGADOT is a utility to turn an existing .tga file into an array of POV objects. An object will be created for every pixel in the scene, each with a colour depending on the corresponding pixel in the .tga file.

View Point object to RAW converter
vp2raw.zip (34,250 bytes)

Save 3D Models to Various Formats
wc2pov25.zip (530,231 bytes)
This program allows 3d models to be read, viewed, modified, and saved in several different formats:

- RAW - RAW 3d file, (Both import and export)
- NFF - Neutral file format as define by Eric Haines (Both import and export)
- TPO - TPoly files (Both import and export)
- OBJ - Wavefront 3d objects (Both import and export)
- GEO - AOFF files (Both import and export)
- DXF - A 3d subset (Both import and export)
- POV - POVRay V2.x (Export only)
- 3DS - 3D studio support - added for this version.

Harry's Note: I LOVE this program. Get It! Written by Keith D. Rule, the editor of PovZine. Keith is working on version 3.0. FREEWARE !

GUI Front-ends

Bob's POV-Ray v1.0/2.0 Front End
bpfe10.zip (75,543 bytes)
Bob's POV-Ray Front End, BPOVFE, provides a Windows point and click interface for specifying command line options for the Persistence of Vision Raytracer, POV-Ray. This program is freeware. Requires Windows and POV-Ray. From Bob's Software. Written by Bob Hayes. Version 1.0, January 15, 1994.

Simple Windows 3.1 POV launcher
launch.zip (9,096 bytes)

POV-Launch (Povlaun.exe) is a simple Windows 3.1 launcher for the Persistence of Vision raytracing program. It lets you select a .POV input file using a Windows 3.1 "open file" common dialog, and then runs Povray.exe.

POV Assistant Windows Front-End for POV 2.2
povass.zip (163,979 bytes)
Requires vbrun300.dll

POV COMMANDER (ver 1.3) for POV 2.0
povcom.zip (177,682 bytes)
This program is designed to make the use of the POV raytracing program easier. The documentation assumes that you have a basic knowledge of how either POV 1.0 or 2.0 works. POV Commander also allows you to assign other programs, 3D modellers, and ray-tracing utilities to User Configurable buttons - and the ability to run all of these and POV with a click of the mouse.

POV-RAY DOS Front End for POV 2.0
povctrl.zip(67,818 bytes)

POVMenu v2.1, 11/29/93
povmnu22.zip (182,054 bytes)
POVMenu v2.2 - Completly rewritten from scratch in Visual Basic For DOS. A project orented user interface for The Persistence Of Vision Raytracer (POV-Ray v2.0). Features include: Unlimited number of projects, clone from an existing project, calls editor with include file(s) loaded, support for 4 convert-to-GIF methods. ALL command line parameters for POV-Ray supported, render to screen, file, or both, online HyperText help, 6 user definable 'buttons', total control of all render options, configuration saved for EACH project, POVMenu is FreeWare!, and MANY MORE FEATURES. Mouse recomended. POVMenu can help you come up to speed with POV-Ray very quickly.

POVPANEL 2.0 -- A Control Panel for POV 2.0
povpan.zip (140,294 bytes)

PovShell FAQ (povshe.zip 3,293 bytes)
This is a summary of Frequently Asked Question on PovShell, the Development Environment to the POV-Raytracer. It also contains lists of bugs and new features coming. Written and uploaded by the author of PovShell, Andreas Peetz.

POVSHELL v3.0 Front End for POV 2.2
povshe30.zip (199,108 bytes)
POVSHELL is a user-friendly easy-to-use interface to POV-Ray. With POVSHELL you can: a) load and edit one or more POV-files. b) set all raytrace-parameters via pulldown-menus and option-windows. c) and finally call POV-Ray to trace the scene-file you are currently editing with the parameters you have set.

Windows Front-end for Running Polyray v1.7
pw17c.zip (137,361 bytes)

Modellers

Grand Unified Modeller v0.91, File 1 of 5
gum091ex.zip (534,194 bytes)
GUM, the interactive 3D modeller for Windows your 486 was waiting for. Supports POV, Polyray and Rayce. Features: Solid Evaluation, Trimmed surfaces, realtime pan and zoom, 3D direct manipulation. Written by Lex van der Sluijs.

Grand Unified Modeller v0.91, File 2 of 5
gum09dl.zip (96,722 bytes)

Grand Unified Modeller v0.91, File 3 of 5
gum091dm.zip (311,102 bytes)

Grand Unified Modeller v0.91, File 4 of 5
gum091hq.zip (431,241 bytes)

Grand Unified Modeller v0.91, File 5 of 5
gum091in.zip (3,788 bytes)

MicroLathe Version 1.5.1 for Windows
lathe151.zip (90,360 bytes)
MicroLathe is an easy to use modeling tool for Windows 3.1 that allows you to create three dimensional objects using the metaphor of the carpenter's lathe.

Midnight Modeller v2.0 beta for POV-Ray 2.x
mnm2be.zip (493,826 bytes)
MNM is one of the most powerful modelers for POV to date, if not the most powerful! MNM features very complex surface creation commands, and Point TOOLS only found in high end CAD programs costing $1,000+. Harry's Note: If you have QEMM 7.5 installed, make sure to turn off the DPMI host. With thanks from Dan Farmer!

Driver to allow MNM v2.0 beta to Run in a DOS Box
mnmwin.zip (7,992 bytes)
If you need run MNM v2.0 from a Windows 3.x DOS box, you must have WINDPMI.386 installed. This file provides uncommitted memory support and some floating-point support.

WinModel v0.2 POV modeller by Pete Goodwin
model2.zip (215,333 bytes)

Moray v1.53 Protected Mode Shareware Version
mray.zip (550,010 bytes)
This is an interim version that is a protected-mode, non-FPU version. After the flood of complaints and problems the real-mode, FPU shareware version was causing and after finding a couple of bugs that it had, I've decided to release this maintenance version. It has no artificial limits as far as memory is concerned, but you will be nagged to register if you use the program extensively. This version writes encrypted MDL files. The registered version can read encrypted and normal files and writes unencrypted MDL files. Harry's Note: I am a registered owner.

ProtoCAD 3D v2.00 Modeller for POV 2.2
pc3d2b.zip (341,255 bytes)
ProtoCAD 3D v2.00 <ASP> - New! Fast 3D CAD/Rendering program from TRIUS, Inc. Ultrafast Z-buffer technology combined with camera positioning produces amazing renderings. In combination with StarFlic, can produce flic file animations. True Vision TGA output. Trackball interface for camera positioning. DXF Import/Export. From TRIUS, Inc. $59.00 (+s&h). Harry's Note: This program only outputs a POV 1.0 scene file and provides only default colors. You cannot change POV specific items within ProtoCAD such as textures and lighting.

v0.99 PoVSB Pov Modeller For Windows
psb99_16.zip (721,651 bytes)
16 bit executable for Windows 3.1 and WFWG 3.11

v0.99 PoVSB Pov Modeller For Windows
psb99_32.zip (478,580 bytes)
32 bit executable for Windows NT and Windows/WFWG 3.11 with Win32s extenstions

PV3D Modeler v2.0 for POV-Ray 2.0 and Vivid 2.0
pv3dv2.zip (371,189 bytes)
PV3D modeler Version V2.00 For POVRAY 2.0 and VIVID 2.00 Graphics interface (GUI) with mouse. Many function are modified since the last V1.00 version !!! Include : 3D animation function 3D visualisation with camera /

look_at . Vectoriel object structure. XMS Support POV primitives support, Blob structure Height Field Shape, Mapping Texture Bumpping Function , GIF viewer VIVID 2.00 primitives support External Textures Library (POV/VIVID) Dynamic Rotate Move Scale (R-M-S) NEW! Support Groupe and Object Library CSG, Constructive Solide Geometry Direct generation of POV-RAY 2.0 files Direct generation of VIVID 2.0 files And more.and more ..., Splines, ... Smooth and Patch TXT shapes POV / VIVID Freeware Unregistered Version.

PV3D v2.0 Light Primitive Update
pv3d_a.zip (7,839 bytes)
For PV3D V 2.0 (SDF-SHA files). Light_Source and Area_Light Primitive. Freeware Unregistered Version

PV3D modeler Update File V2.10
pvupv210.zip (125,992 bytes)

POVCAD v4.01 modeller for POV and Polyray
pvcwn401.zip (360,626 bytes)
Shareware $35.00 registration. By Alfonso Hermida and Robert McGregor. Requires VBRUN300.DLL

Create 3D swept objects for POV-Ray and VIVID 2
rayl21.zip (77,360 bytes)
RayLathe is a text based tool that allows you to create three dimensional objects using the metaphor of the machinist's lathe. Objects can be created for POVRAY 1.0/2.0 & Vivid 2.0 ray tracing programs. Also .RAW output to many other tracers. uLathe (lathe151.zip) is an excellent front end to RayLathe. Bug fix release. Author: Ken Koehler v2.10

Text and Font Utilities

Converts .FNT & .Set into .INC files
fnt2pov2.zip (3,600 bytes)
BASIC source, no executable.

Converts .FNT & .Set into .INC files
Fnt2povf.zip, (59,251 bytes)
Includes the .FNT and .SET fonts

Character set made of tubes for POV 2.2
font.zip (7,020 bytes)
This is an include file for POV ver 2.2 that defines characters for use in POV. The characters are tubes instead of boxes like the ones included with POVRay. Characters can be easily resized. Includes pov source to render. By Michael Hartman.

Creates 3D fonts for POV and RAW Format
font3d11.zip (174,146 bytes)
Font3D is a utility for the creation of a three-dimensional character descriptions in a variety of formats including:

- POV-Ray triangles
- POV-Ray smooth_triangles
- RAW triangles

Any typeface can be used for which you have a TrueType font file description (*.TTF), and there are a number of other options available for fine-tuning the program's output.

Win TTF's to POV shapes Fonts
fs11r2.zip (116,187 bytes)
FS Fonts & bitmaps to shapes for POV. FS is a font & bitmap conversion program. Specifically, it converts MS-Windows fonts & bitmaps into a format compatible with the Persistence of Vision (POV) ray tracing program. FS is shareware and unregistered versions have a couple of minor limitations.

Create text objects for POV 2.0
gfont10.zip (83,125 bytes)
This is a simple utility to create text objects for POV. Requires MS-DOS and POV 2.0.

StringPV v1.0, Requires Font3d11.zip to Run
stringpv.zip (82,215 bytes)
By John Lagerquist, January 13, 1995, StringPV generates a POV include file for a given text string. This program requires Todd Prater's Font3D program

91 INC files of different fonts for POV-Ray v2.0
tms_rom2.zip (319,605 bytes)

Converts TT Fonts to 3D Objects, Vowels Crippled
tt2obt.zip (57,045 bytes)
This Windows 3.1 software converts TT-fonts to 3d objects including all sorts of bevel generation. You can convert single characters, the whole character-set or compose text strings as 3d object. This version is just a demo. The software works like the full version, but it will not convert any vowels.

Creates 3D fonts for VIVID/Poly/POV
vvfont18.zip (36,443 bytes)
VVFont reads Borland's chr stroke fonts and writes scripts for Stephen Coy's Vivid, Alexander Enzmann's Polyray, and Persistence of Vision. VVFont creates a file with fonts changed into raytrace primitives. Fonts are created with the width along the X-axis and the height along the Y-axis. Output is created in 2 basic styles--Round and Square. Square letters with no depth make Flat letters. Square letters with different top and bottom radii (radii are used because the ends of the strokes are cones and rings) make Tapered letters.

Convert VVFONT Output to CTDS Output
v2c.zip (10,939 bytes)

Harry Rowe runs the Windows World BBS which specializes in DOS and Windows based raytracing files. Harry can be reached at harry.rowe@wedowind.lexis-nexis.com, or ac401@dayton.wright.edu. His BBS may be reached at (513) 866 - 8181.
■

How to Find these Resources

Now that you've read about these tools, you may wonder how you can find them yourself. Here are several ways to find PovRay related tools.

Internet

Mosaic/NetScape

Try the URL ftp://ftp.povray.org/pub/povray. You will find several subdirectories. The most fruitful will probably be the utilities subdirectory. You should also check out some of the Web pages described on page 30.

Archie

If you know the name of the tool but don't know where to find it, then archie is a good place to start. If you don't know what archie is I would recommend getting the book "Internet Starter Kit", by Engst, Low, and Simon, ISBN: 1-56830-094-8.

ftp

If you don't have access to Mosiac or NetScape you can use ftp. The anonymous ftp site ftp.povray.org contains many PovRay related files.

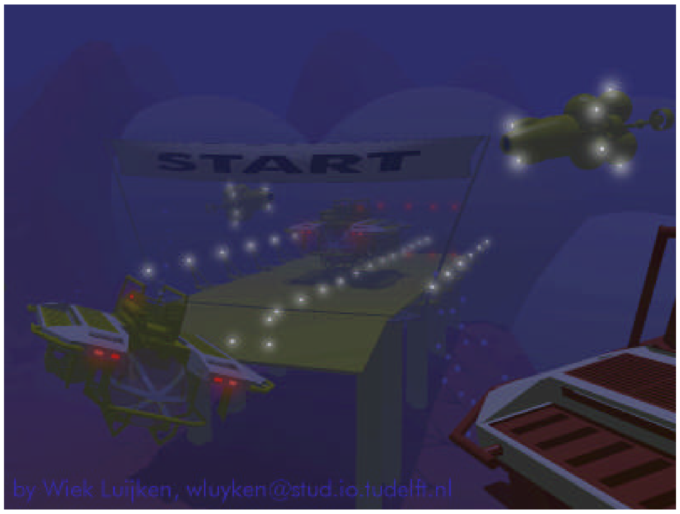
Online Services

There are several BBS systems that specialize in raytracing. Look in the classified section of this issue for references to two of them. Harry is the sysop for the Windows World BBS.

CompuServer's Graphic Developer's Forum (Go GRAPHDEV).
■

Art Gallery

Winner of the Dec 94 Raytracing Contest.



This image has been produced for the competition which has "games" as a theme. Instead of using a standard subject like a pool-table or a chessboard, I wanted to make something different. The result is something that is hard for some people to understand at first. It represents a submarine-race in the near future. The image showed here is a picture of the start of the race. There are three sub's of the formula 2000 class visible as are some camera-droids.

On the background there are some parts of an underwater base visible. Also there is a flag which represents the start of this endurance race.

The tools I used to create this image are:

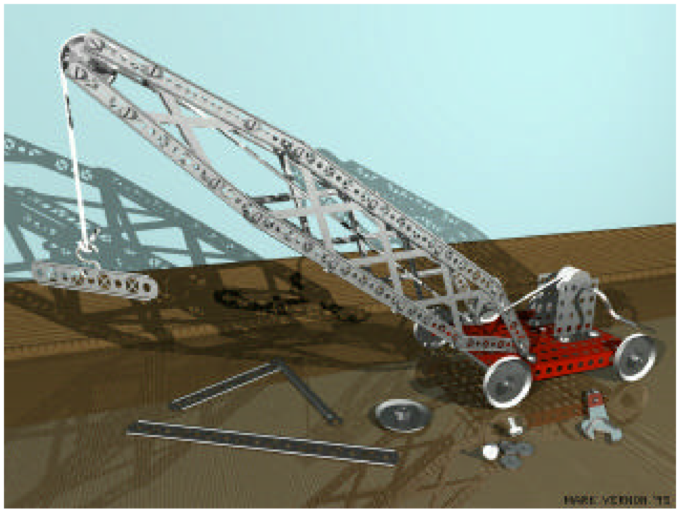
- GUM (0.89 registered version, you should all buy it, it's great, it's fantastic, it's brilliant, it's even better, it's getting better than 3D-STUDIO and Lex(the creator of GUM) is a friend of mine.
- Polyray 1.7, which I used for raytracing. It's a very fast raytracer, but has some problems with Unions.

The whole project took me about a month modeling time and rendering-time. The final image at 1200*900 took 3 hours on a dx2-50. The scene is big, the biggest I ever created. That's why I didn't raytrace the picture in one go. I traced the picture with one submarine and used this as a background for another scene with just one submarine and repeated this again.

This was done to keep the tracing time manageable in case something went wrong. The lights were a different story altogether. I tried to make them glow like they do now, using the formulas in the polyray example file "spot1.pi" but that took ages to raytrace. So the only solution was to use aldus photostyler and airbrush them by hand. I'm still trying to do it the right way, but it's difficult because the lights don't have ranges(like in 3d-studio) and keep shining on and on. The result of this is that everything makes shadows and is bright as the sun.

My address is :
wluyken@stud.io.tudelft.nl
Kloosterkade 208
2628 JJ Delft
The Netherlands

Winner Jan. 95 Raytracing Contest



Erector Set Crane

Mark Vernon
mvernon@metronet.com

This 640 X 480 32-bit rendering of a Crane made from erector set parts was created using PovRay 2.2 on a Mac II ci with an Applied Engineering Transwarp 40 MHz 68040 Accelerator and 20 Mb Ram. All calculations and coding were done by hand.

In trying to come up with an image for the January competition, a friend suggested doing something with an erector set. This sounded great but I wasn't sure if I could render the girder, pulleys, brackets, support members, screws, nuts, etc. necessary to build an object, but I could feel 'the challenge'. I picked up the gauntlet and tried to create a girder. Having accomplished that, I was off and shortly after that, the idea of a crane hit me.

Since this is, for all practical purposes, my first full-fledged rendering from start to finish, I must say that I learned a lot about memory and time efficient coding while doing this. I initially used a hand made bi-cubic patch for the rib on each side of the girder, but found out quickly how memory hungry bi-cubic patches are. I realized that I could difference a slightly smaller cylinder from a bigger cylinder (creating a hollow pipe) and then clip 1/2 of it to produce the same rib that the bi-cubic patch had. It reduced my memory by over 1/3 and significantly sped up the trace time.

Unfortunately it still grew to where 20 MB of ram was not enough. I rewrote the girder code so that instead of creating a box with a hole differenced out and then instantiating that many times with an offset, I created 1 long box and differenced out all of the holes as one, then added the side ribs as long ones rather than short ones on each instantiated piece. This reduced my memory requirement by almost 6 MB. Another trick that was recommended by a friend is when you need to difference out a round hole, use a cylinder rather than a sphere because a cylinder's radius is defined as the square root of the sum of the squares of 2 dimensions and a sphere is of 3 dimensions. This saves on additional math and rendering time.

I think that during the coarse of this project, I have now read 'Ray Tracing Creations' (by Drew Wells and Chris Young) as well as the PovRay docs all the way through several times.

I have included the scene files [See the anonymous ftp site ftp.povray.org for these files - KR]. Any recommendations or constructive criticisms for memory (it takes about 14Mb to render) and coding efficiency would be welcome.

Enjoy!

Mark Vernon

■

Tips & Tricks

Q Has anyone been able to program a succesful grid texture in POV? I've tried making a two layered texture, using marble without any turbulence and making one layer black with a touch of red (.99) and the second layer clear with a touch of red. Unfortunatly I can't seem to get it right. any ideas? - HB

A I've used layered gradients to do it. You don't say what it is about it that you "can't seem to get right". Essentially (without testing), you'd do something like this - Dan Farmer.

```
#include "colors.inc"
#include "textures.inc"

#declare Wireframe =
texture {
   pigment {
          gradient x
          color_map {
                 [0.1 color Red ]
                 [0.1 color Clear ]
          }
   }
}
texture {
   pigment {
          gradient y
          color_map {
                 [0.1 color Red ]
                 [0.1 color Clear ]
          }
   }
}
texture {
    pigment {
          gradient z
          color_map {
                 [0.1 color Red ]
                 [0.1 color Clear ]
          }
   }
}

sphere {<0,0,0>, 10 texture{Wireframe}}

camera {
   location <0, 1, 25>
   right 4/3*x
   direction z
   up y
   look_at <0, 1, 0>
}
```

Q How do you make a visible spotlight beam?

A

```
// POV-Ray 2.2 scene file
// by Kari Kivisalo kkivisal@vipunen.hut.fi
// Demonstrates use of a radial texture to
// make a spotlight beam

#include "colors.inc"
background { color Gray15 }

camera {
     location <0, 3, -7>
     direction z*1.3
     up y
     right x*1.33
     look_at <0, 1, 0>
}

//Color of spotlight
#declare cos=color rgbf<1,1,1,0.6>

//Radius of spotlight
#declare ros=1

//Corresponding value in color map
#declare rcm=ros/360

//Fall off angle of spotlight
#declare fos=25

//Corresponding value in color map
#declare focm=fos/360

#declare beam =
plane{y,0
  texture{
    pigment{
      radial
      color_map{
        [0.00      color Clear]
        [0.5-focm  color Clear]
        [0.5-rcm   color cos]
        [0.50      color cos]
        [0.5+rcm   color cos]
        [0.5+focm  color Clear]
        [1.00      color Clear]
      }
    }
    finish{ambient 1 diffuse 0}
  }
  rotate<-90,0,90>
}

#declare spot=
union{
  object{beam}
  light_source { <0, 0, 0> color cos
    spotlight
    point_at <0, -1, 0>
    tightness 0
    radius ros
```

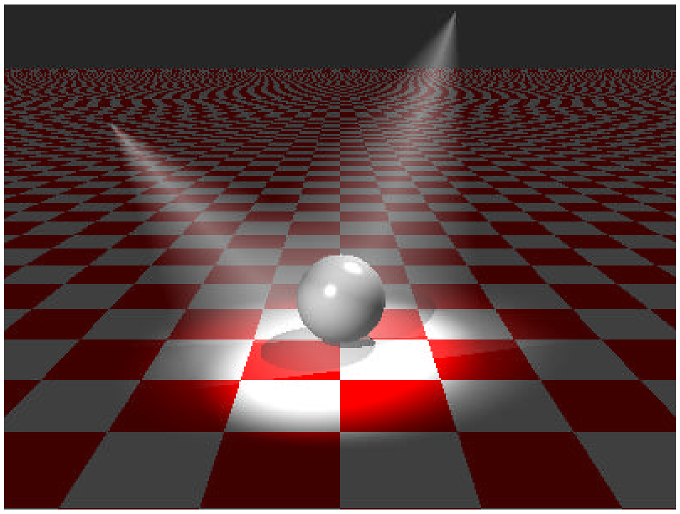
```
      falloff fos
    }
    no_shadow
  }


  object{
    spot translate <0, 4, 0>
    rotate < 15, 0,-20>
  }

  object{
    spot rotate x*10 translate<0, 4, 0>
    rotate <-40, 0, 30>
  }

  plane { y, 0
    texture {
      pigment {
            checker color Red color White}
      finish { diffuse 0.75 ambient 0.25 }
    }
  }

  sphere{<0,0.5,0>,0.5
    pigment{color rgb<1,1,1>}
    finish{diffuse 0.8 phong 1}
  }
```



Q How can I speed up the rendering of my image?

A If you have access to more than one PC (perhaps at work) and need to render a complex picture or one that will take a lot of time then you need GLUETGA.EXE written by Aaron A. Collins.

GLUETGA joins partially rendered TGA files into one TGA file. POVRAY can be instructed to produce a partial TGA file (a horizontal strip) by using the +s and +e switches.

For example, to render a picture on 4 machines run the following commands (one on each PC):

```
povray -imyscene.pov +e0.25 +ot1.tga
povray -imyscene.pov +s0.25 +e0.50 +ot2.tga
povray -imyscene.pov +s0.50 +e0.75 +ot3.tga
povray -imyscene.pov +s0.75        +ot4.tga
```

and then after moving the t?.tga files into one place the following command glues them together:

```
gluetga myscene t1 t2 t3 t4
```

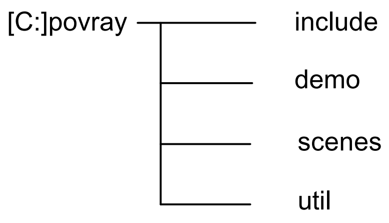
GLUETGA.ZIP is available from ftp.povray.org in /pub/povray/utilities

Chris Hart

Q I'm trying to render my first PovRay image and PovRay can't find the files colors.inc and textures.inc. Where can I download them?

A This is a common problem for beginning PovRay users. The include files for PovRay are found in the PovRay distribution in the file `povdoc.zip`.

The PovRay documentation suggest a directory structure as follows:



When you unzip the PovRay distribution files be sure you are at the toplevel of the directory structure and use the following command: `pkunzip -d povdoc.zip`. This will assure that the include subdirectory is properly created when the files are unzipped.
■

Letters

PovRay Wishlist

The POV team has stated that they are no longer accepting suggestions for the next release, but please keep sending your ideas. New features have a better chance of being implemented if they have already been coded, and programmers (hopefully) look at the list.

Features already guaranteed to be in the next release have been removed from the list. If you would like a list of features scheduled for the next release of POV-Ray, send mail to the address below.

Current Wishlist features (01/15/95)

Additional sphere image maps / Riemann sphere maps. Example:

All image borders meet at one point or pole or all image borders aligned along equator.

Starfield backgrounds / textures, optimized for single pixel emulation. Example:

```
texture { starfield <density> }
```

Wireframe / hidden line previews. Example:

command line `+q0` (quality zero)

Lensflare calculations. Example:

```
camera { flare <amount> }
```

Additional graphic output formats Example:

ppm output `+fp`

User defined textures via mathematical expressions. Example:

```
texture {
      pigment math
      sqrt(Px*Px+Py*Py+Pz*Pz)
      -colormap- }
```

Texture maps. Example:

```
texture_map {
      [0 1 texture White_Marble
      texture Jade] }
```

Irregular tile patterns for masonry. Example:

```
texture { masonry {
            texture Granite
            masonry2
            texture Jade ... } }
```

Function to return y vector at given point of a height_field. Example:

```
height_function {
<x,z> file_type "filename" }
```

If you have a suggestion or feature you wish added to this list please feel free to send it to janderson@comnet.usc.vcu.edu. He is the keeper of the 'official' wishlist. - KR

PovZine Download Size

You should release a TEXT only version for those who might not want to download a large file. The postscript format is alright and the content is great. But think of those who have slow connections and those who have no way of viewing post script. I hope my input helps you and good luck with further releases. - Arthur Phan

Thanks for the suggestion. As you've probably noticed, PovZine isn't exactly getting smaller. I will starting releasing a text version with this issue. - KR
■

Internet Resources

3D graphic links by others

`http://www.jyu.fi/~kuru/sirds/others.html`

Links to 3d images, SIRDS info, and technical papers. This is definitely a good starting point for browsing the web.

Eric G. Suchanek

`http://info.acm.org/~esuchanek/homepage.html`

If you're interested in using PovRay to visualize molecules then this is the home page for you.

sTaTiC DeSiGnS

`http://192.96.7.160/~kon/index.html`

A funky hompages with links to lots of graphics related stuff, including PovRay images.

Directory of /pub/povray

`ftp://ftp.povray.org/pub/povray/`

The base directory of ftp.povray.org. A great place to download just about any PovRay related thing. Lots of software, and images.

LEGO CAD

`http://www.rahul.net/gyugyi/legocad/legocad.html`

If you have fond childhood memorys of the those little building blocks, then check this out.

The Ray Tracing Home Page

`http://www.cm.cf.ac.uk:/Ray.Tracing/`

The title says it all.

Conferences & Workshops: SIGGRAPH 95

`http://www.siggraph.org/conferences/siggraph95/siggraph95.html`

If you are interested in the technical aspects of computer graphics then SIGGRAPH is the social event of the year.

Computer Graphics FUNdamentals

`http://www.hp.com/mhm/CompGrfxFUNdamentals/CompGrfxFUNdamentals.html`

A tongue-in-cheek explanation of computer graphics.

Stefan Maes WWW Homepage

`http://www.uia.ac.be/u/maes/index.html`

If you like raytracing, or DOOM check this out.

3dviewer

`http://www.render.com/oneday/viewer/`

If you'd like a free 3ds viewer, check this out.

3ds

`http://www.render.com/oneday/viewer/objects/3ds.html`

Need some 3ds models? Then browse this page.

POV-Ray CDROM

`http://www.cdrom.com/titles/pov.html`

Everyone's been talking about it. Here's where to get first hand info. ■