POVRAY MOVIE GENERATION

This is a howto on generating movies from POVRAY files.

Contents:

Ronald Marsh, Ph.D.
School of Electrical Engineering and Computer Science
University of North Dakota
Grand Forks, ND
October 31, 2019.

POVRAY

POVRAY (www.povray.org) is used to generate a more realistic looking scene than OpenGl alone can achieve.

POVRAY is available for Windows, Mac and Linux operating systems. However, please note that these instructions assume that you are installing POVRAY on a Linux system.

Installing POVRAY:

1. Downloaded the source tarball from: http://www.povray.org/download/
2. Untar using tar xvfz povray*.tar
3. Cd to the newly created povray directory
4. Configure by typing: ./configure COMPILED_BY="your name"
5. Su to "root"
6. Compile using "make" by typing: make
7. Test the binary by typing: make check (povray should generate a picture)

Using POVRAY:

POVRAY has many command line options, but to generate a basic image, you only need to use a few. Here is the command line syntax to use to generate and view the resulting image on the screen (does not save the resulting image):

> *Path_to_povray_executable* +i*Path_to_pov_file* –f +d +p +v +w*Width* +h*Height* +a.03 +L*Path_to_povray_include_files*

Here is the command line syntax to use to generate and save the resulting image to a file (does not view the resulting image on the screen):

> Path_to_povray_executable +i*Path_to_pov_file* **+fn -d** +p +v +w*Width* +h*Height* +a.03 +L*Path_to_povray_include_files* +o***Output_file_name***

Note that user supplied parameters are in italics.
Note that the differences between the two command line examples are boldfaced.

Where:

- Path_to_povray_executable – Probably /usr/local/bin/povray
- Path_to_pov_file – Example /usr/local/share/povray-3.6/scenes/advanced/desk.pov
- Width – Desired width of generated image
- Height – Desired height of generated image
- Path_to_povray_include_files – Probably /usr/local/share/povray-3.6/include
- Output_file_name – Depends on "f" flag, +fn indicates a .png file.

On the CS Cluster machine you will need to copy the povray.conf file from /etc/povray/3.7 to /home/yourname/.povray/3.7!

MPLAYER & MENCODER

MPlayer (http://www.mplayerhq.hu) is a movie player which runs on many systems and plays most MPEG/VOB, AVI, Ogg/OGM, VIVO, ASF/WMA/WMV, QT/MOV/MP4, RealMedia, Matroska, NUT, NuppelVideo, FLI, YUV4MPEG, FILM, RoQ, PVA files, supported by many native, XAnim, and Win32 DLL codecs. You can watch VideoCD, SVCD, DVD, 3ivx, DivX 3/4/5 and even WMV movies.

MEncoder is a movie generator companion to MPlayer.

Installing MPlayer:

1. Make a directory called MPLAYER.
2. Cd to the MPLAYER directory.
3. Download the mplayer source by typing:
   cvs -z3 -d:pserver:anonymous@mplayerhq.hu:/cvsroot/mplayer co -P main
4. Download the codec source by typing:
   cvs -z3 -d:pserver:anonymous@mplayerhq.hu:/cvsroot/ffmpeg co -P ffmpeg
5. Copy the codec source to the MPLAYER "main" directory by typing:
   cp -R ffmpeg/libavcodec main
6. and by typing:
   cp -R ffmpeg/libavutil main
7. Copy the codec make information to the MPLAYER "main" directory by typing:
   cp ffmpeg/common.mak main
8. and by typing:
   cp ffmpeg/common.mak main/libavcodec
9. Change to the MPLAYER "main" directory by typing: cd main
10. Su to "root"
11. Configure by typing: ./configure
12. Compile using "make" by typing: make
13. Install by typing: make install

Using MPlayer:

MPlayer has many command line options, but to play a basic movie, you only need to use a few. Here is the command line syntax to use to play an avi file:

Mplayer filename.avi

Using MEncoder:

MEncoder is the encoding (movie making) companion to MPlayer. MEncoder has **many** command line options, but to generate a basic movie, you only need to use a few. Here is the command line syntax to use to generate an avi file (without sound):

```
mencoder "mf://*.png" -mf fps=25 -o $3 -ovc lavc -lavcopts
vcodec=msmpeg4:vbitrate=14400:vhq
```

This example will compile all of the .png files in the current directory into a video with a frame of rate of 25 frames per second and will encode the resulting avi file with divx3 using 14,400 bits per second and very high quality. I have found that this set of parameters works very, generating a good quality movie, yet a movie that plays on most players. Note that you can decrease the avi file size (and movie quality) by lowering the bitrate, or you can increase the avi file size (and movie quality) by increaing the bitrate.

```
mencoder “mf://@list.txt” -o balloon.avi -mf fps=3:type=jpg:w=640:h=480 -ovc
lavc -lavcopts vcodec=mpeg4 -oac copy
```

This example will compile all of the .jpg files listed in the file “list.txt” into a video with a frame of rate of 3 frames per second and will encode the resulting avi file with mpeg4. The advantage of using a file that lists all of the files to include, is that some have reported a limit of 7500 files when using the *.jpg (as in our first example) method.

* Note that there are many additional encoding options, but not all are well supported.

Here is the command line syntax to use to merge and existing avi file with a sound file:

```
mencoder -oac copy -ovc copy -o output.avi -audiofile sound.mp3 input.avi
```

This example will **merge** the sound file sound.mp3 with the soundless avi file input.avi creating the file output.avi.

For more examples on codecs see:
http://www.mplayerhq.hu/DOCS/HTML/en/mencoder.html

GRIDRAM

GridRAM is a program I wrote to spread the work of rendering the POVRAY files across a single-core or a multi-core machine. GridRAM requires an initialization file to specify the cluster parameters. This initialization file (gridRAM.ini) is shown below:

```
in_extension__: .pov
out_extension_: .png
core_per_node_: 32
POV_data_path_: /home/rmarsh/HPC/GRIDRAM-THREADS/BIN
#
#Data_process_script:
#
/usr/bin/povray +L/usr/local/share/POVRAY/include +A0.3 +W640 +H480
+I$1 +FN -D -GA +WL0 -V +O$2
```

Where:

- **in_extension** specifies the extension of the files to process.
- **out_extension** specifies the extension of the resultant files.
- **core_per_node** specifies the number of cores/threads you want to use.
- **POV_data_path** specifies the **full** path to the .pov files.

Note that the gridRAM.ini file also includes a version of the POVRAY script shown in the **Data_process_script** section. This script is read in and parsed to insert the proper filename (note the UNIX $ variables) where required. Most importantly, this allows an experienced user to modify the scripts to refine how POVRAY renders their scenes. For example, one set of parameters the user may want to change is the width and height of the rendered scenes – which is specified in the POVRAY script as width=640 and height=480 (+W640 +H480 respectively).

Using gridRAM:

To use gridRAM copy it into the directory where the pov files are located and type: gridRAM **parameter**. Where **parameter** is one of:

- single – To run gridRAM with a single thread.
- thread - To run gridRAM on a multi-core machine (the number of threads used is specified in the gridRAM.ini file - core_per_node).

- **MOVIERAM**

MovieRAM is a program I wrote to simplify the task of generating movies using MEncoder. MovieRAM also requires an initialization file to specify the movie generation parameters. This initialization file (movieRAM.ini) is shown below:

```
include_title_: yes
frame_path____: /home/rmarsh/HPC/GRIDRAM-THREADS/BIN/FRAMES
title_frame___: aFrame
title_time____: 1
UND_frame_____: bFrame
UND_time______: 1
credits_frame_: yFrame
credits_time__: 1
closing_frame_: zFrame
closing_time__: 1
sound_path____: /home/rmarsh/ HPC/GRIDRAM-THREADS/BIN/SOUNDS
sound_file_$5_: 6000.mp3
video_name_$3_: temp.avi
movie_name_$4_: desk.avi
#
#MPLAYER_video_endcoding_command:
#
mencoder "mf://*.png" -mf fps=25 -o $3 -ovc -lavcopts vcodec=msmpeg4
#
#MPLAYER_audio_insertion_command:
#
mencoder -oac copy -ovc copy -o $4 -audiofile $5 $3
```

For CSci-446 (Computer Graphics) I require each movie to have 2 pre-movie title frames (aFrame and bFrame) and two post-movie credit frames (yFrame and zFrame). The first parameter in the file determines if these files will be included or not (the options are "yes" or "no"). Parameter 2 specifies the **full** path to the title and credit frames. Parameters 3 through 10 specify the title and credit frame name and the number of seconds each frame should play in the movie. Parameter 11 specifies the **full** path to the sound files. Parameter 12 specifies the name of the audio file. Parameter 13 specifies the name of the video file (avi file without sound) and parameter 14 specifies the name of the final movie file (avi file with sound). Note that the title frames (aFrame and bFrame) and two post-movie credit frames (yFrame and zFrame) are also png files.

Note that the movieRAM.ini file also includes variations of the MEncoder scripts shown in the MPlayer & MEncoder section. These scripts are read in and parsed to insert the proper filename (note the UNIX $ variables) where required. Most importantly, this allows an experienced user to modify the scripts to refine how MEncoder encodes their movie.

Using movieRAM:

To use movieRAM copy it into the directory where the png files are located and type: movieRAM parameter. Where parameter is one of:

- encode -To encode all png files into an avi file with no sound.
- sound - To merge the sound file and video file.
- both - To encode all png files and merge the sound file into an avi file.

The first title frame (aFrame.png) is where you would put your movie's title. The second title frame (bFrame.png) holds UND/CSci information and **MUST NOT** be changed. The first credit frame (yFrame.png) is where you would put any additional credit information (ie sound track info). The second credit frame (zFrame.png) holds UND/CSci information and **MUST NOT** be changed.

Note that the title and credit frames have been labeled such that they will always appear in the proper location in the final avi. But, only if you name your frames such that the first letter of your individual png files is greater than "b" and less than "y".

Finally, gridRAM, movieRAM, catavi, the 2 title frames, and the 2 credits frames are all in the gridRAM tarball.

CATAVI

In many cases it may be easier to generate movie segments and to concatenate those segments into a larger movie. Catavi allows a user to do this.

Catavi is a perl script written by “rapskat” and posted to the newsgroup “comp.os.linux.advocacy” on Wed, Mar 5, 2003 at 6:24 pm. Catavi will allow the user to concatenate 2 avi files into 1 avi file. Catavi is also part of the GridRAM tarball.

Note: catavi does not adjust the avi header. Therefore, mixing avi different file types (different encodings) is problematic. Also mixing avi files with different sound formats is problematic.

Using Catavi:

Catavi has two command line options. Like the UNIX version of “cat”, one command line option specifies the file to concatenate to and one command line option specifies the file to concatenate with. For example:

```
Catavi file1.avi file2.avi
```

At this point catavi will prompt the user for more details.