AMP

COLLABORATORS

	TITLE : AMP		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 14, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AMP

1.1 AmiDog's Movie Player - The Movie Player for your PPC Amiga!

```
                 AmiDog's Movie Player v1.03, 990605

       Copyright 1998-99

    Mathias "AmiDog" Roslund
```

Visit the homepage at http://www.amidog.com/amp/

1.2 AmiDog's Movie Player - Introduction

This is a Movie Player for Amigas equiped with a PPC board.

The MPEG1/2 decoder is based on mpeg2decode v1.2 from the MPEG Software Simulation Group.

I'm planning to add AVI/QT support as well, I've not yet decided if I'm going to use
the Xanim source or not, that'll be decided as soon as the MPEG plugin is finished.

Please consider this product as Shareware, which means that you can freely use it
for a period of 30 days, but then you must send the author a small fee. Please note
that there is no special registered version, you wont even get a keyfile, but you
will probably be able to sleep much better at night!

Now start the player and enjoy the speed of your PPC board!

1.3 AmiDog's Movie Player - Requirements

Hardware:

* Amiga 1200/3000/4000 with a PowerPC board
* 4 MB RAM
* OS 3.0+

Software:

```
  * WarpUP
  * CGFX v3 or later for CGFX support
```

1.4 AmiDog's Movie Player - Features

```
It currently supports the following multimedia types:

* MPEG1/2 video with sound (optional).
* ...
```

1.5 AmiDog's Movie Player - Installation

```
Just copy it to any place on your HD.
```

1.6 AmiDog's Movie Player - Usage

```
1. Start shell.
2. stack 100000 <ENTER>.
3. amp -gui or amp "filename" <ENTER>.

NOTE: If AMP crashes, try to increase the stack!
```

1.7 AmiDog's Movie Player - Menus

```
                File

                General

                Output

                MPEG

                Misc
                Please click on a menu item.
```

1.8 AmiDog's Movie Player - Menus - File

```
                File

                General

                Output
                MPEG

                Misc
                Open - Select a MPEG to play.

Play - Plays the selected MPEG.


               Edit Prefs
                - Here you change the screenmode preferences.

Save Prefs - Saves the preferences.

About - Clears the listview and prints the about information in there.

Quit - Quits AMP.


NOTE: When the preferences are saved, not only the screenmode preferences are  ←
   saved,
      but also all current settings like HAM Quality, ColorMode etc.
```

1.9 AmiDog's Movie Player - Menus - File - Edit Prefs

```
                Back
                When you choose Edit Prefs, AMP will open a special window in  ←
                   which you can
tell AMP which screenmodes to use.

There are two lists:

* Default ScreenModes - This is the list AMP will use when you have selected  ←
   Default
                        as the screenmode to use, AMP will start at the top and  ←
                           move
                        down until it finds a screenmode which is big enough to  ←
                           show
                        the whole MPEG. One special case exist, adding a  ←
                           screenmode
                        with width and height zero (0) will make AMP use this one ←
                            in
                        case it can't find one that fits.

* Custom Screenmodes - These screenmodes are the one you can force AMP to use,  ←
   choosing
                       one of these always make AMP try to use it, even if it's  ←
                          way too
                       small. The first screenmode in this list always is Default ←
                          , this
                       screenmode can not be deleted or modified.
```

```
Both lists are automatically sorted to make sure that AMP always selects the  ←
   right
screenmode for the loaded MPEG.

Below each list there are three buttons, Add, Modify and Delete which will do  ←
   just that,
add, modify or delete the selected item in the list.

Above the Save, Use and Cancel buttons are a set of edit gadgets, which you use  ←
   to modify
or add screenmodes. By pressing the Choose button, you'll be able to select a  ←
   screenmode
using an ASL requester.

There are two things to remember:

* You can not have two screenmodes with the same width and height in the same  ←
   list!

* In the Custom ScreenModes list, you can not have two screenmodes with the same  ←
   name!
```

1.10 AmiDog's Movie Player - Menus - General

```
                File

                General

                Output

                MPEG

                Misc
                Screenmode:

* Default/?/... - Which screenmode to use. Default will make AMP select the best  ←
   one from
                  the default screenmodes list, any other will force that  ←
                     screenmode.

Limit FPS:

 * From Stream/5/10/15/20/25/30/Maximum - How fast the MPEG should be played.  ←
    NOTE: This
                                          setting is ingored when audio is  ←
                                             enabled since
                                          the audioroutine generates the correct  ←
                                             fps.

Sound:

 * On/Off - Turns sound On/Off.
```

1.11 AmiDog's Movie Player - Menus - Output

```
                 File

                 General

                 Output

                 MPEG

                 Misc
                 ColorMode:

 * Gray  (4/6/8 bit) - 16/64/256 colors, gray display.
 * Color (8 bit)     - 256 colors, ordered dither.
 * HAM   (6/8 bit)   - HAM6 or HAM8.

Gray depth:

 * 4/6/8 - 16/64/256 gray colors, lower depth is faster due to less CHIP accesses ↩
     .

HAM Depth:

 * 6/8 - HAM6 (4096 colors) or HAM8 (262144 colors).


                HAM Width
                :

 * 1/2/4 - The amount of HAM pixels per RGB pixel.


                HAM Quality
                :

 * Normal/High - Only affects 1/2-width, biggest difference on 1-width.


                HAM S-Lores
                :

 * On/Off - If On, AMP will use SuperLores (PAL only) for MPEGs up to 160*128.

Size:

 * 50%/100%/200%/Full screen - The size at which the MPEG is played.


NOTE: On gfx boards the minimum depth is 8 (automatically adjusted if required).
```

1.12 AmiDog's Movie Player - Menus - Output - HAM Width

```
                Back
                To understand this you must first know the basics about how HAM  ↩
                   modes work. I'm
not an expert on this, and just before christmas last year (1998), I didn't know
much at all. Anyway, here goes.

On a HAM screen a pixel is either one from the 16/64 colors in the palette or a
a "finetuned" version of the pixel next to the left, that is, you can modify one
of the three color components Red, Green, or Blue (RGB). Two bits per pixel is
used to determine if "finetune" or palette color should be used. That is why a
HAM6 screen is 6bit but only gives you 16 colors (4bit).

This means that to get one specific RGB value, you must either make sure that it' ↩
   s
one of the 16/64 colors in the palette, or you must use three (3) pixels on  ↩
   screen
to achieve the correct RGB values. This is why AMP supports different HAM widths.

As you might know, there are no 3-width screenmodes available, for example, there
are 320*256 (1-width), 640*256 (2-width) and 1280*256 (4-width) for PAL. So to be
able to get the right RGB value, you must use a screen which is four (4) times as
wide as it is heigh. Since you then will get the right RGB value, these screens  ↩
   are
often callen 12bit/18bit since you get 4/6bit per RGB component and there are a
total of three components per pixel (2^(4+4+4)=2^12=12bit, 2^(6+6+6)=2^18=18bit).

The major let down by using 4-width is that it only works on PAL/NTSC/HighGFX and
that it's terribly slow. That's where the 1-width and 2-width modes comes in.  ↩
   They
tries to achieve the best possible quality with less pixels.

The 2-width uses a simple but very efficient theory about in which order the RGB
components should be changed to get the best quality while only changing two of
them per pixel.

The 1-width uses a similar theory (which I've invented myself) but ofcourse  ↩
   produces
a worse quality, it also requires more computations per pixel and thus is slower, ↩
    but
since a PPC is very powerful, and the main bottleneck is the CHIP accesses, you  ↩
   will
actually get a faster playback using 1-width.

I hope the text above explains why certain HAM widths only work with certain MPEG
sizes, it's for example impossible to use 2-width on anything bigger than 640*512
since there are no available screenmodes, and if you use a VGA monitor, you can't
display any screenmode larger than 640*512 (overscan not counted) which means  ↩
   that
2-width is only supported for MPEGs up to 320*256.
```

1.13 AmiDog's Movie Player - Menus - Output - HAM Quality

```
                Back
                Please read the text about HAM Width first if not already done.

Below you can read what the difference is when using High instead of Normal ↩
   Quality.

2-width - The difference in quality is very tiny here. What AMP does is that it ↩
   takes
          in account which RGB components that most urgently need to be changed ↩
             in
          order to (mathematically) give the best result. On some MPEGs this can
          actually reduce the quality.

1-width - Here the difference in quality is quite hughe. When using high quality
          on this mode, AMP will use a predefined palette as well as using the ↩
             normal
          "finetuning". For each pixel, AMP will calculate something called " ↩
             color
          distance" to determine if it should use one of the 16/64 colors in the
          palette, or "finetuning". This ofcourse makes it even slower, but the
          quality is much better, and this mode works on every monitor, including ↩
              VGA.

NOTE: 1-width high quality is the only mode which uses the palette, all others ↩
   rely on
      "finetuning" for displaying the image.
```

1.14 AmiDog's Movie Player - Menus - Output - HAM S-Lores

```
                Back
                The Amiga actually supports something called Super Lores which ↩
                   aren't used very often.
It's simple the same as a Lores screenmode but with half the height, ie PAL Lores ↩
    is
320*256, and PAL Super Lores is 320*128. I found out that Super Lores also work ↩
   on Hires
screens, ie PAL Hires Super Lores is 640*128.

By using these modes, AMP is able to display a MPEG up to 160*128 in fullscreen ↩
   using
2/4-width HAM since these modes requires a screen which is 2/4 times as wide as ↩
   it's
height. As a small bonus, these MPEGs will be played faster and in fullscreen.

The difference in speed between normal PAL Lores and PAL Super Lores, and which ↩
   is most
clearly visible in 4-width HAM, is due to less CHIP memory bandwidth being used ↩
   by the
```

custom chipset since the screen is smaller, this also gives an idea about how ↩
slow AGA
is when using HAM8.

NOTE: Super Lores is only supported for PAL and 2/4-width HAM.

1.15 AmiDog's Movie Player - Menus - MPEG

File

General

Output

MPEG

Misc

Interpolation
:

* On/Off - On gives a slightly better YUV->RGB quality but is way slower.

Color Quality:

* Normal/High - High gives better quality on 4:2:0 interlaced MPEGs, all
  other MPEGs are not affected.

1.16 AmiDog's Movie Player - Menus - MPEG - Interpolation

Back

I will not get too technical here, but I hope that you all will ↩
have some idea about
what Interpolation is when you've read this text.

First some information about why interpolation "is" nesessary for MPEGs.

MPEGs are not stored in RGB as many pictures formats do, but instead in YUV, a ↩
way which
separates the contrast and color components, the Y value is alwasy the contrast ↩
of the
pixel, while UV gives the color (this is not really true, but it makes it easier ↩
for you
to understand).

A MPEG don't very often have both Y and UV values for all pixels, for example, ↩
4:2:0 means
that Y is there for all pixel, but UV are only there for every other pixel in ↩
every other

```
row, look below.

YUV Y.. YUV Y..
Y.. Y.. Y.. Y..
YUV Y.. YUV Y..
Y.. Y.. Y.. Y..

This is where interpolation comes in. Interpolation is a way of "finding out"  ↩
   which value
the UV components should have had if they would have been supplied. This can be  ↩
   done in
several ways, by either just looking at the pixels next to the "missing" one, or  ↩
   by looking
at several pixels.

Using interpolation will require quite a lot of computations. Depending on how  ↩
   the MPEG is
stored, the amount will differ. A 4:2:2 MPEG only reuires one pass of  ↩
   interpolation per
frame (horizontal), while a 4:2:0 MPEG requires two passes (horizontal+vertical).

Using interpolation will ofcourse give a better quality when later doing the YUV ↩
   ->RGB
conversion which is required since very few gfx boards support YUV directly. The  ↩
   RGB
components will be used to display the image in HAM6/8 or 15/16/24/32bit.

Since the Y value is always there for every pixel, gray will always get a very  ↩
   good quality
and interpolation is only used when required (ie when doing YUV->RGB conversion).

Conclusion: Using interpolation will improve the quality, but since the Y value  ↩
   is there for
every pixel, the result will not get very big, and when it comes to moving frames ↩
   , perhaps
at 25fps, then your eyes aren't fast enough to tell the difference. Perhaps if I  ↩
   add the
possibility to save the decoded MPEG as some kind of ANIM format, it might be  ↩
   usefull.
```

1.17 AmiDog's Movie Player - Menus - Misc

```
                File

                General

                Output

                MPEG

                Misc
                Statistics:
```

```
 * Normal   - Shows size of the MPEG and FPS.
 * Detailed - Show same as Normal plus screensize and chunkybuffer width (only ↩
   for debugging).
 * Off      - Shows no statistics at all.
```

1.18 AmiDog's Movie Player - Speed

```
CGFX speed: (Measurements using v1.02, CGFX v3 and a BVision and a 240*176 MPEG)

 -   Color Mode  -    AMP    - IsisPPC -

 * 8bit (Gray)   : 50.8 fps. :   N/A.
 * 8bit (Color)  : 42.4 fps. : 27.9 fps.
 * 16bit (Color) : 37.3 fps. : 29.4 fps.
 * 24bit (Color) : 37.2 fps. : 28.9 fps.

AGA speed : (Measurements using v1.01, AGA in PAL Lores and a 240*176 MPEG)

 -   Color Mode  -    AMP    - IsisPPC -

 * 4bit (Gray)   : 40.2 fps. :   N/A.
 * 6bit (Gray)   : 37.3 fps. :   N/A.
 * 8bit (Gray)   : 33.8 fps. :   N/A.
 * 8bit (Color)  : 31.0 fps. : 12.9 fps.
 * HAM6 (1 Width): 21.6 fps. :   N/A.
 * HAM8 (1 Width): 20.4 fps. :   N/A.
 * HAM6 (2 Width): 22.1 fps. :   N/A.
 * HAM8 (2 Width): 19.4 fps. :   N/A.
 * HAM6 (4 Width): 20.3 fps. :   N/A.
 * HAM8 (4 Width): 12.2 fps. :   N/A.

 SuperLores speed: (Measurements using v1.01, AGA in PAL and a 160*120 MPEG)

 -   Color Mode  -    On     -   Off   -

 * HAM6 (2 Width): 41.6 fps. : 41.4 fps.
 * HAM8 (2 Width): 38.7 fps. : 37.1 fps.
 * HAM6 (4 Width): 38.1 fps. : 36.4 fps.
 * HAM8 (4 Width): 31.7 fps. : 22.8 fps.

NOTE1: For the HAM modes, Interpolation is Off and HAM Quality is Normal.

NOTE2: SuperLores PAL means a 640/320*128 screen, while a normal Lores is ↩
   640/320*256.

NOTE3: Of some reason, IsisPPC reports 163 frames the first time the 240*176 MPEG ↩
    is played,
       but 161 frames all of the following times, strange!

NOTE4: AMP is 2.40 times faster than IsisPPC on AGA using 8bit color and PAL.

The 240*176 MPEG is a Wallace and Gromit MPEG played from RAM.
The 160*120 MPEG is a Tintin MPEG played from RAM.
```

```
All tests are performed using an A1200T 603e'200, 040'25, 128MB and OS3.1 from ↩
   Workbench.
```

1.19 AmiDog's Movie Player - ShareWare

```
After your free trial period of 30 days, if you decide to keep using it,
please send $15, £10, 20DM, 100SEK or equal amount of any other currency to:

Mathias Roslund
Sveav. 2b, nb
702 14 Orebro
Sweden

Thanks!

NOTE: There is no special registered version, you wont even get a keyfile,
      but you will probably be able to sleep much better at night!
```

1.20 AmiDog's Movie Player - Disclaimer

```
Remember! You use this piece of software at your own risk!
I can never be held responsible for any sort of damage caused
to your software or hardware by the use of this product!

Bugreports and suggestions might be sent to one of my addresses.
```

1.21 AmiDog's Movie Player - Development

```
This product has been developed totaly by me using EGCS/GCC.

Since I'm a student, I just don't have the time to spend
several hours a day developing this product, especially
when approaching Christmas and summer since I then will have
a lot of schoolwork to finish. Therefor please don't write
to me and complain about the slow development! Thanks!

AmiDog's Movie Player is developed using:

v0.00-0.31   Amiga1200HD -> 040/FPU/MMU'25, 603e'200, 2+32MB, AGA.
v0.40-0.50   A1300Ti     -> 040/FPU/MMU'25, 603e'200, 2+128MB, AGA.
v1.00-       A1300Ti     -> 040/FPU/MMU'25, 603e'200, 2+128MB, BVision.
```

1.22 AmiDog's Movie Player - Bugs

```
* None known.

There might be more bugs, so you use it at your own risk!
```

1.23 AmiDog's Movie Player - Contact

```
Bugreports, suggestions, comments or anything else you may
want to contact me about can preferably be sent by e-mail to:

amidog@amidog.com

You may however also contact me by normal mail:

Mathias Roslund
Sveav. 2b, nb
S-702 14 Orebro
Sweden
```

1.24 AmiDog's Movie Player - Thanks!

```
I would like to thank the following persons:

  * Stefan Burström - For answering all my (stupid?) GCC PPC questions.
  * Jesper Svennevid - For giving me the C-only C2P and helping me getting it to ↩
     work
                     and for writing the sound routines.
  * Mikael Kalms - For helping me getting the 4bit C2P to do only 4bit C2P.
```

1.25 AmiDog's Movie Player - Future

```
This is what I currently plan to add, it is NOT in priority order, and it might
change without further notice!

* Frameskip.
* Custom playback size.
* Window playback.
* More speed.
* Preloading of small MPEGs.
* AVI/QT/ANIM/FLI/FLC... support.
```

1.26 AmiDog's Movie Player - History

```
v1.00 -990219
      -Included the MPEG audio code by Jesper Svennevid (it's VERY slow currently ↩
         ).
      -990402
      -Some changes to the code, prepared for 15bit+ support.
      -990405
      -The GUI has been changed and the source has been rearrenged and compiled.
      -Replaced the WPA8 CGFX support with a direct mem copy, the speed (using a  ↩
         160*120 MPEG)
       improved from 28fps to 38fps! Using AGA the speed is 30fps. (Using 8bit  ↩
          color).
      -AMP will now ALWAYS close the GUI and screen if it quits due to a faulty  ↩
         MPEG or
       any other reason there might be!
      -Added 15bit+ support and it works great! I've finished the 32bit (ARGB)  ↩
         mem copy
       routine which increased the speed from 25.3fps (using CGFX call) to 31.6 ↩
          fps!
       (Using the same 160*120 MPEG as above).
      -Minor optimization to the 32bit routine, up from 31.6fps to 31.9fps.
      -Finished the 16bit mem copy routine, 36.0fps using the same MPEG as above.

v1.01 -990411
      -Tried to find the bug which makes it impossible to save the prefs from AMP ↩
          just to
       find out that it always works if you start AMP from shell.
      -Added a delay to the GUI code so that AMP doesn't consume unnessesary 68k  ↩
         CPU power.
      -Minor speedups, the value within parentheses is the old value. 8bit 37.7  ↩
         (37.2),
       16bit 29.8 (29.5), 32bit 25.0 (24.9).
      -Replaced all SetRGB32 calls with LoadRGB32 calls, this is much faster!
      -Updated the speed measurements in this guide.
      -990412
      -Prefs is now loaded from PROGDIR:.
      -990417
      -It appears like the "can't save prefs" bug can't be fixed currently,  ↩
         therefore
       I've removed the icon, and more or less forced you all to start AMP from  ↩
          shell.
      -990418
      -Spent a few hours improving this guide, hope you all are satisfied with it ↩
         .

v1.02 -990428
      -I noticed that using MapROM or similar utility which moves the ROM (only  ↩
         required
       on A1200 where the ROM bus is 16bit) to RAM speeds things up a bit. 8bit  ↩
          color
       gives 42.4fps compared to 40.4fps.
      -Updated the CGFX speed measurements.
      -After reading the WarpOS GameDev guide (yes, I do use WarpUP) I got an  ↩
         idea of how
       to speed things up in 16/32bit. And the difference really is hughe! In 32 ↩
          bit I now
       get 37.2fps compared to 25.0fps which is a BIG difference. The difference  ↩
          for 16bit
       is not that big, "only" 37.3fps compared to 29.8fps.
      -As you can see, the 32bit is just as fast as 16bit, this is due to the  ↩
         fact that
       16bit requires more computations which slow things down.
      -Tried the same idea on 8bit gray, the difference is 50.8fps compared to  ↩
         47.8fps.

v1.03 -990517
      -After adding double buffered audio in AmiGenerator using audio.device, I  ↩
         decided
```

```
 to do the same for AMP, and it seams to work fairly well, I don't have any ↩
     good
 MPEGs to try it out with though.
-The audio is currently always played as mono (left channel).
-990518
-Encoded a MPEG on my P120 to try the sound stuff. And well, It's actually ↩
   better
 than Isis sound support (I had to use skip 80 in Isis to even get near ↩
    uninterrupted
 playback), but the sound is a little late (1sek?) and there are too much ↩
    sound
 decoded each time, giving an ugly result, as well as, some minor sound ↩
    interruption.
-I cleaned up the audio code a little, I still need to rewrite a lof of it ↩
   though.
-990520
-Rewritten most of the audio part, now it only decodes just as much as ↩
   required
 for each frame, the sound is however still a second late or some, but to ↩
    avoid
 this I'll have to you some kind of buffers so that there are some audio ↩
    loaded for
 the frame to be displayed... It's however MUCH better than Isis. Still ↩
    mono though,
 but the audio is decoded in stereo, I'll just have to change a few things ↩
    ...
-990528
-Thanks to you, the users, I have (hopefully) removed the bug which made ↩
   AMP crash
 when using GUI, even if you didn't do anything yourself!
-Removed one audio bug which did hang AMP when playing several MPEG with ↩
   sound
 after each other.
-The Limit FPS setting is now ignored when audio is on since the double ↩
   buffered
 audio.device routine automatically forces the right fps rate.
-990603
-Guess what!? I've been able to compile a WarpUP version, so from now on, ↩
   AMP is
 WarpUP only! I've also removed a bug which didn't appear in the PowerUP ↩
    version.
-990605
-Removed some audio bugs (and implented new ones?).
-Now stereo MPEGs is supported (untested!), if the current MPEG only got ↩
   mono audio,
 then the same sample will be played on both the left and right channel.
-The audiobuffers will now get emptied when the MPEG is stopped (which ↩
   causes a slight
 delay before the screen get's closed).
```