ats.hyper

COLLABORATORS			
	TITLE : ats.hyper		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 5, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ats.hyper

1.1 The accompanying documentation for AfterTitles.

```
   This is the main document (you start here).

 Short and concise info about what this program does.

 Limitations.

 Compiling the program.

 About me.

 Purpose.

 Genlock.

 Contents.

   Version 1.0, Tuesday 27-Jul-93 19:14:37
```

1.2 So, what is this ?

```
   This program reads in an IFF picture, and then
scrolls it from top to bottom. The effect will be
much the same as that you see, when a movie ends,
and a list of the cast scrolls up. In fact, this
is its intended purpose - you make a (large)
IFF picture with your 'cast', and then use this
program to play it for your video or genlock.

To see what it does, type "ats demo.ilbm" at
the shell prompt.

By the way, to be a little more exact, it

- reads in the picture.
    (the busy clock will then disappear, and
```

the screen turn to the proper colours).
– waits for a mouse or key press.
    (having received such, it will scroll the
    picture up, and stop when it reaches the
    bottom).
– waits for another keypress.
    (and then satisfied, it quits).

The initial reading of the picture can take
some time on a 7mhz 68000. Therefore, I
put in the first key-press pause to let it
be easy to trigger off the scrolling. In
the same spirit, the second key-press pause
Is there to avoid, you get a nice workbench
picture on your video. As can be seen from the
demo, the final halt can also be used as an
effect (no bullshit...).

1.3 Things, that should be better.

As can be seen from the included source, the
program is quite simple/primitive, and as
such has its limitations.

(NOTE THOUGH, THIS PROGRAM SHOULD WORK JUST
FINE WITH BOTH PAL AND NTSC).

    Limitation #1: To make my "job" easier,
the program needs at least 2.0. It is no
big task to make a program doing what this
one does, that would work on all Amigas, but
I did what you have here, as this was REALLY easy.

    Limitation #2: Right now, the program
insists on the picture being hires-interlace
(that is, no matter what resolution the picture
was originally in, it will be shown in hireslace).
This is really some limitation – although
640x512(400) is an ideal resolution for video,
it would have been very little work to
let people choose it themselves. However, it
seems superhires is not "described" in the ILBM
file, so if you paint a 1280x512 in superhires,
it is stored a bit like a 1280 pixel wide
page in 640x512 resolution. I cannot be
trusted on this; e-mail me if it is not so
and tell me, how things really work. I will
then probably fix it.

    Limitation #3: The picture can be up
to 8 bitplanes deep – I would be happy to
hear what it does on an A1200 or A4000
with a 256-colour picture. In theory it
should work, but I have no idea. HOWEVER
it is a bit more complicated:

On small Amigas, like my A500, 4 colours
and 2 colours work just fine. You can easily
load an 8 or 16 colour hires-interlace picture,
but on my own machine(A500), it flickers
like hell. This is quite silly, as no data
is moved – a stock A500 should have no
problems doing this. One day, if I figure
out how to do, what this program does,
properly (with views and such), I might fix
it.
   On larger machines, more than 4 colours
should pose no problems. Please tell me, if
it does.

   Limitation #4: I am so stupid, that I
don't even know how to take over the
machine :-) Or rather, I need to wait with
WaitTOF(), and as this is an OS waiting
function, it would enable multitasking (If I
had shut it off). Anyway, what problems
does that introduce to you ? Well, here
on my own machine, I do the following:
I reset my(2.0) machine, hold down both
mouse-buttons. On the boot menu, I disable
the startup-sequence. I then boot on the
partition, where my program is. I go down
to its drawer(by typing dh0:usr/newiff).
I then put an arbitrary disk in the disk
drive. Then, at last, I type ats demo.ilbm

   Why all that ? Well, if not, the
scrolling jumps! If there is no disk
in the disk drive, the drive clicks, and
my scrolling jumps. If I have started workbench,
then the scrolling jumps due to various
commodities, and probably something
else too.

   You can experiment with it, if you like.
The reason for the above inclusion of
a lengthy description of what I do, is so
that you can see, at least a way to make
the scrolling stable.

   Limitation #5: The program works
by opening a screen the size of your
picture. This means, that the whole
picture must fit into CHIP RAM. I don't
see this as a really big problem; a
640x3600 4 colour picture takes up
576k, and that, me thinks, is a
rather large scrolling list.
   The best thing to do would probably
be some kind of script, in which you
put your text, together with some
positioning and font information.
This would be the best to work with, but

large CG fonts tend to use up lots
of memory, so I dropped it again. But
I admit, that typing the list in a
paint program is not very userfriendly.
However, it is flexible :-) You can
make all sorts of neat effects, and
graphics too.

1.4   compiling

I said this program was really easy to
create. Well, apart from using 2.0, this
stems from using the NewIFF package
from CBM, which can be found on Fred
Fish disk #705 (it might be 704, but
I think it is 705). Older versions
of NewIFF would probably also work, I
do not use any advanced features.

I have included my very simple DMakefile
(works with Matt Dillons DMake, probably
also with normal make programs). You
can have various setups when using
CBM's NewIFF package, but the following
will work:
Place ats.c and Dmakefile in the "root"
drawer, that is, the one containing
drawers apps/, modules/ and iffp/.

   Be in that drawer, when you type
Dmake or Make or whatever (If you
no not use Dmake, you had better call
Dmakefile Makefile).

That should do it.

1.5   author

My internet email address is (1993 August)
pilgrim@daimi.aau.dk

and my mail address

Jakob Gårdsted,
Aldersrovej 11,5
82OO Århus N,
Denmark/Europe

This program is strictly PD. I
reserve the copyright or what
ever it is called, but you may
use the source for whatever you'd

want to in your own projects. What
you may not do, is make profit by
distributing my program. I don't
know why everybody keeps writing
"I cannot be held responsible for
what this program does to you or
your dog", but now I've said it too.

If there is any problem with the
program, dependant or independant
of machine configuration, I would like
to hear about it (to fix it).

If there is any problem with the
program, dependant or independant
of machine configuration, and you
know how to fix it, I would like
to hear about it too...

And, lastly, if you have any suggestions,
go ahead and mail them too. You are
almost guaranteed to get a reply, although
it may be along the lines of

"Naah, I'm too lazy to want to do that..."

1.6 purpose

Why I made this ? Because I have not
come across a PD program, that does
this. The only one I've seen, flickered
no matter what you did. This one doesn't
flicker, at least if you run it from the
boot shell...

1.7 genlock

I have not tried this program with a
genlock. I want it to work with genlock,
so if it doesn't, tell me. And if
you know what I should do to make it work
with genlock, tell me too. I have the
RKM Libraries 2.0, so I should be able
to read up on it.

    PS: By working with genlock I mean
"The background colour should be
replaced with the video picture".

1.8 contents

```
This archive (Aftertitles.lha) contains

ats       - the program
demo.ilbm - a picture to make people
            quickly realise what this program
            does, and if they want it.
ats.guide - this very text you're reading.
            Containing the documentation for
            the program.
ats.c     - the C source.
DMakefile - a makefile for it.
Readme    - a short description.
```