Malibu 1.4

Scala Emulation For Hollywood

Andreas Falkenhahn

Table of Contents

1 General information

1.1 Introduction

Malibu is a plugin for Hollywood which contains an import filter for Scala presentations made with the classic Scala software for Amiga computers. As soon as Malibu is installed, Hollywood will suddenly be able to show those Scala presentations or compile them.

With Malibu it is now possible to run those Scala presentations without the actual classic Amiga hardware. Because the whole emulation runs on top of Hollywood's multimedia application layer you can run Scala presentations on every platform supported by Hollywood. In many cases the quality of the pictures that Malibu displays is much better than the original Scala quality because Scala often had to remap many pictures with very different colors to a 8-bit display which resulted in a massive loss of color information. With Malibu, however, all graphics are shown in true colour.

Malibu emulates nearly the whole Scala command set and supports all Scala versions from 1.0 up to Scala InfoChannel 500 which was the last Amiga version of Scala. Emulation is done by the El Capitan microkernel which tries to achieve the best performance possible on the system used. For example, the next page is already calculated while the current page is on the display. Complex transition effects can also be pre-calculated and then displayed smoothly.

Scala is a program which helped the Amiga to become a real Multimedia wonder. There were no products for a long long time that could beat an Amiga with Scala. Therefore Scala was widely used in all different production fields, from movies and television to big shows. Everyone who has worked with Scala knows what powers it has. But this power is only possible because Scala is very hard-coded to the Amiga's custom chips which leads to the problem that it does not run on modern Amiga systems or other platforms. For many years all sorts of users complained that there was no Scala that runs RTG compatible. This time is over now. Malibu revives Scala in a completely system-friendly way.

Here is a quick overview of the features of Malibu:

Graphics:

– Impressive Scala transitions (e.g. Superimpose, Roll on, Flow, Push, Diagonal)
– Support for Scala text styles outline, shadow, underline, charspacing
– Scala forms: Ellipses, rectangles, lines
– Emulation of backgrounds with raster type
– Complete brush support (incl. crop & resize)
– Animations are supported
– Transition effects for all Scala objects
– Support for styles like bevel & background
– Runs completely in 24-bit mode
– Color fonts are supported
– Smooth scrolling for Scala fly-ons
– Emulation of all graphics attributes

– Brush transparency is fully supported
– Objects can be removed from screen with transition effects
– All palette settings are respected and correctly converted

Sound:

– Support for sound modules and samples
– Audio output through AHI incl. full mixing
– Sound fades are supported
– Sound implementation is retargetable for use on Pegasos, Amithlon etc.

Environment:

– All Scala versions up to InfoChannel 500 are supported
– Margin settings and tabulators are respected
– Scala variables can be used
– Buttons and events are emulated
– Almost complete emulation of Scala command set
– Scala presentations can be compiled to executables
– Emulation can be configured in detail
– Completely platform-independent implementation
– Scala presentations can be saved as Hollywood scripts
– Absolutely system-friendly

1.2 Copyright

Malibu is © Copyright 2002-2017 by Andreas Falkenhahn (in the following referred to as "the author"). All rights reserved.

The program is provided "as-is" and the author cannot be made responsible of any possible harm done by it. You are using this program absolutely at your own risk. No warranties are implied or given by the author.

This plugin may be freely distributed as long as the following three conditions are met:

1. No modifications must be made to the plugin.
2. It is not allowed to sell this plugin.
3. If you want to put this plugin on a coverdisc, you need to ask for permission first.

Amiga is a registered trademark of Amiga, Inc.

All other trademarks belong to their respective owners.

DISCLAIMER: THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDER AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE EN-TIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS

WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSE-QUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BE-ING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PAR-TIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PRO-GRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1.3 Requirements

− Hollywood 5.0 or better

1.4 Installation

Installing Malibu is straightforward and simple: Just copy the file `malibu.hwp` for the platform to Hollywood's plugins directory. On all systems except on AmigaOS and compatibles, plugins must be stored in a directory named `Plugins` that is in the same directory as the main Hollywood program. On AmigaOS and compatible systems, plugins must be installed to `LIBS:Hollywood` instead. On Mac OS X, the `Plugins` directory must be inside the `Resources` directory of the application bundle, i.e. inside the `HollywoodInterpreter.app/Contents/Resources` directory. Note that `HollywoodInterpreter.app` is stored inside the `Hollywood.app` application bundle itself, namely in `Hollywood.app/Contents/Resources`.

On Windows you should also copy the file `Malibu.chm` to the `Docs` directory of your Hollywood installation. Then you will be able to get online help from the IDE.

On Linux and Mac OS copy the `Malibu` directory that is inside the `Docs` directory of the Malibu distribution archive to the `Docs` directory of your Hollywood installation. Note that on Mac OS the `Docs` directory is within the `Hollywood.app` application bundle, i.e. in `Hollywood.app/Contents/Resources/Docs`.

2 Usage

2.1 Starting Scala scripts

Once the Malibu plugin has been installed correctly, Hollywood will automatically be able to display your Scala scripts. Therefore you can start your Scala scripts now in the same way you start your Hollywood scripts, e.g. by using the Hollywood GUI or by using the CLI.

Please consult your Hollywood manual for more information on how to start scripts and which arguments are possible.

2.2 Dealing with assigns

Typically, Scala scripts will use path names containing assigns to link to external files. On Amiga systems, this is not a problem because you can just set up those assigns and it will work. On non-Amiga systems, however, there is no concept of assigns. Thus, if a Scala script tries to open files via Amiga assigns (e.g. `SCALA:Scripts/Background.iff`), Malibu will open a requester prompting you to select the path on your hard drive which should correspond to the Amiga assign so that Malibu can find all files. You can choose to have these assigns stored permanently or just for the current session.

If you choose to store assigns permanently, they will be saved to `malibu.ini`. You can also manually add assigns to `malibu.ini` using the following syntax:

```
Assign 1="SCALA:E:/AmigaApps/Scala"
Assign 2="Obligement12:E:/ScalaMags/Obligement12"
```

You can add as many assigns as you want but they need to start from 1 and there mustn't be any gaps in the sequence. Each assign specification consists of the assign name ending on a colon followed by the path that should be used for this assign.

2.3 Setting up font paths

On AmigaOS and compatibles, Malibu will look in `FONTS:` for all fonts used by the script. Since this is not possible on non-Amiga systems, you have to manually specify font lookup paths. This is done by editing the file `malibu.ini` which has to be stored in the same location as the `malibu.hwp` main plugin executable.

If the file `malibu.ini` isn't there yet, then just create it. It's just a simple text file. To add font search paths to Malibu, insert the following line into `malibu.ini`:

```
Font search path 1="C:/Amiga files/Scala/Fonts"
```

The line above will add the path "C:/Amiga files/Scala/Fonts" to Malibu's font search paths. This means that Malibu will always check this path when looking for fonts. You can add as many font search paths as you want. You just have to increase the counter next to the key name in the `malibu.ini` file. To add another two search paths, just add these two lines:

```
Font search path 2="D:/Amiga files/Scala/Fonts"
Font search path 3="E:/Amiga files/Scala/Fonts"
```

2.4 Emulation settings

On AmigaOS and compatibles, Malibu comes with a settings program that allows you to fine-tune the Scala emulation. You can start the settings program from the Plugins menu in the Hollywood GUI or from Workbench. The settings program will write your settings to the file `malibu.ini` which is stored in the same directory as the main `malibu.hwp` executable. You can also modify `malibu.ini` directly - see below for instructions.

On all other platforms there is no dedicated settings program. Instead, you have to modify `malibu.ini` directly. The file `malibu.ini` must be stored in the same directory as the main `malibu.hwp` executable. If it isn't there yet, just create it. Below all options that can be set in the settings program on AmigaOS and compatibles are described. At the end of the description of each option, you will find the key to set in the `malibu.ini` settings file so that you can enable or disable certain options by manually editing `malibu.ini`. For example, to enable script looping add the following line to `malibu.ini`:

```
Loop script=True
```

The following things can be configured:

Emulation options:

– Loop script: If you check this box, every Scala script you start will be looped when it reaches the end. The key for this setting is **Loop script**. Defaults to `True`.

– Emulate pause commands at page end: Most Scala scripts have pause commands at the end of each page. If you check this box, they will be emulated. Unless you have a very fast Amiga, you do not need to check this box because Malibu will start preparing the next page automatically when it reaches the end of a page. And this takes mostly longer than the original pause command. So you should only check this box if you have a very fast Amiga and scripts run too fast. The key for this option is **Emulate pause end**. Defaults to `True`.

– Emulate pause commands during page: If you check this box, pause commands that occur during a page is displayed, will be emulated. It is recommended that you check this box. The key for this option is **Emulate pause page**. Defaults to `True`.

– Emulate pause accurately: If this option is activated, Malibu will try to emulate Scala's pause commands as accurately as possible. Enabling this option can lead to timing problems because Hollywood's transition effects aren't exactly as long as Scala's wipes. That's why it is turned off by default. The key for this option is **Accurate pause**. Defaults to `False`.

Animation options:

– Play animations: If you check this box, animations will be played. Please note that you should also select "Play animations from disk" because animations require a lot of memory especially because they all have to be converted to 24-bit, e.g. a 640x480 animation in 16 colors with 100 frames would take up around 90 megabyte if loaded into memory (!!). Therefore you should really also select "Play animations from disk". The key for this option is **Play anims**. Defaults to `True`.

– Play animations from disk: If you check this box, Malibu will play all animations directly from disk. This is a bit slower but saves a lot of memory. Recommended. The key for this option is **Play anims from disk**. Defaults to `True`.

Transition effect options:

– Picture transition speed: This combo box allows you to specify a speed setting that will be used for all picture transitions. The key for this option is **BGPic FX speed**. A value of 0 means slow, 1 means normal, and 2 means fast. Defaults to 1.

– Default picture transition: Hollywood does not support all Scala transition effects. The transition you choose here will be displayed when Malibu cannot find an equal Hollywood transition for the required Scala transition. The default setting for this is 'random transition'. The key for this option is **Default BGPic FX**. It must be set to the integer value of a Hollywood transition effect increased by one. To use a random effect, set it to 0 which is also the default.

– Object transition speed: This combo box allows you to specify a speed setting that will be used for all object transitions. The key for this option is **Object FX speed**. A value of 0 means slow, 1 means normal, and 2 means fast. Defaults to 1.

– Default object transition: This is the same as 'default picture transition' except that the transition effect you specify here will be used for object transitions. The key for this option is **Default object FX**. It must be set to the integer value of a Hollywood transition effect increased by one. To use a random effect, set it to 0 which is also the default.

Export options:

– Save script to: If you enter the name of a filename here, Malibu will automatically save a copy of the Scala script that has been converted to a Hollywood script here. This allows you to make some manual modifications in the script. The key for this option is **Save script**. Defaults to the empty string which means that Malibu shouldn't save the Hollywood script it generated from the input Scala script.

Miscellaneous options:

– Enable quit with ESC: If you check this box, you will be able to quit any time by pressing the escape key (useful in full screen mode when the window has no close gadget). The key for this option is **Enable escape quit**. Defaults to `True`.

– Show busy pointer when working: If you check this box, Malibu will indicate its busy state by showing the busy pointer. On slower systems it can be useful to check this box. The key for this option is **Show busy pointer**. Defaults to `False`.

– Hide pointer all the time: Check this box if you do not want the pointer visible at all. Useful for presentations. The key for this option is **Hide pointer**. Defaults to `False`.

– Allow relative paths: Scala always uses absolute path definitions for external data. This has the drawback, that scripts do not run on other systems without modifications because every file name is addressed absolutely, e.g. `dh0:Apps/Scala/Scripts/Demo.iff`. If you check this box, Malibu will first look if the file is present in the current directory and load it from there. Otherwise it will try the full path. The key for this option is **Try relative path**. Defaults to `True`.

– Substitute Intellifonts: If this option is selected, Malibu will replace all Intellifonts with a default font. This can be useful if you're running Malibu on a system which doesn't support Compugraphic Intellifonts (such as Windows, Mac OS X, or Linux). The key

for this option is **Replace intelli fonts**. It defaults to `True` on non-Amiga systems and to `False` on Amiga systems.

– Never fail because of missing fonts: If you select this option, Malibu will automatically substitute missing fonts. It's not recommended to set this option because if you do you won't be notified about missing fonts and so you don't have a chance to find and install those fonts. So this option is really only useful for temporary testing purposes. The key for this option is **Never fail on fonts**. Defaults to `False`.

– Emulate font antialiasing: Select this option to enable font antialiasing. Note that this isn't a true vector-based antialiasing but just an approximation based on blurring so it doesn't look too good. That's why it is disabled by default. The key for this option is **Emulate antialias**. Defaults to `False`.

2.5 Emulation information

This chapter contains some information about the Scala lingo commands that are recognized and emulated by Malibu.

AACOLOR:

– not emulated
– anti-aliasing is not yet supported by Hollywood

ANIM:

– 100% emulated

ATTRIBUTES:

– 3d: not emulated
– antialias: not emulated
– bevel: emulated
– bold: emulated
– center: emulated
– edge: emulated
– inactive: not emulated
– italics: emulated
– jam: emulated
– kerning: not emulated
– left: emulated
– none: emulated
– remap: emulated
– right: emulated
– underline: emulated

BLANK:

– 100% emulated

BOX:
- 100% emulated

BRUSH:
- 100% emulated including crop, scale and transparency
- old syntax (Scala 1.0) supported

BUTTON:
- 100% emulated

COLOR:
- 100% emulated

CONTINUE:
- not emulated
- ARexx is not supported by Hollywood

CYCLE:
- not emulated
- palette cycles are not possible in 24-bit screen modes

END:
- 100% emulated

ELLIPSE:
- 100% emulated

EVENT:
- 100% emulated

EX:
- not emulated
- external modules are hardware dependent

EXECUTE:
- CLI execution emulated
- ARexx and WB execution currently not supported

FKEYS:
- 100% emulated

FONT:
- 100% emulated

GETVAR:
- not emulated
- ARexx is not supported by Hollywood

GOTO:
- 100% emulated

GRID:
- not emulated

IF:
- 100% emulated

INTERACTIVE:
- 100% emulated

JOYSTICK:
- not emulated

LINE:
- 100% emulated

MARGINS:
- 100% emulated

MARK:
- box: emulated
- complement: emulated
- fill: emulated
- none: emulated
- replace: emulated

MARKSOUND:
- 100% emulated

MOUSE:
- 100% emulated

NUMKEYS:
- 100% emulated

PALETTE:
- 12bit definitions: emulated
- 24bit definitions: emulated

PAUSE:
– 100% emulated
– both syntax versions of PAUSE are recognized (the old one from Scala 1.0 to MM200 and the new one from MM300 and up)

PICTURE:
– 100% emulated

POINTER:
– not emulated
– hiding the mouse pointer on Workbench screen might confuse the user

QUIT:
– not emulated
– ARexx is not supported by Hollywood

RESOLUTION:
– 100% emulated

RETURN:
– 100% emulated

SCRIPT:
– 100% emulated

SELECT:
– box: emulated
– complement: emulated
– fill: emulated
– none: emulated
– replace: emulated

SELECTSOUND:
– 100% emulated

SET:
– 100% emulated

SETVAR:
– not emulated
– ARexx is not supported by Hollywood

SHOW:
– not emulated
– ARexx is not supported by Hollywood

SOUND:

- play: 100% emulated
- stop: 100% emulated
- volume: 100% emulated
- wait: 100% emulated
- delay: 100% emulated
- fade: 100% emulated
- fadein: 100% emulated
- fadeout: not emulated
- pan: not emulated
- period: 100% emulated

STYLE:

- transparency: emulated
- shadowdir: emulated
- shadowlen: emulated
- 3dlen: not emulated
- boldsize: not emulated
- underpos: emulated
- undersize: emulated
- underair: emulated
- italics: emulated
- spacing: emulated
- linespacing: emulated
- antialiaslevel: emulated
- remap: emulated
- rastertype: emulated
- linethickness: emulated
- bevellen: emulated

TABS:

- 100% emulated

TEXT:

- 100% emulated
- including tabs, variables and escape sequences
- old syntax (Scala 1.0) supported

TEXTOUT:

- 100% emulated

TEXTWIPE:

- emulated
- most important Scala effects use similar Hollywood effects
- all other effects use random transitions

WIPE:

- emulated
- most important Scala effects use similar Hollywood effects
- all other effects use random transitions
- old syntax (Scala 1.0) supported here

WORDWRAP:

- 100% emulated

3 Miscellaneous

3.1 F.A.Q.

Here are the answers to some frequently asked questions. Be sure to read this information before sending e-mails.

Q: I get 'Out of memory' for every script I start!?

A: Select 'Play animations from disk' in the preferences. If you still get 'Out of memory' errors, there is no other possibility than to upgrade your RAM.

Q: Why are animations so slow?

A: Every frame has to be converted from planar format to true colour. This takes some time and therefore playback can be slow but there is no other way to do this. Try to view an animation with MultiView; it will be very slow too.

Q: Why does it take so long to load animations?

A: This is the same problem as the previous question. Hollywood has to convert every frame and if it uses many colors and a high resolution then it will take some time.

Q: Malibu cannot load my animations.

A: Check if there is a datatype for the animation. Sometimes Scala uses a special animation format (ANIM8) which is not supported by the animation.datatype and therefore cannot be loaded by Malibu.

Q: Why does anti-aliasing look so bad?

A: Scala offers anti-aliasing on bitmap and intellifonts using some clever dedicated algorithms. Hollywood, however, only supports antialiasing for TrueType fonts. Thus, Malibu emulates Scala's antialiasing using edge blurring but this of course doesn't look as good as Scala's dedicated algorithms.

Q: Why don't you emulate the Scala schedule feature?

A: This would be too difficult currently because Malibu needs much more time for preparing each page than Scala needed. Therefore a schedule support would not make much sense here. But it is planned for PPC versions where page preparation is fast enough to support it.

Q: Is ARexx support planned?

A: Yes, I'm working on it. However, Hollywood needs to offer commands for this first before I can implement it.

Q: What about Scala EX's?

A: This is not emulated and will not be emulated in the future because it is almost impossible. Scala EX rely heavily on external hardware and it is not clear to me how all the stuff works because I have none of that hardware. You will have to live without it.

Q: What about the cool cycle effect of Scala?

A: This is also almost impossible to emulate. You would need a very fast processor to calculate all the different cycles in 24-bit. Color cycles are tricks that can only be done very fast on palette-based displays. In 24-bit mode you would have to modify every pixel. 68k processors are far away from doing that at an acceptable speed.

3.2 History

Please see the file `history.txt` for a complete change log of Malibu.

3.3 To Do

– add support for 3D style
– add support for linked transitions
– add better support for font anti-alias
– 100% correct support for animations (support objects on all frames!)
– implement more Scala transitions (in Hollywood)
– implement support for checking events during pause states (ScalaVideo.script)
– implement schedule support
– ARexx support (maybe)
– support for sheared intelli fonts
– improve compatibility

Please drop me a mail if you have some nice ideas what shall be implemented in Malibu.

3.4 Credits

Malibu was written by Andreas Falkenhahn between December 2002 and May 2003. It runs as a plugin for the Multimedia Application Layer Hollywood and converts Scala scripts on-the-fly to Hollywood scripts.

Malibu is a high quality software product for Amiga computers produced by Airsoft Softwair.

Hollywood was also written by Andreas Falkenhahn.

Thanks must go to Jean-Yves Auger of Pixel Art.

Malibu was developed on my Amiga 1200 equipped with a Phase5 Blizzard PPC 603e 200mhz with a 68040 CPU, a Phase5 BVision PPC graphics board and 80 Megabyte RAM.

If you need to contact me, you can either send an e-mail to `andreas@airsoftsoftwair.de` or use the contact form on `http://www.hollywood-mal.com`.