

# AKFAvatar

---

a fancy user interface

Andreas K. Foerster

---

This is the documentation for AKFAvatar (version 0.18.0, 2009-11-08).

AKFAvatar is a fancy user interface for textconsole-programs, a text-viewer, a scripting language for making demos, and a library for writing programs in C or Pascal.

Homepage: <http://akfavatar.nongnu.org/>

Copyright © 2007, 2008, 2009 Andreas K. Foerster, <http://akfoerster.de/>

Copying and distribution of this documentation, with or without modification, are permitted in any medium without royalty provided the copyright notice and this notice are preserved. This documentation is offered as-is, without any warranty.

# Table of Contents

<b>1</b>	<b>Overview of AKFAvatar</b> .....	<b>1</b>
1.1	General usage (keys) .....	2
<b>2</b>	<b>How to install AKFAvatar</b> .....	<b>3</b>
2.1	Requirements .....	3
2.2	Installation .....	3
<b>3</b>	<b>Using the tool avatarsay</b> .....	<b>6</b>
3.1	Using avatarsay as fancy text-terminal .....	6
3.1.1	frontend for text-console programs .....	6
3.1.2	the terminal-type .....	6
3.1.3	extensions .....	7
3.2	Using avatarsay as fancy text-viewer .....	7
3.3	How to turn your text into a program .....	8
3.4	Using a different avatar image .....	8
3.4.1	Transparent avatar background .....	8
3.5	Invoking avatarsay .....	9
3.5.1	Environment variables .....	10
3.5.2	Configuration file .....	10
3.6	Commands for avatarsay .....	10
3.7	Pipes .....	13
3.8	Archive-files .....	14
<b>4</b>	<b>Using it under GNOME</b> .....	<b>15</b>
4.1	What you need to use <code>gnome-akfavatar</code> .....	15
4.2	The main-menu of <code>gnome-akfavatar</code> .....	15
<b>5</b>	<b>About file formats and values</b> .....	<b>17</b>
5.1	Colors .....	17
5.2	Images .....	17
5.3	Audio files .....	17
<b>6</b>	<b>Programming with AKFAvatar</b> .....	<b>18</b>
6.1	How to program with AKFAvatar in Pascal .....	18
6.1.1	Simple use cases .....	18
6.1.2	Installing .....	18
6.1.3	Advanced use .....	18
<b>Appendix A</b>	<b>Pascal reference</b> .....	<b>21</b>
<b>Appendix B</b>	<b>C reference</b> .....	<b>28</b>
<b>Appendix C</b>	<b>Names of colors</b> .....	<b>47</b>
<b>Index</b> .....		<b>50</b>

# 1 Overview of AKFAvatar

AKFAvatar is a graphical program and a library, where an avatar appears on the screen, and tells things to the user written in a balloon. There can also be recorded audio files, so that the user even can hear, what it is saying.

AKFAvatar can be used in several different ways.

1. You can use the program `avatarsay` as a text-terminal, just like `xterm`, but being much more fun. In contrast to `xterm` it might even run without an x-server. (Under Windows this function is not available.)
2. You can use it as user interface for your textmode programs. (Under Windows this function is not available.)
3. The command `avatarsay` can be used as a fancy text-viewer, more or less like `more` or `less`.
4. But the command `avatarsay` can do much more than that. It can be used as a simple scripting language for making demos that you can show for example at an information booth or in a shop's window.

Don't worry, it sounds more complicated, than it is. I should rather say, you can spice up your texts with occasional commands.

5. Then there is the library `libakfavatar`, which you can use from programming languages. Currently Free Pascal, GNU-Pascal and C-compatible languages are supported. Especially the Pascal language is suited for beginners to learn how to program. The library is simple to use, like writing command-line programs — but it's much more fun!

## 1.1 General usage (keys)

When an AKFAvatar program runs in a window, you can of course use the close button of your window manager for stopping the program. You can also always stop the program by pressing the key combination `ALT+Q`, for example when it runs in fullscreen mode. Often you can also stop the program by pressing the `ESC` key. But this key might be reserved for other things, for example in the terminal-mode of `avatarsay`.

You can press the `PAUSE` key at any time to pause the running process. Press any other key to continue again.

On some systems you can use the key combinations `ALT+ENTER` or `CRTL+ALT+F` to toggle between the fullscreen and the window mode. Often you can also simply use the key `F11`, but this key might be reserved for other things, for example in the terminal-mode of `avatarsay`.

## 2 How to install AKFAvatar

Although there are binary packages available, you should install AKFAvatar from the source code package to use the software to its full capacity.

This chapter describes the installation for POSIX compatible operating systems, mainly for GNU/Linux systems. There are some hints for other systems at the end of this chapter.

For short: `./configure && make && make install`

### 2.1 Requirements

#### Needed

- SDL-1.2.x (version 1.2.11 or higher recommended)

The 1.3 series is *not* backward compatible!

<http://libsdl.org/download-1.2.php>

You need the “Runtime Libraries” as well as the “Development Libraries”!

Of course SDL needs a graphical environment to run in. For example the X-Window-System or a Linux framebuffer-device. . .

- An ANSI-C Compiler (gcc or others)

<http://gcc.gnu.org/>

- Entries for “linux” and “linux-m” in the terminal database.

This is also needed on systems, which don’t use the kernel Linux, but not on Windows. On Debian or derived distributions, you must have the package “ncurses-term” installed.

#### Recommended

- SDL\_image

[http://www.libsdl.org/projects/SDL\\_image/](http://www.libsdl.org/projects/SDL_image/)

There the Runtime Libraries are sufficient

SDL\_image in turn needs other libraries: libz, libpng, libjpeg, libtiff

X-Pixmaps (XPM), X-Bitmaps (XBM) and uncompressed BMP images are always supported. With SDL\_image you can use lots of other image formats. SDL\_image can be installed after the installation of this package.

- iconv

on a lot of systems that is already included

#### Optional

- GNU-Pascal or Free Pascal

<http://www.gnu-pascal.de/>

<http://www.freepascal.org/>

### 2.2 Installation

#### Compiling

Run `./configure`. If that succeeds run `make` to create the binaries. There are two variants of avatarsay being compiled: 1. “avatarsay” is a special compilation, which can be used without installing it. 2. “avatarsay-d” will only work, when the library is installed. This is the variant that gets installed.

## Compiling — special cases

On some systems you need to use the parameter ‘`--with-iconv`’ with `./configure` to get full iconv support (iconv is a charset-encoding converter). This parameter is not needed when SDL is already configured to use an external iconv. But in some cases it is not, although an external iconv is available (because maybe there is more than one implementation). This software (AKFAvatar) does an educated guess to get the internal encoding for `wchar_t` right. In some cases it might not work. You might then provide that information with this option, like this ‘`--with-iconv=UCS-4LE`’. On many systems you can get a list with the command `iconv -l`. If “WCHAR\_T” or “wchar\_t” is on the list, use that.

If you know that `SDL_image` is always installed, then you can provide the parameter ‘`--enable-link-sdl-image`’ to directly link to it, instead of loading it at runtime.

To support old versions of SDL on a target machine, it might be necessary to use the parameter ‘`--with-oldsdl`’. Note that this is not needed if you just want to run the program on the same machine on which you compile it.

If you want to use it on devices with a small display (eg. netbooks), use the parameter ‘`--enable-size=vga`’ with `configure`. The value `vga` means a size of 640\*480 pixels.

## Testing

The program `avatarsay` is a text reader and a simple scripting language. Try to view this text with that program: ‘`./avatarsay --pager INSTALL`’.

You can stop `avatarsay` any time with the ESC key. On some systems you can toggle between window and fullscreen mode using the key F11. Another useful key is the PAUSE key.

There are some example scripts in the package. Try to run ‘`./fsdemo-en.avt`’ or whatever language you prefer. Open ‘`fsdemo-en.avt`’ in a text-editor to see how to write such scripts. . .

You can also use it as a fancy manpage reader. Try ‘`avtman man`’. . .

## Installing

On GNU/Linux systems first make sure ‘`/usr/local/lib`’ is mentioned in the file ‘`/etc/ld.so.conf`’; either directly or indirectly. Also make sure that ‘`/usr/local/bin`’ is in your `PATH` environment variable.

Now get root privileges and run ‘`make install`’ to install it in ‘`/usr/local`’. If you are low on disk space, you can instead use ‘`make install-strip`’. This installs binaries stripped from debugging information.

If you want to uninstall it later, you can use the command ‘`make uninstall`’.

## Special targets

With the command ‘`make example`’ you can compile the program ‘`example.c`’. The file ‘`example.c`’ is an example, which you can use to start your own programs.

## Trouble-shooting

- Problem: the system can’t find the library ‘`libakfavatar.so`’

Solution: first try to add ‘`/usr/local/lib`’ or wherever you installed it to the environment variable `LD_LIBRARY_PATH`: ‘`export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib`’

As a more permanent solution make sure ‘`/usr/local/lib`’ is directly or indirectly included in the file ‘`/etc/ld.so.conf`’. Then run the command `ldconfig` with root privileges.

- Problem: some characters are not shown correctly with `avatarsay`

Solution: There are different character sets. Try to use the parameters ‘`--utf-8`’ or ‘`--latin1`’.

## Other systems

### BSD

AKFAvatar is known to be compilable with FreeBSD 6.2. To get iconv support, install `libiconv` and use `./configure --with-iconv`.

### Windows

Windows is just partly supported. I provide binaries which are build with a crosscompiler. I haven't compiled them directly under Windows yet, so you're on your own here. I'm sorry.

For systems like these SDL supports a fallback implementation for iconv. The charset Windows-1252 (often called "ANSI" but it has nothing to do with ANSI) is not fully supported. That charset is partly compatible to the ISO-8859-1 standard, which is supported. Some characters are missing then. If you need these characters, save your text as UTF-8, which is also supported.

## 3 Using the tool `avatarsay`

This chapter explains the various uses of the tool `avatarsay`.

### 3.1 Using `avatarsay` as fancy text-terminal

The program `avatarsay` can be used as a fancy text-terminal and as a frontend for textconsole programs on POSIX-compatible systems (not for Windows).

From the main menu you can choose “terminal-mode”, or you can start `avatarsay` with the option ‘`--terminal`’, or ‘`-t`’ for short. This gives you a terminal session with the default shell for your system-account. If you don’t like colors being used in the balloon, you can use the option ‘`--nocolor`’, or ‘`-b`’ for short.

If you want to use a different shell or starting directory, you can set the environment variables `SHELL` and `HOME` to different values. By the way, the program can get the correct values also when these variables are not set.

#### 3.1.1 frontend for text-console programs

You can use `avatarsay` as a frontend for text-console programs. Use the option ‘`--execute`’, or ‘`-x`’ for short, followed by the name of the program to execute. You can of course also combine this with the option ‘`--nocolor`’. Options after the programs name are passed to the executed program. That means, the order of the options is important.

For example to log in into another machine, using `ssh`:

```
avatarsay --execute ssh example.net
```

To write an e-mail with the program `mutt`:

```
avatarsay -bx mutt pal@example.net
```

Write the mail with a mail program on the remote machine:

```
avatarsay -bx ssh -t example.net mutt pal
```

Note: when you use `ssh` this way, you have to use the option ‘`-t`’ for `ssh`.

Surf the web with `lynx`:

```
avatarsay -bx lynx
```

As you can see, you can use it to actually do very weird things with it.

#### 3.1.2 the terminal-type

The program `avatarsay` simulates the text-console of the kernel Linux. So the environment variable `TERM` is set to either `linux` or `linux-m`.

Because this choice might lead to some confusion, I try to explain, what this does *not* mean:

- It does *not* mean, that the code is limited to systems, that use Linux as kernel. It was successfully tested on FreeBSD and should run fairly well on other POSIX-compatible systems also.
- It does *not* mean, that the variable `TERM` should be changed on other systems. Leave it as it is. But the system should have entries for `linux` and `linux-m` in its terminal database(s).
- It does *not* mean, that I have used code from Linux. I just read the manpage `console_codes` and the Terminfo and Termcap entries and some other documents to get information.
- It does *not* mean, that the terminal is not compliant to standards. In fact, the console emulation of the kernel Linux is very compatible to ANSI X3.64 (ECMA-48). But there are things missing and some things, like key-codes are extra. So the value `linux` is actually the right one.

### 3.1.3 extensions

The terminal-emulation of `avatarsay` supports extensions, which are not in ANSI X3.64 (ECMA-48) and which are not supported by the Linux terminal emulation.

To use these extensions in your own program, it is recommended first to check for the existence of the environment variable `AKFAVTTERM`.

`CSI ? 56 h`

switches the slowprint-mode on (deprecated: use APC)

`CSI ? 56 l`

switches the slowprint-mode off (deprecated: use APC)

`CSI 8 ; height ; width t`

set the height and width (in characters) of the balloon. The value 0 sets the maximum value. (This is actually compatible to the original `xterm`)

APC *command* `ST`

sends a command to `avatarsay`. See [Section 3.6 \[Commands for avatarsay\]](#), page 10.

further extensions are planned.

*Explanations:*

*ESC* (Escape)

is the control code `1Bhex`

*CSI* (Control Sequence Introducer)

can be represented with the control sequence `ESC` followed by `[` or in some charsets with the single control code `9Bhex`.

*APC* (Application Program Command)

can be represented with the control sequence `ESC` followed by `_`, or in some charsets with the single control code `9Fhex`.

*ST* (String Terminator)

can be represented with the control sequence `ESC` followed by `\`, or in some charsets with the single control code `9Chex`.

The variants with `ESC` should be preferred. Especially in UTF-8 there is no advantage in the single control codes, since they also have to be encoded with two bytes.

## 3.2 Using `avatarsay` as fancy text-viewer.

The program `avatarsay` can be used as a fancy text-viewer. If you have a textfile, say `mytext`, then you can read it with the command `avatarsay mytext`. Try it! Now! ;-)

That is easy, isn't it? Well, if your text file is not encoded in the right encoding for your system, you might have trouble with non-ASCII characters. There are different encodings. The most commonly used encodings are ISO-8859-1 (also known as Latin-1) and UTF-8. So if you have trouble with non-English characters, try to use the parameter `--encoding=UTF-8` or `--encoding=ISO-8859-1` — like this: `avatarsay --encoding=ISO-8859-1 mytext`.

Now, sometimes you might not want to show the whole in one continuous stream. So you can structure your text by including a stripline. A stripline is a line like this: `-----`. The line must start at the very first column and there must be at least three successive hyphens (U+002D). Of course you can use more than three ones. When `avatarsay` sees a stripline, it waits a moment and then it starts a new page.

### 3.3 How to turn your text into a program

You don't always have to use the command `avatarsay` for viewing your texts. You can turn your text itself into a “program”. How that works depends on your system.

On a POSIX compatible system, such as the GNU system you can put a special line at the beginning of your file. The line looks like this: `#!/usr/local/avatarsay`. Of course you have to make sure that you use the correct pathname. This has to be the very first line. You can put one or more empty lines after this line — they are ignored. Then you have to set the executable bit of your text file. That goes like this: `chmod +x mytext`. Then your text is executable. It accepts almost all the options that `avatarsay` accepts.

On systems like Windows or ReactOS this line doesn't work — but it doesn't harm either. On those systems you have to use a different trick. You could rename your file such, that it has a special extension; for example `.avt`. Now you have to configure your system, that the file extension `.avt` gets associated with the program `avatarsay.exe`. Then you can double-click on your “text” and it is executed.

### 3.4 Using a different avatar image

Up to now you have always seen the same avatar — Well, that is just the default avatar. You can exchange it.

There are different methods to exchange the avatar for `avatarsay`:

You can change it globally for any use of `avatarsay`. This can be done either with an environment variable or you can use a system-wide configuration file.

Well, first you might want to try the environment variable. That is simple. The variable is called `AVATARIMAGE`. So with the GNU `bash` you can set this variable like this: `export AVATARIMAGE=/usr/local/share/pixmaps/myavatar.xpm`. **Important:** You should always use the full path!

If you want to use a configuration file, create a file named `/etc/avatarsay` and put something like in this example into it:

```
AVATARIMAGE=/usr/local/share/pixmaps/myavatar.xpm
```

If you write a text you want to be shown by `avatarsay`, you can change the avatar-image on a text-by-text basis, using a command in the text-file. This is explained in [Section 3.6 \[Commands for avatarsay\]](#), page 10.

Which file-formats AKFAvatar supports depends on which libraries you have installed (see [Chapter 5 \[Formats\]](#), page 17).

#### 3.4.1 Transparent avatar background

The avatar-image should have a transparent background of course. Well, a lot of image formats don't support transparency at all. Therefore AKFAvatar has a trick. If the avatar-image has no transparency, then it looks up the first color in the image, that is the color in the upper left corner, and defines this color to be transparent. So when you prepare an image as replacement for the avatar, make sure the upper left corner is “empty” and make sure to choose a background color, which doesn't appear in the part of the image which is meant to be visible. Also make sure, that the background is “flat” with only one single color without any variations. Because of this requirement the JPEG format is not appropriate, you cannot get a really flat background in that format.

So you should use an image format which supports transparency. AKFAvatar will not interfere with it then. Note however that the trick explained above is always used when the image *has* no transparency, independent from the question, whether the image format *could* have transparency.

### 3.5 Invoking `avatarsay`

The format for running `avatarsay` is:

```
avatarsay [options] [textfiles]
avatarsay [options] --execute program [program options]
```

If `textfiles` is `'-'` then the input is read from `'stdin'` and the program doesn't loop.

The program `avatarsay` supports the following options:

```
--help
-h          show a short summary about the invocation of avatarsay

--version
-v          show the version of the command

--terminal
-t          terminal mode, ie. run a shell in the balloon

--execute
-x          execute program in the balloon
           A program name must be given. Options after the program name are options for
           the executed program.

--nocolor
-b          no color for executed programs and the terminal mode

--window
-w          try to run the program in a window (default)

--fullscreen
-f          try to run the program in fullscreen mode

--fullfullscreen
-F          like '-f', but use the current display size
           This option is also useful when there is only a fullscreen mode, but switching the
           screen resolution doesn't work; for example with the VESA framebuffer device of the
           kernel Linux
           This option is only supported with SDL version 1.2.10 or newer.

--encoding=name
           the input data is encoded in the encoding name
           It depends on your systems iconv implementation, which encodings are supported.
           On some systems you can get a list with the command iconv -l.

--latin1
-l          the input data is encoded in Latin-1

--utf-8
--utf8
-u8
-u          the input data is encoded in UTF-8

--once
-1          run only once (don't loop)

--popup
-p          popup, ie. don't move the avatar in
           Use this for a fast popup text. Use the command [stop] to also get a fast end.
```

```

--no-delay
-n          don't delay the text output
--raw
-r          output raw text (don't handle any commands or striplines)
--ignoreeof
-i          ignore end of file conditions; use this when the input is not a file

```

### 3.5.1 Environment variables

The command `avatarsay` supports the following environment variables:

```

AVATARIMAGE    image file with an avatar with the full path
DATADIR        the directory, where images and audio files are located (has no influence on the
                environment variable AVATARIMAGE)
LC_ALL
LC_CTYPE
LANG           these variables influence the default encoding and the language
SHELL          the shell for the terminal-mode
HOME           the starting directory for the terminal-mode

```

### 3.5.2 Configuration file

You can also use a configuration file with the name `‘/etc/avatarsay’` to set values for `AVATARIMAGE` and `DATADIR`.

for example:

```

AVATARIMAGE=/usr/local/share/pixmaps/myavatar.xpm
AVATARDATADIR=/usr/local/share/akfavatar

```

See [Section 3.4 \[Different avatar image\], page 8](#).

## 3.6 Commands for `avatarsay`

With the program `avatarsay` you can also write some simple demos. You don't have to learn a programming language for that.

A line starting with a hash sign (`#`, `U+0023`) is a *comment*. Those lines are simply ignored by the program. **Attention:** unlike in other scripting languages there may even be no whitespace in front of the hash sign.

You can structure your text by including a *stripline*. A stripline is a line like this: `‘-----’`. The line must start at the very first column and there must be at least three successive hypens (`U+002D`). Of course you can use more than three ones.

A *command* for an `avatarsay`-demo has to be enclosed in square brackets and it must be in the very first position of a new line.

*remark:* Earlier versions of `avatarsay` used a different style.

Most of these commands can also be used from the terminal-emulation. From a shell you can use the command `avtcmd` to send commands to `avatarsay`. Use it like this: `‘avtcmd size 5, 40’`. You can use the APC escape sequence from your own programs or scripts.

`datadir` *directory*

with this command you can change the directory. This is needed to load images and sound files.

The data directory can also be set with the environment variable `AVATARDATADIR`. The command has precedence over the environment variable.

**avatarimage** *imagefile*

with this command you can use a different image for the avatar.

You can leave the name of the imagefile away to switch back to the default avatar again. The special name `info` uses a small info-icon as avatarimage. The special name `none` removes the avatarimage. This gives you the maximum size for the text-area, but it looks boring.

**Attention:** This command should be used before the text starts This command also removes the avatar name.

The avatar image can also be set with the environment variable `AVATARIMAGE`. The command has precedence over the environment variable.

Which file-formats AKFAvatar supports depends on which libraries you have installed (see [Chapter 5 \[Formats\], page 17](#)).

**avatarname** *name*

set a name for the avatar

**encoding** *encodingname*

just for demos: sets the encoding of the text; such as ‘ISO-8859-1’ or ‘UTF-8’

**Attention:** This command has to be used at the beginning of the file. You can no longer change the encoding inside of the text (which was possible in previous versions).

This command can only be used with ASCII compatible encodings, such as the ISO-8859-series or UTF-8. The encodings UTF-16 (UCS2) and UTF-32 (UCS4) can not be set with this command. But these encodings are in most cases detected automatically. (Earlier versions of `avatarsay` could not handle these encodings at all.)

It depends on your systems `iconv` implementation, which encodings are supported. On some systems you can get a list with the command `iconv -l`.

**title** *title*

changes the title of the window. When you use this command without a title name, the title is reset to “AKFAvatar”.

**scrolling** *off | on*

switches scrolling off or on.

**slow** *on | off*

switches the slow-printing mode on or off

**backgroundcolor** *color-definition*

set a different background color

The *color-definition* is explained in [Chapter 5 \[Formats\], page 17](#).

**Attention:** This command should used before the text starts

**ballooncolor** *color-definition*

set a different balloon color

The *color-definition* is explained in [Chapter 5 \[Formats\], page 17](#).

**Attention:** This command should used before the text starts

**textcolor** *color-definition*

set a different text color

The *color-definition* is explained in [Chapter 5 \[Formats\], page 17](#).

**Attention:** The default color for the terminal is not changed by this. Be careful when you combine this with terminal codes for changing the text-color.

`left-to-right`

`right-to-left`

changes the text direction; this is useful if you have text written in Hebrew or Yiddish (Arabic is not supported)

You can only switch the text-direction on a line by line basis. Different text directions inside of one line is not supported.

`size height, width`

set the size of the balloon.

The size cannot exceed the maximum size. The value 0 sets the maximum size. the command with no values sets the maximum size for the whole balloon.

`height height`

set the height of the balloon.

The size cannot exceed the maximum size. The value 0 or no value sets the maximum size.

`width width`

set the width of the balloon.

The size cannot exceed the maximum size. The value 0 or no value sets the maximum size.

`flip`

flip the page; the same effect as with a stripline: wait some time and then clear the text-area

`clear`

clears the text-area; unlike `flip` it doesn't wait, but clears it immediately

`move out | in`

moves the avatar out or in

`wait milliseconds`

waits a given time, or a default amount of time, when no value is given

`pause`

a longer pause; waits a while, then the avatar is shown without the balloon for some time

`image imagefile`

waits some time and then shows the image for a while without the avatar

The image is centered on the screen. If the image is larger than the screen, the screen is centered on the image.

You can use a stripline *after* this command if you wish. The stripline doesn't have any effect then.

Which file-formats AKFAvatar supports depends on which libraries you have installed (see [Chapter 5 \[Formats\]](#), page 17).

`rawaudiosettings samplingrate encoding mono|stereo`

settings for loading raw audio files

This command is needed before you can play any raw audio files. Raw audio files contain audio data with no header or container format. The *encoding* can be one of `u8`, `s8`, `u16le`, `u16be`, `u16sys`, `s16le`, `s16be`, `s16sys`, `mu-law` | `u-law`, `A-law`. For example `u8` means *unsigned 8-Bit linear PCM* or `s16le` means *signed 16-Bit linear PCM, little endian*.

For example: `'rawaudiosettings 16000 mu-law mono'`

`audio audiofile`

loads and plays an audio file

The text continues to be shown, so you can play an audio file with the recorded words of the following text.

Supported are AU-files with linear PCM, mu-law and A-law encoding. Furthermore supported are WAV-files with PCM or ADPCM encoding. After the command `rawaudiosettings` is used, it can also play raw audio files.

`audioloop` *audiofile*

like `audio`, but the audio-data is played in an endless loop

`loadaudio` *audiofile*

loads an audio file for later playback

Only one single audio file can be loaded at a time. An eventually running audio-output is stopped by this command.

Supported are AU-files with linear PCM, mu-law and A-law encoding. Furthermore supported are WAV-files with PCM or ADPCM encoding. After the command `rawaudiosettings` is used, it can also play raw audio files.

`playaudio`

plays an audio file loaded by `loadaudio`

`playaudioloop`

plays an audio file loaded by `loadaudio` in an endless loop

`stopaudio`

stops the audio output immediately

`waitaudio`

wait until the audio output ends; a loop is ended by this command

This can be used to synchronize the recorded and the written text to some extend.

`effectpause`

short pause while the text stays visible

`back` *number* *text*

delete the last *number* of characters and print the *text*

You could use this command after an `[effectpause]`-command for a nice effect.

`read`

reserved for later versions

`credits` *textfile*

shows a textfile as “final credits”

`end`

just for demos: end of the text

The avatar moves out.

Anything behind the `end` command is ignored.

`stop`

just for demos: stops the output immediately

The avatar is not moved out, but the text ends immediately.

Anything behind the `stop` command is ignored.

### 3.7 Pipes

You don't always have to write static texts to use AKFAvatar. You can use the command `avatarsay` also to show you the output of other commands. Therefore you can use a single hyphen (`'-`, `U+002D`) as option.

It is best to try these features in a window based environment, where you have one window for your command line and `avatarsay` opens up in a separate window.

So try this: `echo "Hey pal, what's GNU?" | avatarsay -`. The avatar appears and says that words. Be careful with using the exclamation mark though, some shells have problems with that.

You can also see the output of any other command with `avatarsay`. For example try this: `df | avatarsay -`, or this: `dir | avatarsay -`. You could even read this manual this way: `makeinfo --plaintext akfavatar.texinfo | avatarsay -`, although that's probably a bit too long.

### 3.8 Archive-files

The program `avatarsay` can also read special archive-files. For making your own archive-files you need additional software. Archive-files are in the *AR*-format. You can use for example the `ar`-program that comes with the "GNU binutils". But be careful: there are some requirements and limitations.

Archive files should get the file-extension `.avt`.

The very first archive member must be the script for `avatarsay` and it must have the name `AKFAvatar-demo`. The other archive members can be added in any order you like.

The names of archive members are limited to 15 characters. GNU `ar` and some other `ar` implementations have extensions to support longer file names, but `avatatarsay` cannot read that. On the other hand, I think that 15 characters is more than enough. Note that archive members need not have file-extensions, they are recognized from their content. To make sure that filenames get cut at 15 characters you can use the option `-f` with GNU `ar` or `-T` for some other `ar` implementations. The TARGA image file format cannot be used in archive files.

An archive-file should have a member with the name `info`. So to get more information about an archive you can use the command `ar p demo.avt info`. This `info`-member can be the same you use for the final-credits.

Other programs can recognize archive-files for `avatarsay` from the beginning of the file. It always starts with `!<arch>\nAKFAvatar`, where `\n` is one single character with the value 10.

## 4 Using it under GNOME

This chapter explains the use of the tool `gnome-akfavatar`. It is mainly a frontend for the command `avatarsay` with some of its functionality presented in a menu.

### 4.1 What you need to use `gnome-akfavatar`.

The command `gnome-akfavatar` is a shell script, so it needs a Bourne shell. It has been tested with GNU `bash`, but any POSIX-compatible Bourne shell should be sufficient.

Then it needs the command `avatarsay`. It is searched in the `PATH` and also in the current directory.

For the visible interface it uses the program `zenity`. As text-editor it needs `gedit` and as help browser it uses `yelp`. These commands have to be in your `PATH`. They are usually included with GNOME.

Since the command `avatarsay` is also looked up in the current directory the package need not be installed. But the documentation has to be installed in order to view it from `gnome-akfavatar`.

### 4.2 The main-menu of `gnome-akfavatar`

When you invoke the command `gnome-akfavatar` you will get a menu from which you can choose what to do.

You get the following menu-items:

`show a demo or textfile`

You get a file selection box and you can choose a text-file or a demo, which is then opened in `avatarsay`. A *demo* is either a text-file, which uses commands for `avatarsay` (see [Section 3.6 \[Commands for avatarsay\]](#), page 10), or it can also be a package that also contains images or audio-files.

`create or edit a demo`

can be used to create a new demo or edit an existing one. When you enter a filename which doesn't exist yet, the file is created with an appropriate `#!`-line. If you chose a different avatar-image before invoking this item, a command `[avatarimage]` is also included in this file. The executable bit will be set on the created file. Note: All this is just be done for *new* files, not for files which already existed before.

`show a manpage`

you are prompted to enter the name of a manpage you want to view. If you want to see a manpage from a special section, you can enter the section number in front of the name separated by a space. For example `'6 intro'` shows the intro manpage of section 6.

`show the output of a command`

shows the output of a command in `avatarsay`. This is only for commands which print something to the standard output or standard error channel. It is not for interactive commands or commands that use curses.

`change avatar image`

lets you select a different avatar image. See [Section 3.4 \[Different avatar image\]](#), page 8. This setting is only used for the current session unless you use the item *save settings*.

`fullscreen or window mode`

click on `Okay` to select the fullscreen-mode, or `Cancel` to select the window-mode. This setting is only used for the current session unless you use the item *save settings*.

**save settings**

saves the settings, that is the avatar image and whether to run in fullscreen or window mode. Note: these settings are just for `gnome-akfavatar`; they have no influence when you call `avatarsay` directly or any other program using the AKFAvatar library.

**help for AKFAvatar**

shows this document in the help-browser of GNOME

**Website** opens the homepage for AKFAvatar in a web-browser.

**Exit** exit the program. Note: you can also exit the program by clicking on **Cancel** or the **close** button of the window or pressing the ESC-key.

## 5 About file formats and values

### 5.1 Colors

One way to define a color is to name it in plain English. Names with a blank can sometimes not be used. But you can write it together in one word then. An overview over the recognized names is in [Appendix C \[Color names\]](#), page 47.

Another way is to write the RGB values in hexadecimal, introduced by the hash character (#). You can use a 6 or 3 digits value. For example for intensive red you can write either #f00 or #ff0000. The first value (one or two digits) is for red, the next for green and the last for blue. With these values you can mix any possible color. More examples: black is #000, white is #fff, yellow is #ff0, violet is #f0f and so on. For darker colors you simply use lower values: #808 is a dark violet, while #fdf a very light violet.

### 5.2 Images

AKFAvatar supports images in the X-Pixmap (XPM) format, in the X-Bitmap (XBM) format, and in the uncompressed BMP format out of the box. The XPM format is well suited for avatar-images, because it supports transparency. For other formats the software has to do a very dirty hack to achieve some pseudo-transparency (see [Section 3.4 \[Different avatar image\]](#), page 8). XPM and XBM area also much better for programmers, because the files can easily be included in code, especially in code of C compatible languages. But there is also one drawback: XPM files with lots of colors can get very large. So better just use few colors. But there is no arbitrary limit.

If you have the library `SDL_image` installed, you can use a couple of other image formats too, including PNG and JPEG, if you also have the libraries for them. Note however that JPEG is not suited for avatar images (the “dirty trick” described above doesn’t work with JPEG at all).

The XPM format, the XBM format and the BMP format are not compressed, so they occupy some space on the harddisk. But diskspace should not be a big issue nowadays. Note however: when you put them into a compressed archive file for distribution, then this doesn’t make much difference. Then these files will actually be compressed, while files, that already were compressed, cannot be compressed much further. So that evens out. (Only the JPEG format has an advantage here, because it is a lossy format.)

### 5.3 Audio files

Supported are AU-files with linear PCM, mu-law and A-law encoding. Files with 24 or 32 bit linear PCM can be read, but they are only played with a quality of 16 bit.

Also supported is “raw” audio data with 8 or 16 bit linear PCM or mu-law or A-law encodings.

If you want to keep your files small, use mu-law or A-law. These encodings just use 8 bit per sample, just like 8 bit PCM, but they sound much better. A good samplingrate for speech recordings would be 16000Hz. If you want to save more space and phone-quality is good enough for you, you can use a samplingrate of 8000Hz.

Furthermore supported are Wave-files with 8 or 16 bit linear PCM or ADPCM encoding.

All these formats can be stored in mono or stereo. More than two channels are not supported. Please note that for all of these formats stereo needs twice as much space as mono does.

## 6 Programming with AKFAvatar

If you want to learn how to program, Pascal is a very good choice as your first language. But unfortunately Pascal is not so widely spread. Nevertheless, with **GNU-Pascal** and **Free Pascal** we have two very good Free Software implementations for Pascal, which are available for many different platforms.

Before you ask: Yes, it would have been possible to write the whole thing in Pascal. But C is more widely spread, and that has its consequences. Every modern system comes with a C compiler and almost every other language can use libraries written in C.

### 6.1 How to program with AKFAvatar in Pascal

You can use AKFAvatar for your Pascal programs.

First of all the library must already be installed on your system. You also need the file `'akfavatar.pas'`. That file usually is not automatically installed, because there is no standard directory for that. But it is found, if it simply is in the current directory.

#### 6.1.1 Simple use cases

In the source-package you will find the scripts `'gpcavatar'` and `'fpcavatar'` in the subdirectory `'pascal/'`. If you have GNU-Pascal (`gpc`) just try this: `'./gpcavatar example.pas'` This should compile a program `'example'`.

You can compile any pascal program, that just uses plain standard input/output like this. There are no changes to the source-code of the pascal program necessary!

If you want to start a new page, simply use the command `page;`.

That is easy, isn't it?

*Note:* If the program uses the CRT unit, you have to make changes to your program as described below.

#### 6.1.2 Installing

Have a look into the scripts `'gpcavatar'` and `'fpcavatar'`. There are some variables you should adapt to your needs. Copy the scripts into `'/usr/local/bin'`, so you can use them from anywhere. Copy the file `'akfavatar.pas'` to where you want to have your personal units and make sure that directory is listed in the scripts.

#### 6.1.3 Advanced use

It is also possible to port programs, which use the CRT unit with AKFAvatar. But then you have to make changes to your program.

The scripts `'gpcavatar'` and `'fpcavatar'` both define the symbol `AKFAVATAR`, so you can use `{$IFDEF AKFAVATAR}` or `{$IFNDEF AKFAVATAR}` to control what to do when it is compiled with AKFAvatar or without.

To make sure your program can still be used with the CRT unit, you can make something like this: `'{$IFNDEF AKFAVATAR} uses CRT; {$ENDIF}'`. The program `'multiply.pas'` is an example for this technique.

AKFAvatar supports most commands and variables from the original CRT unit. For example the command `ClrScr` clears the text-area (not the whole screen!), `ClrEol` clears the rest of the line. You can use the command `GotoXY(x, y);` to go to a special position inside the text-area, or use the functions `WhereX` and `WhereY` to find out the current position.

**Attention:** The width of the display is 80 characters as with a terminal display, but the height is often much smaller.

You can find out the dimensions of the text-area with `ScreenSize.x` and `ScreenSize.y`. The avatar is shown, when it wasn't visible yet. So if you want to change the avatar-image or the background color, you have to do it, before asking for the size.

You can use the command `TextColor(color)`; to change the color of the text. You can even set the background color of the text with the command `TextBackground(color)`;, but that doesn't look as good as on a text-terminal.

Even the variable `TextAttr` is fully supported as in the original CRT unit. But it is better to avoid the use of that.

The commands `HighVideo` and `LowVideo` are supported, although with a slightly different meaning. `HighVideo` switches bold font on and `LowVideo` switches bold font off. They do not change the color as with the CRT unit. Please keep that in mind, when you combine them with calls to `TextColor`.

With `Underlined(true)`; you can switch underlined mod on and `Underlined(false)`; you can switch it off.

To switch back to the "normal" text-attributes, you should use the command `NormVideo`;. The "normal" text-color is different when you use it with AKFAvatar or with the CRT unit. So with this command you can ensure, that you get the desired result with both. `NormVideo`; also switches the bold- and the underlined-mode off.

The function `ReadKey` waits for a key to be pressed and returns the code of the pressed key. To find out, whether a key was pressed without blocking the program, you can use the function `KeyPressed`.

**Attention:**

Function keys are not supported yet. The key ESC stops the program immediately. If you need the ESC-key for something else, you can set the variable `CheckEsc` to `false`.

The program also stops when CTRL+C is pressed. You can set the variable `CheckBreak` to `false` to avoid that.

The variable `CheckBreak` is compatible to the CRT unit, while `CheckEsc` is an extension.

There are many additional commands, which are not compatible to the CRT unit.

You can use a different image for the avatar with the command

```
{$IfDef AKFAVATAR}
AvatarImageFile ('/usr/local/share/pixmaps/teacher.xpm');
{$EndIf}
```

This is of course only an example. This command must be used before any input/output took place! The size of the avatar image also influences the size of the text-area. Thus this must be used, before using the function `ScreenSize`.

If you want to change the background color of the window (not the text-area), you can use the command `setBackgroundcolor(red, green, blue)`; at the beginning of the program. The value for red, green and blue represent the intensity of that color. You can mix up any possible color with these 3 values. The maximum is 255 (\$FF). For example `'setBackgroundcolor(0, 0, 100);'` sets a dark blue background color.

You can define in your program which charset encoding you use. If you have an old system based on the Latin-1 charset, use the command `setEncoding('ISO-8859-1')`;. If you have to handle with different charsets at the same time, you can repeat this command whenever you want to change it. Different charsets can be visible in the text-area at the same time. (Internally a Unicode charset is used.)

There are also characters for Hebrew or Yiddish texts. To use them you have to change the text direction. You can do this with the command `setTextDirection (RightToLeft);` and `setTextDirection (LeftToRight);`. Please make sure that you start a new line or page before or after using this command. Mixing different text directions in one line is not supported.

The input/output is done with the common Pascal commands (`Read`, `ReadLn`, `Write`, `WriteLn`, `Page`). You can use the full Pascal formatting-syntax. For example: `WriteLn ('Pi is ', Pi:0:8, ' and so on.');`

If you want to enforce a new page, use the command `page;`. This is in fact also a Standard-Pascal command, which is seldom used; so seldom, that it is even missing in Free Pascal. But my unit defines it also for Free Pascal. Normally the command `page;` is used to get a new page on a printer.

You can also clear the text-area with the command `ClrScr;`. Unlike `page;` the command `ClrScr;` doesn't wait, but clears the text-area immediately. You can put the cursor in a deliberate position with the command `GotoXY (x, y);`. You find out, where the cursor is with the functions `WhereX` or `WhereY`. The coordinates 1, 1 is in the upper left corner (independent from the text direction). You can find the maximum positions with `ScreenSize.x` and `ScreenSize.y`. All these names were chosen to get some compatibility with the `CRT` unit. Actually they don't handle the screen, but only the text-area!

If you want to print warnings, error messages or debugging information to the text-terminal, use the file-descriptor `stderr`. (This doesn't work on Windows or ReactOS systems.) For example: `WriteLn (stderr, 'Error: ', AvatarGetError);` Well, the function `AvatarGetError` gets an error message from the library.

The command `ShowAvatar;` shows only the avatar without the balloon. The commands `MoveAvatarIn;` or `MoveAvatarOut;` move the avatar in or out. The Command `Delay (500);` waits approximately for 500 milliseconds. If you prefer the scale in seconds use `Delay (seconds(0.5));`.

A full reference for the Pascal language can be found in [Appendix A \[Pascal reference\]](#), page 21.

## Appendix A Pascal reference

The following text shows the interface of the Pascal unit `akfavatar.pas`:

```
unit akfavatar;

interface

{ length of an output line }
{ input is one less }
const LineLength = 80;

{ Default encoding of the system }
{$IfDef LATIN1}
  const DefaultEncoding = 'ISO-8859-1';
{$Else}
  const DefaultEncoding = 'UTF-8';
{$EndIf}

{ defaults for SetTextDelay and SetFlipPageDelay }
const
  DefaultTextDelay = 75;
  DefaultFlipPageDelay = 2700;

{ Colors for TextColor/TextBackground }
{ compatible to the CRT unit }
const
  Black      = 0;
  Blue       = 1;
  Green      = 2;
  Cyan       = 3;
  Red        = 4;
  Magenta    = 5;
  Brown      = 6;
  LightGray  = 7;
  DarkGray   = 8;
  LightBlue  = 9;
  LightGreen = 10;
  LightCyan  = 11;
  LightRed   = 12;
  LightMagenta = 13;
  Yellow     = 14;
  White      = 15; { as background-color -> balloon-color }
  Blink      = 128; { ignored }

{ for TextBackground }
const BalloonColor = 15;

{$IfDef FPC}
  type LineString = AnsiString;
{$Else}
  { In UTF-8 encoding one char may take up to 4 Bytes }
  type LineString = string (4 * LineLength);
```

```

    type ShortString = string (255);
{$EndIf}

type TextDirection = (LeftToRight, RightToLeft);

type TScreenSize = record x, y: integer end;

{
  Text Attributes
  mostly compatible to the CRT unit
  the "blink-bit" means a bright background color
}
var TextAttr : byte;

{ methods to stop the program }
var
  CheckBreak: boolean; { compatible to CRT }
  CheckEsc: boolean;

{ compatible to the CRT unit }
var
  CheckEof: boolean;
  CheckSnow: boolean;
  DirectVideo: boolean;

{ for CRT compatiblity, use ScreenSize for new programs }
{ Just for reading! }
{ These variables are only set after the avatar is visible }
var WindMin, WindMax: word;

{ load the Avatar image from a file }
{ should be used before any output took place }
{ but it can be changed later }
procedure AvatarImageFile(FileName: string);

{ load the Avatar image from memory }
{ should be used before any output took place }
{ but it can be changed later }
procedure AvatarImageData(data: pointer; size: LongInt);

{ load the Avatar image from XPM-data }
{ use the tool xpm2pas to import the data }
{ should be used before any output took place }
{ but it can be changed later }
{ example: AvatarImageXPM(addr(image)); }
procedure AvatarImageXPM(data: pointer);

{ load the Avatar image from XBM-data }
{ use the tool xbm2pas to import the data }
{ should be used before any output took place }
{ but it can be changed later }
{ example:

```

```

AvatarImageXBM(addr(img_bits), img_width, img_height, 'black'); }
procedure AvatarImageXBM(bits: pointer; width, height: integer;
    colorname: string);

{ give the avatar a name }
procedure AvatarName(const Name: string);

{ set a different background color }
{ should be used before any output took place }
procedure setBackgroundColor(red, green, blue: byte);
procedure setBackgroundColorName (const Name: string);

{ set a different balloon color }
{ should be used before any output took place }
procedure setBalloonColor(red, green, blue: byte);
procedure setBalloonColorName (const Name: string);

{ change pace of text and page flipping }
{ the scale is milliseconds }
procedure setTextDelay(delay: integer);
procedure setFlipPageDelay(delay: integer);

{ change the encoding }
procedure setEncoding(const newEncoding: string);

{ change text direction (for hebrew/yiddish texts) }
{ you should start a new line before or after this command }
procedure setTextDirection(direction: TextDirection);

{ The "Screen" is the textarea }
{ The name is chosen for compatibility with the CRT unit }
{ This causes the library to be initialized }
{ The avatar-image and the background color should be set before this }
function ScreenSize: TScreenSize;

{ assign text-variable to the avatar }
procedure AssignAvatar(var f: text);

{ the same for CRT compatibility }
procedure AssignCrt(var f: text);

{ Restore Input/Output system }
{ use this to output help or version information }
procedure RestoreInOut;

{$IFDEF FPC}
    { the page command is defined in the Pascal standard,
      but missing in FreePascal }

    { action: wait a while and then clear the textfield }

    procedure page(var f: text);

```

```
    procedure page;
{$EndIf}

{ switch cursor on or off }
{ extensions compatible to Free Pascal }
procedure CursorOn;
procedure CursorOff;

{ keyboard handling }
{ partly CRT compatible - only Latin1 chars so far }
function KeyPressed: boolean;
function ReadKey: char;

{ clear the keyboard buffer }
procedure ClearKeys;

{ wait for a key }
procedure WaitKey;

{ wait some time }
{ compatible to CRT unit }
procedure delay(millisecond: integer);

{ example use: delay (seconds (2.5)); }
function seconds(s: Real): integer;

{ clears the window (not the screen!) }
{ the name was chosen for compatibility to the CRT unit }
procedure ClrScr;

{ clears rest of the line }
{ compatible to CRT unit }
procedure ClrEol;

{ deletes current line, the rest is scrolled up }
procedure DelLine;

{ inserts a line before the current line, the rest is scrolled down }
procedure InsLine;

{ set the text color }
{ compatible to CRT unit }
procedure TextColor (Color: Byte);

{ set the text background color }
{ compatible to CRT unit, but light colors can be used }
procedure TextBackground (Color: Byte);

{ set black on white text colors, switch bold and underlined off }
{ name compatible to CRT unit, but the colors differ }
procedure NormVideo;
```

```
{ switch bold mode on or off }
{ this is different from the CRT unit }
{ be careful, when you combine this with TextColor }
procedure HighVideo;
procedure LowVideo;

{ switch underline mode on or off }
procedure Underlined (onoff: boolean);

{ shows the avatar without the balloon }
procedure ShowAvatar;

{ moves the avatar in or out }
procedure MoveAvatarIn;
procedure MoveAvatarOut;

{ loads image
  after that call delay or waitkey
  the supported image formats depend on your libraries
  XPM and uncompressed BMP is always supported
}
function ShowImageFile(FileName: string): boolean;
procedure ShowImageData(data: pointer; size: LongInt);

{ use the tool xpm2pas to import the X Pixmap data }
{ example: ShowImageXPM(addr(image)); }
procedure ShowImageXPM(data: pointer);

{ use the tool xbm2pas to import the X Bitmap data }
{ example:
  ShowImageXBM(addr(img_bits), img_width, img_height, 'black'); }
procedure ShowImageXBM(bits: pointer; width, height: integer;
                       colorname: string);

{ play a short sound as with chr(7) }
procedure Beep;

{ a short visual flash on the screen }
procedure Flash;

{ loads Audio File
  AU or WAV files supported }
function LoadSoundFile(const FileName: string): pointer;
function LoadSoundData(data: pointer; size: LongInt): pointer;
procedure FreeSound(snd: pointer);
procedure PlaySound(snd: pointer; loop: boolean);

{ wait until the end of the audio output }
procedure WaitSoundEnd;

{ play a sound of a given frequency }
{ Note: not fast, needs a delay(200) at least }
```

```

procedure Sound(frequency: integer);

{ stop sound output }
procedure NoSound;

{ handle coordinates (inside the balloon) }
{ compatible to CRT unit }
function WhereX: integer;
function WhereY: integer;
procedure GotoXY(x, y: integer);
procedure Window(x1, y1, x2, y2: Byte);

{ whether the cursor is in the home position? }
function HomePosition: boolean;

{ set the size of the balloon }
{ the window is reset to the new full size }
procedure BalloonSize(height, width: integer);
procedure BalloonWidth(width: integer);
procedure BalloonHeight(height: integer);

{ set/get scroll mode }
{ 0 = off (page-flipping), 1 = normal }
procedure SetScrollMode(mode: integer);
function GetScrollMode: integer;

{ get last error message }
function AvatarGetError: ShortString;

{ ignore TextColor TextBackground and so on }
{ compatible with GNU-Pascal's CRT unit }
procedure SetMonochrome(monochrome: boolean);

{ for positive/negative questions }
{ keys for positive: + 1 Enter }
{ keys for negative: - 0 Backspace }
function Decide: boolean;

{ Navigate }
{
  navigation bar

  buttons is a string with the following characters
  'l': left
  'r': right (play)
  'd': down
  'u': up
  'x': cancel
  'f': (fast)forward
  'b': (fast)backward
  'p': pause
  's': stop
}

```

```
'e': eject
'*': circle (record)
'+': plus (add)
'-': minus (remove)
'?': help
' ': spacer (no button)
```

Pressing a key with one of those characters selects it.  
 For the directions you can also use the arrow keys,  
 The [Pause] key returns 'p'.  
 The [Help] key or [F1] return '?'.

the function returns the letter for the selected option

example:

```
  case Navigate('lrx') of ...
}
```

```
function Navigate(buttons: String): char;
```

```
{ choice for several items }
{ result is the choice number, starting from 1 }
{ startkey may be #0 }
function Choice(start_line, items: integer; startkey: char;
               back, fwd: boolean): integer;
```

```
{ show a very long text in a pager }
{ You can navigate with up/down, page up/page down keys,
  Home and End keys, and even with the mouse-wheel }
procedure PagerString (const txt: string; startline: integer);
procedure PagerFile (const filename: string; startline: integer);
```

```
{ lock or unlock updates - can be used for speedups }
{ when true the text_delay is set to 0 }
{ when false the textarea gets updated }
{ use with care! }
procedure LockUpdates(lock: boolean);
```

## Appendix B C reference

The following is the C header file `akfavatar.h`:

```

/*
 * AKFAvatar library - for giving your programs a graphical Avatar
 * Copyright (c) 2007, 2008, 2009 Andreas K. Foerster <info@akfoerster.de>
 *
 * needed:
 *   SDL1.2 (recommended: SDL1.2.11 or later (but not 1.3!))
 * recommended:
 *   SDL_image1.2
 *
 * This file is part of AKFAvatar
 *
 * AKFAvatar is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * AKFAvatar is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, see <http://www.gnu.org/licenses/>.
 */

#ifndef _akfavatar_h
#define _akfavatar_h

/* SDL redefines main on some systems */
#if defined(__WIN32__) || defined(__MACOS__) || defined(__MACOSX__)
# include "SDL.h"
#endif

/* to get the systems definition of wchar_t */
#include <stddef.h>

#define AKFAVATAR 1

/* maximum linelength */
#define AVT_LINELENGTH 80

/* for avt_initialize */
#define AVT_AUTOMODE -1
#define AVT_WINDOW 0
#define AVT_FULLSCREENNOSWITCH 2

#ifdef AVT_NOSWITCH
# define AVT_FULLSCREEN AVT_FULLSCREENNOSWITCH
#else

```

```

# define AVT_FULLSCREEN 1
#endif

/* for _avt_STATUS */
#define AVT_NORMAL 0
#define AVT_QUIT 1
#define AVT_ERROR -1

/* for boolean expressions */
/* see avt_bool_t in the type definitions */
#define AVT_TRUE 1
#define AVT_FALSE 0
#define AVT_MAKE_BOOL(x) ((x) != 0)

/* for avt_set_text_delay and avt_set_flip_page_delay */
#define AVT_DEFAULT_TEXT_DELAY 75
#define AVT_DEFAULT_FLIP_PAGE_DELAY 2700

/* for avt_text_direction */
#define AVT_LEFT_TO_RIGHT 0
#define AVT_RIGHT_TO_LEFT 1

/*
 * example: avt_wait(AVT_SECONDS(2.5)) waits 2.5 seconds
 */
#define AVT_SECONDS(x) ((x)*1000)

/* macro for marking unused symbols */
#ifdef __GNUC__
# define AVT_UNUSED __attribute__((__unused__))
#else
# define AVT_UNUSED
#endif /* __GNUC__ */

#ifdef __cplusplus
# define AVT_BEGIN_DECLS extern "C" {
# define AVT_END_DECLS   }
#else
# define AVT_BEGIN_DECLS
# define AVT_END_DECLS
#endif /* __cplusplus */

/* for later use */
#define AVT_API extern

/*****
/* type definitions */

/*
 * boolean are chars for this library
 * use AVT_TRUE or AVT_FALSE as values,
 * or AVT_MAKE_BOOL(x) to convert values.

```

```

*
* You can use true or false and the type bool from stdbool.h in C
* or the equivalent things from C++ - it's all compatible.
* I just make my own type for old compilers.
*/
typedef unsigned char avt_bool_t;

/*
* general types for avatar images and audio data
* may change in the future!
*/
typedef void avt_image_t;
typedef void avt_audio_t;

/* for streams (use FILE from your programs) */
typedef void avt_stream;

AVT_BEGIN_DECLS

/*****
/* functions */
/* most functions return the status, unless otherwise stated */
*****/

/*****
/* initialization / finalization */

/*
* initialize the avatar system
*
* mode is either AVT_WINDOW or AVT_FULLSCREEN or AVT_FULLSCREENNOSWITCH.
* The original image is freed in this function!
* So you can directly put calls to avt_defaults
* or the avt_import_* functions here.
* the image may be NULL if no avatar should be shown
* title and/or icontitle may also be NULL
*/
AVT_API int avt_initialize (const char *title,
                          const char *icontitle,
                          avt_image_t *image,
                          int mode);

/*
* quit the avatar system
* can be used with atexit
*/
AVT_API void avt_quit (void);

/*
* call avt_wait_button (); avt_move_out (); avt_quit ();
* can be used with atexit

```

```

*/
AVT_API void avt_button_quit (void);

/*****
/* getting an avatarimage */

/*
* these functions can be used directly with
* avt_initialize or avt_change_avatar_image
* they call avt_make_transparent if appropriate
*/

/*
* X-Pixmaps (XPM), X Bitmaps (XBM) and uncompressed BMP are always supported
* other image formats are supported with SDL_image
*/

/* get the default avatar image */
AVT_API avt_image_t *avt_default (void);

/* import an avatar from XPM data */
AVT_API avt_image_t *avt_import_xpm (char **xpm);

/* import an avatar from XBM data */
AVT_API avt_image_t *avt_import_xbm (const unsigned char *bits,
                                   int width, int height,
                                   const char *colorname);

/* RGB gimp_image */
AVT_API avt_image_t *avt_import_gimp_image (void *gimp_image);

/* import avatar from image data */
AVT_API avt_image_t *avt_import_image_data (void *img, int imgsize);

/* import avatar from file */
AVT_API avt_image_t *avt_import_image_file (const char *filename);

/* import avatar from stream */
AVT_API avt_image_t *avt_import_image_stream (avt_stream *stream);

/*****
/* other functions for avatarimages */

/*
* change avatar image while running
* if the avatar is visible, the screen gets cleared
* the original image is freed in this function!
* the image may be NULL if no avatar should be shown
* on error AVT_ERROR is set and returned
* an avatar name is cleared
*/
AVT_API int avt_change_avatar_image (avt_image_t *image);

```

```

/*
 * free avt_image_t images
 * (automatically called in avt_initialize and avt_change_avatar_image)
 */
AVT_API void avt_free_image (avt_image_t *image);

/*
 * make background transparent
 * pixel in the upper left corner is supposed to be the background color
 */
AVT_API avt_image_t *avt_make_transparent (avt_image_t *image);

/*****
/* actions without or outside the balloon */
/* see also "showing images without the avatar" */

/* show an empty screen with the background color */
AVT_API void avt_clear_screen (void);

/* show just the avatar without the balloon */
AVT_API void avt_show_avatar (void);

/* like avt_show_avatar, but the avatar is moved in */
AVT_API int avt_move_in (void);

/* move the avatar out => empty screen */
AVT_API int avt_move_out (void);

/*
 * make a short sound, when audio is initialized
 * else it is the same as avt_flash
 * same as with \a in avt_say
 * the sound is actually not a bell ;- )
 */
AVT_API void avt_bell (void);

/* visual flash of the screen */
AVT_API void avt_flash (void);

/* update, ie handle events and give some time to other processes */
/* use this in a longer loop in your program */
AVT_API int avt_update (void);

/* wait a while */
AVT_API int avt_wait (int milliseconds);

/*****
/* say or ask stuff with wchar_t (Unicode: UTF-32) */

/*
 * prints a L'\0' terminated string in the balloon

```

```

* if there is no balloon, it is drawn
* if there is no avatar, it is shown (not moved in)
* interprets control chars including overstrike-text
*/
AVT_API int avt_say (const wchar_t *txt);

/*
* writes string with given length in the balloon
* the string needn't be terminated and can contain binary zeros
* if there is no balloon, it is drawn
* if there is no avatar, it is shown (not moved in)
* interprets control characters including overstrike-text
*/
AVT_API int avt_say_len (const wchar_t *txt, const int len);

/*
* writes a single character in the balloon
* if there is no balloon, it is drawn
* if there is no avatar, it is shown (not moved in)
* interprets control characters, but not for overstrike-text
*/
AVT_API int avt_put_character (const wchar_t ch);

/*
* get string (just one line)
* the maximum length is LINELENGTH-1
* size is the size of s in bytes (not the length)
*
* (I don't use size_t for better compatibility with other languages)
*/
AVT_API int avt_ask (wchar_t *s, const int size);

/*
* get a character from the keyboard
* only for printable characters, not for function keys
* (ch is a pointer to one character, not a string)
*/
AVT_API int avt_get_key (wchar_t *ch);

/*****
/* say or ask stuff with multy-byte encodings */

/* set encoding for mb functions */
AVT_API int avt_mb_encoding (const char *encoding);

/*
* prints a 0 terminated string in the balloon
* if there is no balloon, it is drawn
* if there is no avatar, it is shown (not moved in)
* interprets control chars including overstrike-text
*
* converts from a given charset encoding

```

```

    * (see avt_mb_encoding)
    */
AVT_API int avt_say_mb (const char *txt);

/*
 * the same with a given length
 * the string needn't be terminated then
 * and can contain binary zeros
 */
AVT_API int avt_say_mb_len (const char *txt, int len);

/*
 * get string (just one line)
 * converted to the given encoding
 *
 * for UTF-8 encoding s should have a capacity of 4 * LINELENGTH Bytes
 */
AVT_API int avt_ask_mb (char *s, const int size);

/*
 * decode a string into wchar_t
 * size in bytes
 * returns number of characters in dest (without the termination zero)
 * dest must be freed by caller with avt_free
 */
AVT_API int avt_mb_decode (wchar_t **dest, const char *src, const int size);

/*
 * encode a string from wchar_t
 * len is the length
 * returns number of characters in dest (without the termination zero)
 * dest must be freed by caller with avt_free
 * (the size of dest may be much more than needed)
 */
AVT_API int avt_mb_encode (char **dest, const wchar_t *src, const int len);

/* free memory allocated by this library */
AVT_API void avt_free (void *ptr);

/*****
/* informational stuff */

/* is it initialized? */
AVT_API avt_bool_t avt_initialized (void);

/* 0 = normal; 1 = quit-request; -1 = error */
AVT_API int avt_get_status (void);

/* set status */
AVT_API void avt_set_status (int status);

/* get error message */

```

```

AVT_API char *avt_get_error (void);

/* which version of the linked library is used? */
AVT_API const char *avt_version (void);

/* get copyright information */
AVT_API const char *avt_copyright (void);

/* get license information */
AVT_API const char *avt_license (void);

/* get color name of given number, or NULL on error */
AVT_API const char *avt_get_color_name (int nr);

/*
 * get the color definition
 * returns the color name or NULL on error
 */
AVT_API const char *avt_get_color (int nr, int *red, int *green, int *blue);

/*
 * get color values for a given color-name
 * returns 0 on success or -1 on error
 */
AVT_API int avt_name_to_color (const char *name,
                              int *red, int *green, int *blue);

/*****
/* settings */

/*
 * change the title and/or the icontitle
 * use NULL for the unchanged part
 * in newer SDL-versions it is to be encoded in UTF-8
 * if possible stick to ASCII for compatibility
 */
AVT_API void avt_set_title (const char *title, const char *icontitle);

/*
 * set name for the avatar
 * set to NULL to clear the name
 */
AVT_API int avt_set_avatar_name (const wchar_t *name);
AVT_API int avt_set_avatar_name_mb (const char *name);

/* switch to fullscreen or window mode */
AVT_API void avt_switch_mode (int mode);

/* toggle fullscreen mode */
AVT_API void avt_toggle_fullscreen (void);

/* get mode */

```

```
AVT_API int avt_get_mode (void);

/*
 * set the baloon width and height in number of characters
 * 0 or less for maximum width
 * if it's actually changed, the balloon is redrawn and emptied
 * see also avt_get_max_x () and avt_get_max_y ()
 */
AVT_API void avt_set_balloon_size (int height, int width);
AVT_API void avt_set_balloon_width (int width);
AVT_API void avt_set_balloon_height (int height);

/* activate the text cursor? (default: no) */
AVT_API void avt_activate_cursor (avt_bool_t on);

/*
 * define the background color
 * values in the range 0x00 .. 0xFF
 * can and should be called before avt_initialize
 * if the balloon is visible, it is cleared
 */
AVT_API void avt_set_background_color (int red, int green, int blue);
AVT_API void avt_set_background_color_name (const char *name);

/*
 * define the balloon color
 * values in the range 0x00 .. 0xFF
 * can be called before avt_initialize
 * the text-background-color is set to the balloon-color too
 * if the balloon is visible, it is cleared
 */
AVT_API void avt_set_balloon_color (int red, int green, int blue);
AVT_API void avt_set_balloon_color_name (const char *name);

/*
 * change the text color
 * values in the range 0x00 .. 0xFF
 */
AVT_API void avt_set_text_color (int red, int green, int blue);
AVT_API void avt_set_text_color_name (const char *name);
AVT_API void avt_set_text_background_color (int red, int green, int blue);
AVT_API void avt_set_text_background_color_name (const char *name);

/* set text background to balloon color */
AVT_API void avt_set_text_background_ballooncolor (void);

/*
 * set text direction
 * the cursor is moved to start of the line
 * in a text, you might want to call avt_newline after that
 */
AVT_API void avt_text_direction (int direction);
```

```
/*
 * delay time for text-writing
 * default: AVT_DEFAULT_TEXT_DELAY
 */
AVT_API void avt_set_text_delay (int delay);

/*
 * delay time for page flipping
 * default: AVT_DEFAULT_FLIP_PAGE_DELAY
 */
AVT_API void avt_set_flip_page_delay (int delay);

/* set underlined mode on or off */
AVT_API void avt_underlined (avt_bool_t onoff);

/* get underlined mode */
AVT_API avt_bool_t avt_get_underlined (void);

/* set bold mode on or off (not recommended) */
AVT_API void avt_bold (avt_bool_t onoff);

/* get bold mode */
AVT_API avt_bool_t avt_get_bold (void);

/* set inverse mode on or off */
AVT_API void avt_inverse (avt_bool_t onoff);

/* get inverse mode */
AVT_API avt_bool_t avt_get_inverse (void);

/* set default color and switch off bold, underlined, inverse */
AVT_API void avt_normal_text (void);

/*
 * set scroll mode
 * -1 = off, 0 = page-flipping, 1 = normal
 */
AVT_API void avt_set_scroll_mode (int mode);
AVT_API int avt_get_scroll_mode (void);

/* set newline mode (default: on) */
AVT_API void avt_newline_mode (avt_bool_t mode);

/* set auto-margin mode (default: on) */
AVT_API void avt_auto_margin (avt_bool_t mode);

/*
 * origin mode
 * AVT_FALSE: origin (1,1) is always the top of the textarea
 * AVT_TRUE:  origin (1,1) is the top of the viewport
 */
```

```

AVT_API void avt_set_origin_mode (avt_bool_t mode);
AVT_API avt_bool_t avt_get_origin_mode (void);

/*
 * with this you can switch the mouse pointer on or off
 * use this after avt_register_mousehandler
 */
AVT_API void avt_set_mouse_visible (avt_bool_t visible);

/*****
/* handle coordinates inside the balloon */

/*
 * the coordinates start with 1, 1 in the upper left corner
 * and are independent from the text direction
 */

/* get position in the viewport */
AVT_API int avt_where_x (void);
AVT_API int avt_where_y (void);

/*
 * is the cursor in the home position?
 * (also works for right-to-left writing)
 */
AVT_API avt_bool_t avt_home_position (void);

/* maximum positions (whole text-field) */
AVT_API int avt_get_max_x (void);
AVT_API int avt_get_max_y (void);

/* put cursor to specified coordinates */
AVT_API void avt_move_x (int x);
AVT_API void avt_move_y (int y);
AVT_API void avt_move_xy (int x, int y);

/* save and restore current cursor position */
AVT_API void avt_save_position (void);
AVT_API void avt_restore_position (void);

/* set a viewport (sub-area of the textarea) */
AVT_API void avt_viewport (int x, int y, int width, int height);

/*****
/* activities inside the balloon */

/* new line - same as \n in avt_say */
AVT_API int avt_new_line (void);

/* wait a while and then clear the textfield - same as \f in avt_say */
AVT_API int avt_flip_page (void);

```

```
/*
 * clears the viewport
 * if there is no balloon yet, it is drawn
 */
AVT_API void avt_clear (void);

/*
 * clears from cursor position down the viewport
 * if there is no balloon yet, it is drawn
 */
AVT_API void avt_clear_down (void);

/*
 * clears from cursor position up the viewport
 * if there is no balloon yet, it is drawn
 */
AVT_API void avt_clear_up (void);

/*
 * clear end of line
 * depending on text direction
 */
AVT_API void avt_clear_eol (void);

/*
 * clear beginning of line
 * depending on text direction
 */
AVT_API void avt_clear_bol (void);

/* clear line */
AVT_API void avt_clear_line (void);

/*
 * forward one character position
 * ie. print a space
 */
AVT_API int avt_forward (void);

/* delete last character */
AVT_API void avt_backspace (void);

/* insert spaces at current position (move rest of line) */
AVT_API void avt_insert_spaces (int num);

/* delete num characters at current position (move rest of line) */
AVT_API void avt_delete_characters (int num);

/*
 * erase num characters from current position
 * don't move the cursor or the rest of the line
 */
```

```

*/
AVT_API void avt_erase_characters (int num);

/* go to next or last tab stop */
AVT_API void avt_next_tab (void);
AVT_API void avt_last_tab (void);

/* reset tab stops to every eighth column */
AVT_API void avt_reset_tab_stops (void);

/* clear all tab stops */
AVT_API void avt_clear_tab_stops (void);

/* set or clear a tab in position x */
AVT_API void avt_set_tab (int x, avt_bool_t onoff);

/*
 * delete num lines, starting from line
 * the rest ist scrolled up
 */
AVT_API void avt_delete_lines (int line, int num);

/*
 * insert num lines, starting at line
 * the rest ist scrolled down
 */
AVT_API void avt_insert_lines (int line, int num);

/*
 * lock or unlock updates of the balloon-content
 * can be used for speedups
 * when set to AVT_FALSE, the textarea gets updated
 * when set to AVT_TRUE, the text_delay is set to 0
 * use with care!
 */
AVT_API void avt_lock_updates (avt_bool_t lock);

/*****
/* showing images without the avatar */
/* you should call avt_wait or avt_waitkey thereafter */

/*
 * X-Pixmaps (XPM), X Bitmaps (XBM) and uncompressed BMP are always supported
 * other image formats are supported with SDL_image
 */

/*
 * load image file and show it
 * on error it returns AVT_ERROR without changing the status
 */
AVT_API int avt_show_image_file (const char *file);

```

```

/*
 * load image from stream and show it
 * on error it returns AVT_ERROR without changing the status
 */
AVT_API int avt_show_image_stream (avt_stream *stream);

/*
 * show image from image data
 * on error it returns AVT_ERROR without changing the status
 */
AVT_API int avt_show_image_data (void *img, int imgsize);

/*
 * show image from XPM data
 * on error it returns AVT_ERROR without changing the status
 */
AVT_API int avt_show_image_xpm (char **xpm);

/*
 * show image from XBM data with a given color
 * the background is transparent
 * on error it returns AVT_ERROR without changing the status
 */
AVT_API int avt_show_image_xbm (const unsigned char *bits,
                               int width, int height,
                               const char *colorname);

/*
 * show gimp image
 * on error it returns AVT_ERROR without changing the status
 */
AVT_API int avt_show_gimp_image (void *gimp_image);

/*****
/* high-level functions */

/* wait for a keypress while displaying a button */
AVT_API int avt_wait_button (void);

/*
 * show positive or negative buttons
 * keys for positive: + 1 Enter
 * keys for negative: - 0 Backspace
 *
 * returns the result as boolean
 * on error or quit request AVT_FALSE is returned and the status is set
 * you should check the status with avt_get_status()
 */
AVT_API avt_bool_t avt_decide (void);

/*
 * navigation bar

```

```

*
* buttons is a string with the following characters
* 'l': left
* 'r': right (play)
* 'd': down
* 'u': up
* 'x': cancel
* 'f': (fast)forward
* 'b': (fast)backward
* 'p': pause
* 's': stop
* 'e': eject
* '*': circle (record)
* '+': plus (add)
* '-': minus (remove)
* '?': help
* ' ': spacer (no button)
*
* Pressing a key with one of those characters selects it.
* For the directions you can also use the arrow keys,
* The [Pause] key returns 'p'.
* The [Help] key or [F1] return '?'.
*
* the function returns the letter for the selected option
* or AVT_ERROR or AVT_QUIT
*
* example:
*   r = avt_navigate ("lxr");
*   if (r < 32) exit (0);
*   select (r) ...
*/
AVT_API int avt_navigate (const char *buttons);

/*
* avt_choice - use for menus
* result:      result code, first item is 1
* start_line:  line, where choice begins
* items:      number of items/lines
* key:        first key, like '1' or 'a', 0 for no keys
* back, forward: whether first/last entry is a back/forward function
*
* returns AVT_ERROR and sets _avt_STATUS when it cannot get enough memory
*/
AVT_API int
avt_choice (int *result, int start_line, int items, int key,
            avt_bool_t back, avt_bool_t forward);

/*
* show longer text with a text-viewer application
* if len is 0, assume 0-terminated string
* startline is only used, when it is greater than 1
*/

```

```

AVT_API int avt_pager (const wchar_t *txt, int len, int startline);
AVT_API int avt_pager_mb (const char *txt, int len, int startline);

/* show final credits */
AVT_API int avt_credits (const wchar_t *text, avt_bool_t centered);
AVT_API int avt_credits_mb (const char *text, avt_bool_t centered);

/*****
/* plumbing */

/*
 * reserve single keys (Esc, F11)
 * use this with avt_register_keyhandler
 */
AVT_API void avt_reserve_single_keys (avt_bool_t onoff);

/*
 * type for keyhandler
 * see avt_register_keyhandler
 */
typedef void (*avt_keyhandler) (int sym, int mod, int unicode);

/* register an external keyhandler */
AVT_API void avt_register_keyhandler (avt_keyhandler handler);

/*
 * type for mousehandler
 * see avt_register_mousehandler
 */
typedef void (*avt_mousehandler) (int button, avt_bool_t pressed,
                                   int x, int y);

/* register an external mousehandler
 *
 * it is only called, when a mouse-button is pressed or released
 * The coordinates are the character positions if it's inside of
 * the balloon or -1, -1 otherwise.
 */
AVT_API void avt_register_mousehandler (avt_mousehandler handler);

/*****
/* audio stuff */

/* must be called AFTER avt_initialize! */
AVT_API int avt_initialize_audio (void);

/*
 * supported audio formats:
 * AU: linear PCM with up to 32Bit, mu-law, A-law
 * WAV: linear PCM with up to 16Bit, MS-ADPCM, IMA-ADPCM
 * Both: mono or stereo

```

```

*
* the current implementation can only play sounds with
* up to 16Bit precision, but AU-files with more Bits can
* be read.
*/

/*
* loads an audio file in AU or Wave format
* not for headerless formats
*/
AVT_API avt_audio_t *avt_load_audio_file (const char *filename);

/*
* loads audio in AU or Wave format from a stream
* not for headerless formats
*/
AVT_API avt_audio_t *avt_load_audio_stream (avt_stream *stream);

/*
* loads audio in AU or Wave format from memory
* must still be freed with avt_free_audio!
*/
AVT_API avt_audio_t *avt_load_audio_data (void *data, int datasize);

/* values for audio_type */
#define AVT_AUDIO_UNKNOWN 0 /* doesn't play */
#define AVT_AUDIO_U8 1 /* unsigned 8 Bit */
#define AVT_AUDIO_S8 2 /* signed 8 Bit */
#define AVT_AUDIO_U16LE 3 /* unsigned 16 Bit little endian */
#define AVT_AUDIO_U16BE 4 /* unsigned 16 Bit big endian */
#define AVT_AUDIO_U16SYS 5 /* unsigned 16 Bit system's endianness */
#define AVT_AUDIO_S16LE 6 /* signed 16 Bit little endian */
#define AVT_AUDIO_S16BE 7 /* signed 16 Bit big endian */
#define AVT_AUDIO_S16SYS 8 /* signed 16 Bit system's endianness */
#define AVT_AUDIO_MULAW 100 /* 8 Bit mu-law (u-law) */
#define AVT_AUDIO_ALAW 101 /* 8 Bit A-Law */

/* for channels */
#define AVT_AUDIO_MONO 1
#define AVT_AUDIO_STEREO 2

/*
* loads raw audio data from memory
* the data buffer is copied and can be freed immediately
*
* audio_type is one of the AVT_AUDIO_* constants
* channels is AVT_MONO or AVT_STEREO
*
* must be freed with avt_free_audio!
*/
AVT_API avt_audio_t *avt_load_raw_audio_data (void *data, int data_size,
int samplingrate, int audio_type, int channels);

```

```

/*
 * frees memory of a loaded sound
 */
AVT_API void avt_free_audio (avt_audio_t *snd);

/*
 * plays a sound
 */
AVT_API int avt_play_audio (avt_audio_t *snd, avt_bool_t doloop);

/*
 * wait until the sound ends
 * this stops a loop, but still plays to the end of the sound
 */
AVT_API int avt_wait_audio_end (void);

/* stops audio immediately */
AVT_API void avt_stop_audio (void);

/*****
 * deprecated functions - only for backward comatibility */
/* don't use them for new programs! */

/* macro for marking deprecated functions in this header */
#if defined (__GNUC__) && ! defined (_AVT_NO_DEPRECATED)
# define AVT_DEPRECATED __attribute__ ((__deprecated__))
#else
# define AVT_DEPRECATED
#endif /* __GNUC__ */

AVT_API void avt_set_delays (int text, int flip_page) AVT_DEPRECATED;
AVT_API void avt_stop_on_esc (avt_bool_t on) AVT_DEPRECATED;
AVT_API int avt_wait_key (const wchar_t *message) AVT_DEPRECATED;
AVT_API int avt_wait_key_mb (char *message) AVT_DEPRECATED;
AVT_API void avt_quit_audio (void) AVT_DEPRECATED;
AVT_API avt_audio_t *avt_load_wave_file (const char *file) AVT_DEPRECATED;
AVT_API avt_audio_t *avt_load_wave_data (void *data, int datasize)
    AVT_DEPRECATED;

AVT_API int avt_menu (wchar_t *ch, int menu_start, int menu_end,
                    wchar_t start_code, avt_bool_t back,
                    avt_bool_t forward) AVT_DEPRECATED;

AVT_API int
avt_get_menu (wchar_t *ch, int menu_start, int menu_end, wchar_t start_code)
AVT_DEPRECATED;
AVT_API avt_image_t *avt_import_XPM (char **xpm) AVT_DEPRECATED;
AVT_API int avt_show_image_XPM (char **xpm) AVT_DEPRECATED;

AVT_END_DECLS

```

```
#endif /* _akfavatar_h */
```

## Appendix C Names of colors

Color names are case-insensitive. Names with a space in between can sometimes not be used. But you can always write them together in one word.

Other ways to define colors are described in [Chapter 5 \[Formats\]](#), page 17.

The following color names are recognized:

snow, ghost white, GhostWhite, white smoke, WhiteSmoke, gainsboro, floral white, FloralWhite, old lace, OldLace, linen, antique white, AntiqueWhite, papaya whip, PapayaWhip, blanched almond, BlanchedAlmond, bisque, peach puff, PeachPuff, navajo white, NavajoWhite, moccasin, cornsilk, ivory, lemon chiffon, LemonChiffon, seashell, honeydew, mint cream, MintCream, azure, alice blue, AliceBlue, lavender, lavender blush, LavenderBlush, misty rose, MistyRose, white, black, dark slate gray, DarkSlateGray, dark slate grey, DarkSlateGrey, dim gray, DimGray, dim grey, DimGrey, slate gray, SlateGray, slate grey, SlateGrey, light slate gray, LightSlateGray, light slate grey, LightSlateGrey, gray, grey, light grey, LightGrey, light gray, LightGray, midnight blue, MidnightBlue, navy, navy blue, NavyBlue, cornflower blue, CornflowerBlue, dark slate blue, DarkSlateBlue, slate blue, SlateBlue, medium slate blue, MediumSlateBlue, light slate blue, LightSlateBlue, medium blue, MediumBlue, royal blue, RoyalBlue, blue, dodger blue, DodgerBlue, deep sky blue, DeepSkyBlue, sky blue, SkyBlue, light sky blue, LightSkyBlue, steel blue, SteelBlue, light steel blue, LightSteelBlue, light blue, LightBlue, powder blue, PowderBlue, pale turquoise, PaleTurquoise, dark turquoise, DarkTurquoise, medium turquoise, MediumTurquoise, turquoise, cyan, light cyan, LightCyan, cadet blue, CadetBlue, medium aquamarine, MediumAquamarine, aquamarine, dark green, DarkGreen, dark olive green, DarkOliveGreen, dark sea green, DarkSeaGreen, sea green, SeaGreen, medium sea green, MediumSeaGreen, light sea green, LightSeaGreen, pale green, PaleGreen, spring green, SpringGreen, lawn green, LawnGreen, green, chartreuse, medium spring green, MediumSpringGreen, green yellow, GreenYellow, lime green, LimeGreen, yellow green, YellowGreen, forest green, ForestGreen, olive drab, OliveDrab, dark khaki, DarkKhaki, khaki, pale goldenrod, PaleGoldenrod, light goldenrod yellow, LightGoldenrodYellow, light yellow, LightYellow, yellow, gold, light goldenrod, LightGoldenrod, goldenrod, dark goldenrod, DarkGoldenrod, rosy brown, RosyBrown, indian red, IndianRed, saddle brown, SaddleBrown, sienna, peru, burlywood, beige, wheat, sandy brown, SandyBrown, tan, chocolate, firebrick, brown, dark salmon, DarkSalmon, salmon, light salmon, LightSalmon, orange, dark orange, DarkOrange, coral, light coral, LightCoral, tomato, orange red, OrangeRed, red, hot pink, HotPink, deep pink, DeepPink, pink, light pink, LightPink, pale violet red, PaleVioletRed, maroon, medium violet red, MediumVioletRed, violet red, VioletRed, magenta, violet, plum, orchid, medium orchid, MediumOrchid, dark orchid, DarkOrchid, dark violet, DarkViolet, blue violet, BlueViolet, purple, medium purple, MediumPurple, thistle, snow1, snow2, snow3, snow4, seashell1, seashell2, seashell3, seashell4, AntiqueWhite1, AntiqueWhite2, AntiqueWhite3, AntiqueWhite4, bisque1, bisque2, bisque3, bisque4, PeachPuff1, PeachPuff2, PeachPuff3, PeachPuff4, NavajoWhite1, NavajoWhite2, NavajoWhite3, NavajoWhite4, LemonChiffon1, LemonChiffon2, LemonChiffon3, LemonChiffon4, cornsilk1, cornsilk2, cornsilk3, cornsilk4, ivory1, ivory2, ivory3,

ivory4, honeydew1, honeydew2, honeydew3, honeydew4, LavenderBlush1, LavenderBlush2, LavenderBlush3, LavenderBlush4, MistyRose1, MistyRose2, MistyRose3, MistyRose4, azure1, azure2, azure3, azure4, SlateBlue1, SlateBlue2, SlateBlue3, SlateBlue4, RoyalBlue1, RoyalBlue2, RoyalBlue3, RoyalBlue4, blue1, blue2, blue3, blue4, DodgerBlue1, DodgerBlue2, DodgerBlue3, DodgerBlue4, SteelBlue1, SteelBlue2, SteelBlue3, SteelBlue4, DeepSkyBlue1, DeepSkyBlue2, DeepSkyBlue3, DeepSkyBlue4, SkyBlue1, SkyBlue2, SkyBlue3, SkyBlue4, LightSkyBlue1, LightSkyBlue2, LightSkyBlue3, LightSkyBlue4, SlateGray1, SlateGray2, SlateGray3, SlateGray4, LightSteelBlue1, LightSteelBlue2, LightSteelBlue3, LightSteelBlue4, LightBlue1, LightBlue2, LightBlue3, LightBlue4, LightCyan1, LightCyan2, LightCyan3, LightCyan4, PaleTurquoise1, PaleTurquoise2, PaleTurquoise3, PaleTurquoise4, CadetBlue1, CadetBlue2, CadetBlue3, CadetBlue4, turquoise1, turquoise2, turquoise3, turquoise4, cyan1, cyan2, cyan3, cyan4, DarkSlateGray1, DarkSlateGray2, DarkSlateGray3, DarkSlateGray4, aquamarine1, aquamarine2, aquamarine3, aquamarine4, DarkSeaGreen1, DarkSeaGreen2, DarkSeaGreen3, DarkSeaGreen4, SeaGreen1, SeaGreen2, SeaGreen3, SeaGreen4, PaleGreen1, PaleGreen2, PaleGreen3, PaleGreen4, SpringGreen1, SpringGreen2, SpringGreen3, SpringGreen4, green1, green2, green3, green4, chartreuse1, chartreuse2, chartreuse3, chartreuse4, OliveDrab1, OliveDrab2, OliveDrab3, OliveDrab4, DarkOliveGreen1, DarkOliveGreen2, DarkOliveGreen3, DarkOliveGreen4, khaki1, khaki2, khaki3, khaki4, LightGoldenrod1, LightGoldenrod2, LightGoldenrod3, LightGoldenrod4, LightYellow1, LightYellow2, LightYellow3, LightYellow4, yellow1, yellow2, yellow3, yellow4, gold1, gold2, gold3, gold4, goldenrod1, goldenrod2, goldenrod3, goldenrod4, DarkGoldenrod1, DarkGoldenrod2, DarkGoldenrod3, DarkGoldenrod4, RosyBrown1, RosyBrown2, RosyBrown3, RosyBrown4, IndianRed1, IndianRed2, IndianRed3, IndianRed4, sienna1, sienna2, sienna3, sienna4, burlywood1, burlywood2, burlywood3, burlywood4, wheat1, wheat2, wheat3, wheat4, tan1, tan2, tan3, tan4, chocolate1, chocolate2, chocolate3, chocolate4, firebrick1, firebrick2, firebrick3, firebrick4, brown1, brown2, brown3, brown4, salmon1, salmon2, salmon3, salmon4, LightSalmon1, LightSalmon2, LightSalmon3, LightSalmon4, orange1, orange2, orange3, orange4, DarkOrange1, DarkOrange2, DarkOrange3, DarkOrange4, coral1, coral2, coral3, coral4, tomato1, tomato2, tomato3, tomato4, OrangeRed1, OrangeRed2, OrangeRed3, OrangeRed4, red1, red2, red3, red4, DebianRed, DeepPink1, DeepPink2, DeepPink3, DeepPink4, HotPink1, HotPink2, HotPink3, HotPink4, pink1, pink2, pink3, pink4, LightPink1, LightPink2, LightPink3, LightPink4, PaleVioletRed1, PaleVioletRed2, PaleVioletRed3, PaleVioletRed4, maroon1, maroon2, maroon3, maroon4, VioletRed1, VioletRed2, VioletRed3, VioletRed4, magenta1, magenta2, magenta3, magenta4, orchid1, orchid2, orchid3, orchid4, plum1, plum2, plum3, plum4, MediumOrchid1, MediumOrchid2, MediumOrchid3, MediumOrchid4, DarkOrchid1, DarkOrchid2, DarkOrchid3, DarkOrchid4, purple1, purple2, purple3, purple4, MediumPurple1, MediumPurple2, MediumPurple3, MediumPurple4, thistle1, thistle2, thistle3, thistle4, gray0, grey0, gray1, grey1, gray2, grey2, gray3, grey3, gray4, grey4, gray5, grey5, gray6, grey6, gray7, grey7, gray8, grey8, gray9, grey9, gray10, grey10, gray11, grey11, gray12, grey12, gray13, grey13, gray14, grey14, gray15, grey15, gray16, grey16, gray17, grey17, gray18, grey18, gray19, grey19, gray20, grey20, gray21, grey21, gray22, grey22, gray23, grey23, gray24, grey24, gray25, grey25, gray26, grey26, gray27, grey27,

gray28, grey28, gray29, grey29, gray30, grey30, gray31, grey31, gray32, grey32, gray33, grey33, gray34, grey34, gray35, grey35, gray36, grey36, gray37, grey37, gray38, grey38, gray39, grey39, gray40, grey40, gray41, grey41, gray42, grey42, gray43, grey43, gray44, grey44, gray45, grey45, gray46, grey46, gray47, grey47, gray48, grey48, gray49, grey49, gray50, grey50, gray51, grey51, gray52, grey52, gray53, grey53, gray54, grey54, gray55, grey55, gray56, grey56, gray57, grey57, gray58, grey58, gray59, grey59, gray60, grey60, gray61, grey61, gray62, grey62, gray63, grey63, gray64, grey64, gray65, grey65, gray66, grey66, gray67, grey67, gray68, grey68, gray69, grey69, gray70, grey70, gray71, grey71, gray72, grey72, gray73, grey73, gray74, grey74, gray75, grey75, gray76, grey76, gray77, grey77, gray78, grey78, gray79, grey79, gray80, grey80, gray81, grey81, gray82, grey82, gray83, grey83, gray84, grey84, gray85, grey85, gray86, grey86, gray87, grey87, gray88, grey88, gray89, grey89, gray90, grey90, gray91, grey91, gray92, grey92, gray93, grey93, gray94, grey94, gray95, grey95, gray96, grey96, gray97, grey97, gray98, grey98, gray99, grey99, gray100, grey100, dark grey, DarkGrey, dark gray, DarkGray, dark blue, DarkBlue, dark cyan, DarkCyan, dark magenta, DarkMagenta, dark red, DarkRed, light green, LightGreen

# Index

## #

#! ..... 8

## A

archive-files ..... 14  
 audio files ..... 17  
 avatarsay ..... 6

## C

change the avatar ..... 8  
 color names ..... 47  
 colors ..... 17  
 command line tool ..... 6  
 commands for avatarsay ..... 10  
 configuration file for avatarsay ..... 10  
 console based programs ..... 6

## D

different avatar ..... 8

## E

exchange the avatar ..... 8  
 executable text ..... 8

## F

frontend ..... 6  
 fullscreen mode ..... 2

## G

GNOME ..... 15

## I

iconv ..... 4  
 images ..... 17  
 installation ..... 3

## K

keys ..... 2

## M

menu of `gnome-akfavatar` ..... 15

## N

names for colors ..... 47

## O

other commands, using `avatarsay` with ..... 13  
 overview ..... 1

## P

Pascal ..... 18  
 Pascal reference ..... 21  
 pausing (Pause key) ..... 2  
 pipes ..... 13  
 programming ..... 18

## R

reader ..... 7  
 recorded audio ..... 17  
 reference for C ..... 28  
 reference for Pascal ..... 21  
 requirements for `gnome-akfavatar` ..... 15

## S

sound files ..... 17  
 stopping ..... 2  
 stripline ..... 7  
 switching fullscreen/window ..... 2

## T

tearline ..... 7  
 text-terminal ..... 6  
 text-viewer ..... 7  
 toggle fullscreen/window ..... 2  
 transparency (avatar image) ..... 8  
 trouble-shooting ..... 4

## V

VGA ..... 4  
 viewer ..... 7

## W

window mode ..... 2  
 wrapper ..... 6