MagicMenu

Olaf Barthel

COLLABORATORS

	TITLE : MagicMenu	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Olaf Barthel	October 29, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MagicMenu

1.1 MagicMenu

```
                                  MagicMenu 2.18
        A utility to enhance the Intuition
               pull-down menu system

           Written by Martin Korndörfer
                        and
               Olaf 'Olsen'~Barthel

 Artwork created by Mario 'padrino' Cattaneo

      I.
Installation
                        II.
DANGERS
                        III.
The preferences editor
                        IV.
Keyboard control
                        V.
Frequently asked questions
                        VI.
History
                        VII.
Translations
                        VIII.
Authors & credits
                        IX.
Registration
                        X.
Support
```

1.2 installation

```
MagicMenu consists of two programs. Drop the "MagicMenu" program  ↩
    into the
```

"WBStartup" drawer of your system boot partition and put "MagicMenuPrefs" into the "Prefs" drawer on the same partition. Two versions of the program are provided. The "68k" versions will run on any Amiga machine, while the "020" versions will require an Amiga with an 68020 process or better. Take care, the "020" versions are bound to crash on machines they were not designed for.

If you do not have "gtlayout.library" installed already, you should copy it into the "Libs:" drawer. The preferences editor depends upon it.

Both programs can use configuration information as found in the icon tooltypes:

1.3 MagicMenu tooltypes

MagicMenu is a commodities tool. This means, it filters user input and reacts to it. It also sports a well-defined set of icon tooltypes:

CX_PRIORITY=<-128..127>

All commodities tools receive input events through a prioritized queue of handlers. This means, the higher the CX_PRIORITY value, the sooner a tool will receive an event.

CX_POPKEY=<hotkey>

You can invoke the MagicMenu preferences editor with a hotkey combination. The default is "control alt space", which means that you need to hold down the [Ctrl] and [Alt] key and then press the [Space] key to invoke the preferences program. For a complete description of the hotkey syntax, see your AmigaOS manual.

CX_POPUP=<YES or NO>

If you want MagicMenu to open the MagicMenu preferences editor immediately when it is started, set this tooltype value to "YES". Otherwise, leave it as "NO".

DONOTWAIT

The Workbench will process this tooltype when MagicMenu is started from the WBStartup drawer. It tells Workbench not to wait for MagicMenu to exit before launching the next program it finds in this drawer.

TOOLPRI=<-128..127>

This is another tooltype the Workbench will pay attention to. It defines the task priority MagicMenu will have assigned when it is launched.

STARTPRI=<-128..127>

This tooltype defines the order in which MagicMenu will be launched with the other tools in the WBStartup drawer. The higher this value, the sooner it will be started.

1.4 MagicMenuPrefs tooltypes

The MagicMenu preferences editor works like every other AmigaOS preferences editor you have ever used. It supports the following tooltypes:

EDIT

This brings up the preferences editor window, which is the default action for this program.

SAVE

The current MagicMenu settings will be saved permanently. If you select a MagicMenu settings file and then double-click on the preferences editor, it will read the file and save it permanently to disk.

USE

Reads the currently active MagicMenu settings file and tells MagicMenu to use these. You can also select a MagicMenu settings file and then double-click on the preferences editor.

PUBSCREEN=<public screen name>

The name of a public screen the preferences editor window is to open on. If no screen name is given or if the named screen cannot be found, the editor window will open on the default public screen.

CREATEICONS=<YES or NO>

If you wish to save settings files with the preferences editor, this tooltype controls whether the files will have icons attached (and thus, will be visible from Workbench) or not.

CX_PRIORITY=<-128..127>

All commodities tools receive input events through a prioritized queue of handlers. This means, the higher the CX_PRIORITY value, the sooner a tool will receive an event.

CX_POPKEY=<hotkey>

You can invoke the MagicMenu preferences editor with a hotkey combination. The default is "control alt space", which means that you need to hold down the [Ctrl] and [Alt] key and then press the [Space] key to invoke the preferences program. For a complete description of the hotkey syntax, see your AmigaOS manual.

CX_POPUP=<YES or NO>

If you want MagicMenu to open the MagicMenu preferences editor immediately when it is started, set this tooltype value to "YES". Otherwise, leave it as "NO".

1.5 dangers

MagicMenu exploits features of the Amiga operating system man was not meant to meddle with. Essentially, this means MagicMenu qualifies as a hack or kludge. You can quickly get into trouble when using it, while on the other hand it may work perfectly for you, with no problem whatsoever.

MagicMenu interacts with Intuition, the entity that runs the Amiga user interface and the pull-down menus. It intercepts normal menu rendering by looking for mouse or keyboard events that would normally get Intuition to do menu operations. The normal Intuition menus are replaced by what MagicMenu has to offer. The catch is that for Intuition menu rendering is something very special. For example, while the menus are displayed no window will open, no text on the screen will scroll and no window contents will be updated. This is because Intuition enters a special state when doing the menus in which all other display operations are delayed until the users lets go of the menu button.

Unfortunately, MagicMenu cannot do the same trick, which can lead to collisions. The machine may appear to lock up, the mouse will stop moving and you'll see every sign of a system crash. In such a case Intuition has tried to get a hold of the resources MagicMenu owns at the moment and is waiting for MagicMenu to release them. But MagicMenu is unable to release these resources until Intuition has stopped waiting for them to be released. The net effect is a classical deadlock.

MagicMenu will eventually back out of the deadlock after a certain amount of time but the deadlock can be avoided altogether if you configure MagicMenu to run in non-blocking mode (see Preferences/Non blocking for more information). However, the non-blocking mode has the effect that display updates by other programs that have windows open on the same screen will not be halted or delayed. Thus, if you are used to halt text scrolling by holding down the menu button, it will no longer work in non-blocking mode.

Some programs may interact strangely with the way MagicMenu renders and builds its menus. If you suspect foul play, hold down the [Ctrl] key or one of the [Alt] keys while you hold down the menu button: MagicMenu will get out of the way and let Intuition handle the pull-down menu business as usual.

1.6 The preferences editor

The preferences editor window consists of three pages of configuration information:

```
                Look and usage

                Keyboard control

                Colour control
                   Note: this page is available only under Kickstart 3.x and if  ↩
                       the screen
          the preferences editor window opens on has enough colour slots
          available to display the menu sample image (64 colours are the
          minimum requirements).

In addition to the configuration pages, there is set of pull-down menus
to choose from:


                Project

                Edit

                Settings
```

1.7 Look and usage

```
Pull-down menu:

   Usage

      Three options are available:

         - Intuition compatible

           The menus will appear when you hold down the menu button and
           will go away when you let go of the button.

         - Sticky mouse button

           The menus will appear when you hold down the menu button and
           will stay when you let go of the button. To make them go away
           press the menu button again.

         - Smart select

           The menus will appear when you hold down the menu button and
           will stay when you let go of the button. To pick menu items,
           use the select button (left mouse button). To make the menus
           go away press the menu button again.

   Look

      Three options are available:

         - Standard

           Menus are rendered in the familiar flat black/white style
           Intuition uses.
```

- Old 3D

  This menu rendering style is similar to the 3D button style the operating system uses. It also has custom imagery for the checkmark and the Amiga key.

- Multicolour 3D

  The "gaudy" style popularized by MagicWB. This has custom imagery for the seperator bars, the checkmark and the Amiga key. It makes use of a special set of configurable user interface colours.

Pop-Up menu:

Usage

Three options are available:

- Intuition compatible

  The menus will appear when you hold down the menu button and will go away when you let go of the button.

- Sticky mouse button

  The menus will appear when you hold down the menu button and will stay when you let go of the button. To make them go away press the menu button again.

- Smart select

  The menus will appear when you hold down the menu button and will stay when you let go of the button. To pick menu items, use the select button (left mouse button). To make the menus go away press the menu button again.

Look

Three options are available:

- Standard

  Menus are rendered in the familiar flat black/white style Intuition uses.

- Old 3D

  This menu rendering style is similar to the 3D button style the operating system uses. It also has custom imagery for the checkmark and the Amiga key.

- Multicolour 3D

  The "gaudy" style popularized by MagicWB. This has custom imagery for the seperator bars, the checkmark

and the Amiga key. It makes use of a special set of configurable user interface colours.

Centre boxes

If this switch is enabled, MagicMenu will try to centre the menu title, menu and submenu boxes right below the mouse pointer.

General:

Type

Three options are available:

- Pull-down menu only

  MagicMenu will stick to the original style of rendering menus from the top of the screen downwards. The menus may look different, but the placement of the items will be just like Intuition would choose it.

- Pop-up menu only

  Every time you invoke the menus, they will pop up right below the mouse pointer.

- Mouse pointer position dependent

  If the mouse pointer is in the screen title bar, MagicMenu will display a pull-down menu. If the mouse pointer is in a different position, it will display pop-up menus instead.

Mark submenus

Menus that have submenus attached will receive a marker at the right hand side of the menu box (unless the items are already marked as such).

Double borders

This affects the menu rendering style in the old and multicolour 3D modes: the rendering style of the menu borders will be changed and the item highlighting style will be reversed.

Non blocking

This switch toggles between two fundamentally different modes of MagicMenu operation:

- If disabled, menu rendering takes place on a screen only MagicMenu has access to. Windows will not open, text will not scroll, windows will not be updated. This is most similar to the way standard Intuition menus are rendered. But that’s not the whole story. The big difference is that Intuition does not really know that MagicMenu has taken

control of the screen. This may cause Intuition to fall into a trap when it badly wants to take control of the screen as well. In that case the Amiga will appear to lock up, the mouse will cease moving and the whole thing will have the uncanny appearance of a machine that has not only crashed, but joined the choir invisible. This can be a little unnerving at times, especially if the machine does not only appear to be crashed, but really has joined its maker. You'll know for sure when MagicMenu backs out of this deadlock after a few seconds. If it doesn't, you'll have to reboot the machine. So there.

Another effect this mode has is that all menu rendering will go straight to the screen bitmap. So if a menu layout is a little "off", it will happily trash innocent memory when it is rendered.

Think again, if you want trouble, this is the right setting for you.

- If this switch is enabled, display updates will not be blocked until the menu closes. This has several advantages over the blocking mode:

  · Intuition can still take control of the screen when it wants to. There is close to no chance at all for Intuition to lock up in this mode.

  ·~Menu rendering is safe. Even weird, exotic and broken menu item layouts will work without trashing memory.

  The big disadvantage of this mode is that it does not block display updates.

Menus open delayed

Normally, menus and submenus open as soon as the mouse pointer moves across them. This can slow things down with complex menus. If you choose to have menus opening delayed, you will have to stop the mouse from moving for a tick before the menus will eventually unfold.

Draw frames around menu items

In 3D mode, the single menu entries are surrounded by a recessed 3D frame once they become "active". This can make the text a little hard to read. If you do not want the frames to appear, you can turn them off with this switch.

Cast drop shadows

You can tell MagicMenu to draw drop shadows below the menus it displays if you enable this switch. Please note that the shadows will appear only in 3D mode.

1.8 Keyboard control

This page holds the controls for the keyboard control option, which is what you get when you press a special key combination or [Right Amiga]+[Right Alt]. The menus will open, for you to select an item with the cursor keys.

Keyboard control enabled

This switch enables the keyboard control mode.

Move mouse pointer to menu bar

While the keyboard control mode is in effect, the mouse pointer will be repositioned at the lower right corner of the screen title bar.

Activate with [Right Amiga]+[Right Alt]

This switch enables the easiest keyboard shortcut. Whenever you use the standard Intuition mouse button "emulation" MagicMenu will take over and go into keyboard control mode.

Keyboard shortcut

This is another shortcut to get MagicMenu into keyboard control mode. The default key combination is "ramiga space", which means that you will have to hold down the [Right Amiga] key and then press the [Space] key to get into keyboard control mode.

1.9 Colour control

Note: this page is available only under Kickstart 3.x and if the screen the preferences editor window opens on has enough colour slots available to display the menu sample image (64 colours are the minimum requirements).

On this page you can configure the colours MagicMenu will use when rendering the multicolour 3D look menus.

On the left hand side of the window you can see a menu sample with a button below it. To select a menu colour to edit, either pick it with this button or click the mouse inside the menu sample image (the editor will then use the colour of the pixel you have clicked on).

On the right hand side of the window you find will a vertical slider, a colour wheel and three Red/Green/Blue or Hue/Saturation/Brightness sliders below it. This is where you change the colour you have selected.

Colour matching precision

When picking the colours to use for menu rendering from the set available to a screen MagicMenu may not get exactly the colours it was asking for but whatever the operating system can provide and comes close to what

was requested. How much error MagicMenu will tolerate can be controlled with this slider, which can be set to four positions: "Exact" is the most precise (it will accept no error) and "GUI" is the least precise.

Prefer screen colours

Normally, MagicMenu will use the colours you have selected above when choosing how the menus should look like. But you can also have MagicMenu use the current user interface colours of the screen instead. To do so, enable this switch.

1.10 Project

Open...

This where you select a settings file to load into the editor.

Save as...

With this menu item you can store the currently loaded settings in a separate file.

About...

Displays information about the authors and the program.

Quit

This terminates the program.

1.11 Edit

Reset to defaults

This will reset the current program settings to "factory defaults".

Last saved

The editor will try to retrieve the last saved settings file. If this is not possible, the current settings will be left untouched.

Restore

This will restore the settings that were in effect when the editor was started.

1.12 Settings

Create icons?

This menu controls whether settings files saved with this editor will have icons attached (and thus, will be visible from Workbench) or not.

Slider color model

The sliders below the colour wheel on the Colour control page can operate either in Red/Green/Blue mode or in Hue/Saturation/Brightness mode. This is where you choose which mode to use.

1.13 Keyboard control

Menus are normally entirely mouse-driven, but there are few things you can do by keyboard:

Activate the menu

You can activate the menu and bring MagicMenu into keyboard control mode by holding down a special combination of keys. The default key combinations are [Right Amiga]+[Space] and [Right Amiga]+[Right Alt]. The menu will open and allow you to select the menu item you want with the cursor keys. Press [Return] to select an item or to open a submenu. Press [Esc] to return to the previous menu hierarchy or to close the menu altogether.

Use the normal Intuition menus

You can tell MagicMenu to get out of the way and let Intuition take care of the menus by holding down either [Alt] or [Ctrl] key when you press the menu button.

Change the look of the menus

While the menu is active, press [Ctrl] to toggle between 3D mode and the flat standard mode.

1.14 Frequently asked questions

MagicMenu and the preferences editor may not always do what you expect them to. Here is a short list of such situations and an explanation of what has happened:

I cannot configure the multicolour menu colours
The preferences program needs at least 24 colours to display the settings page that controls the multicolour menu. As the operating system will always allocate 11 colours for itself, this means you will need to run the preferences program on a screen with at least 64 colours available.

The multicolour menu never shows up, I always get the old 3D look menu
For the multicolour menu to work, MagicMenu needs to allocate the colours that make up its display from the set the screen has to offer it is to open on. Not all screens allow for this to work, as they may not have unallocated pens to work with or a colour palette which already has the correct colours in the right places. In these cases MagicMenu will fail to get the colours it needs and step back to the old style 3D menu. MagicMenu will also refuse to use the multicolour menu if the screen to use is a low resolution screen.

Sometimes I don't get any menu to show up at all
MagicMenu can run out of memory trying to display the menu and it can also fail to display a menu due to not enough room available on the screen. In such cases MagicMenu will have stopped building the menu for display before it has a chance to back out and hand over the job to Intuition. Sorry, that's all it can do then.

It's also possible that MagicMenu backs out of displaying a menu in order to avoid a fatal system deadlock. This may happen so quickly you won't even notice that the menus came up after all.

In both situations, nothing should be keeping you from retrying to bring up the menus again. It should eventually work.

The menu changes the screen colours when it appears
Some applications open screens which allow other software to allocate colours from the palette. But not all these applications seem to know that some of these colours may be allocated and changed by software like MagicMenu. The MultiView Workbench utility is such an application. MagicMenu will end up changing the screen colour palette since the screen parameters permit this. Other applications may make the same mistake. The only way to avoid this is to put MagicMenu into the old 3D colour look.

Images and the text frames overlap the menu item text
The program to create the menus also determines how they are placed. The items need not be placed vertically, below one another, but may be placed in any order there is. MagicMenu cannot just choose to move the items around to make room without running the risk of covering other items and really screwing things up. Thus, the text frames can be only as large as the menu item sizes permit.

MagicMenu conflicts with another utility that maps a certain feature to the menu button
Provided that the "other" utility is a commodities tool just like MagicMenu you can use the CX_PRIORITY=<number> tool type to select the order in which input events are processed. If you want MagicMenu to take care of processing the menu button, select a lower priority, e.g. make "CX_PRIORITY=-1".

Menu operations become terribly sluggish on 15/16/24 bit screens
Well? That's supposed to happen. You might be able to cut the time by turning off the Non blocking option.

MagicMenu fails with "Error initializing system patches".
There are three reasons why MagicMenu can fail with this error:

- It ran out of memory when preparing to install the patches

- MagicMenu is already running on that machine (unlikely; execution would have taken an entirely different path as this condition is

```
  detected a lot earlier during program startup).

- You had an older MagicMenu version running on your machine before
  you tried to start the new version.

The last reason is the most likely. Unless you do something
very strange in your Workbench startup procedure the problem should
go away after replacing the old version with the new one and
rebooting your machine.

The menus don't show up at all, the Amiga just freezes and
crashes when the right mouse button is pressed.
Take care if you have several utilities running that plaster the
operating system with patches. For some of the patches the order
in which they are installed matters greatly. Whether MagicMenu
is the first program to install its patches or the last one can
make the decisive difference between a crash and a working system.

(to be continued)
```

1.15 history

```
MagicMenu versions 0.01 through 2.3 were developed by Martin Korndörfer. With
V2.4 development passed into the hands of Olaf 'Olsen' Barthel.

(Most recent change comes last)

MagicMenu 2.4 (beta)
====================

- Improved overall stability under CyberGraphX; the menu rendering code was
  actually peeking Screen->BitMap instead of Screen->RastPort.BitMap, which
  among other things could screw up rendering in deep display modes. Menu
  rendering now works consistently in display depths beyond 8 bit.

- Removed references to SetABPenDrMd() where they were in the wrong place.

- The code that sends the IntuiMessage should be less timing critical and
  simpler now. It fires off the message and starts to wait for a response
  if necessary rather than allocating timer requests, message ports, etc.
  on the spot and then jumping into action.

- Removed the routine that complains about timed out IntuiMessages. It's
  nice to know that something went wrong, but not that helpful if you
  cannot do anything about it.

- Since IntuiMessage processing works differently now, MagicMenu now checks
  for pending, unreplied messages.

- The routine that displays the menu and handles the IDCMP_MENUVERIFY
  case should be better protected against Intuition state changes and
  stale pointers. There is still a catch in that the Window to receive
  the menu event may close or dispose of its menu strip before the
  event arrives. But actually, Intuition protects itself against
  such eventualities, anyway.
```

- For a Window without a UserPort, MagicMenu could lose memory. This
  has been fixed.

- When an IDCMP_MENUVERIFY message times out, the window to cancel
  it now properly receives its IDCMP_MOUSEBUTTONS/MENUUP and
  IDCMP_MENUPICK/MENUNULL messages (as per the RKMs). When menu
  rendering is finished the other inactive IDCMP_MENUVERIFY windows
  on the screen now receive the proper IDCMP_MOUSEBUTTONS/MENUUP
  messages.

- Changed the way BOOPSI images in menus are handled. MagicMenu no longer
  tries to remap them as if they were BitMap images. Right now it turns
  them into BitMap images and then remaps them.

- Redid the image remapping code. It no longer fiddles with the raw BitMap
  data but uses the Blitter.

- Added a patch for ObtainGIRPort() which should make all the SafeGIRPort
  patches redundant that existed just in support of MagicMenu. The patch
  checks if the screen it should work upon is currently locked by
  MagicMenu. If so, ObtainGIRPort() will return NULL. This is a little
  dangerous since some built-in Intuition classes do not protect themselves
  against ObtainGIRPort() returning NULL. I can't make the patch tell
  Intuition to wait until the screen is ready again either, as this would
  eventually deadlock Intuition.

- Changed the new 3D look text highlight colour to be more consistent with
  the way gadget text highlighting works.

- MagicMenu no longer detaches from the Shell it was started from. If
  you need such a feature, use "run >nil: MagicMenu".

- MagicMenu now works with Final Writer. Final Writer calls
  Clear/ResetMenuStrip whilst in MENUVERIFY state. Previously,
  MagicMenu did not allow this to happen, causing the verify message
  to time out.

- The phony IntuiMessages are now ExtIntuiMessages with a NULL tablet
  data pointer and proper time stamp information.

MagicMenu 2.5 (beta)
====================

- The ObtainGIRPort() patch now handles the NULL parameter case
  gracefully. Some BOOPSI gadget dispatchers actually end up invoking
  the draw method in the OM_NEW case with the GInfo pointer not
  yet initialized.

MagicMenu 2.6 (beta)
====================

- The new 3D colour mode could screw up the screen colour palette.
  The code did not check for pen allocation failures and, since the
  pen variables were unsigned, the deallocation code would end up

freeing the same pen over and over again.

- Changed the commodities filter setup code. All the initializations are now done at program startup time. MagicMenu enables the filter later instead of building the filter when the menu comes up.

- Changed the way bevel boxes are rendered. Instead of calling Move..Draw..Draw over and over again it now uses RectFill() where possible. For vertical and horizontal lines this is actually faster than calling Move/Draw.

- Made small visual changes to the menu box and title bar rendering in new 3D look mode.

- With the non-blocking option enabled, the new 3D look menus now sport drop shadows.

- In non-blocking mode MagicMenu no longer uses SuperBitMap windows.

- Removed the "demo" menu from the prefs program.

- Integrated Mario's new imagery, including the new colour map that goes with it.

MagicMenu 2.7 (beta)
====================

- The drop shadows no longer cause windows to be moved away from the screen right and bottom edges.

- Disabled menus, items and subitems are no longer rendered with a frame in new 3D mode.

- The drop shadows were only transparent with CyberGraphX. I added another patch to allow this for any Amiga (and which magically also boosts layer creation speed).

- One WaitBlit() was missing in the remapped image cleanup code.

- Removed the "clipping" option. It is now enabled by default in order to avoid big trouble on the way.

- If MagicMenu cannot reuse the patch table installed by an older MagicMenu version that was removed from memory just before the new one was run, it now complains and exits without crashing.

- I tried to make MagicMenu safer by reducing the rendering options to two alternatives only: you either select non-blocking or you don't. Both options imply clipped drawing operations, making it more difficult to trash memory on the way. The "direct draw" option is implicitely enabled for non-blocking mode and implicitely disabled for blocking mode. All this will slow down menu operations a bit, but better a little slower than a little sooner to crash.

- No longer fakes ClipRects and uses sleight-of-hand tricks to get

```
  away with it. Instead of calling SwapBitsRastPortClipRect()
  it now uses a much simpler technique to exchange the on-screen
  BitMap data and the menu imagery. This technique neither has
  nor requires the side-effects SwapBitsRastPortClipRect() has.

- Removed the CyberGraphX chunky option; it is now enabled all
  the time.

- New and improved colour remapping code. Much faster than the
  old blitter based stuff.

- All new preferences program. The prefs editor goes into
  SYS:Prefs. The new configuration files go into ENVARC:.
  Note that the new configuration files are not compatible
  with the old ones.

- Prefs program and main program now support localization.
  No catalogs are available yet, sorry.

- Non-blocking operation is now the default mode since it
  is less likely to trash memory, freeze the machine or
  crash it.


MagicMenu 2.8 (beta)
====================

- Added two more patches for WindowToBack() and
  MoveWindowInFrontOf().

- The prefs program should now consistently find the current
  preferences settings, even if MagicMenu is not running.

- Removed the ghosted text colour options. These colours are
  now connected to the background colour.

- The prefs program now has a proper version string.

- The keyboard control hotkey combination can be changed at
  run-time now.

- Resetting the preferences to defaults now properly updates
  the display and the colours.

- The layer patch was using the wrong rectangle offsets,
  causing superbitmap windows to screw up.

- The prefs program and the main program now both have
  new default minimum stack size limits. For the prefs
  program it's 10K and 8K for the main program.

- The menu image remapping routines show now treat images
  properly which make use of the PlanePick/PlaneOnOff combo.

- The main program will now find the prefs program when
  it should do. Previously, it would only start the prefs
  program if the main program was run from Shell.
```

- The colour remapping now also takes place in selected state
  images. The same colour remapping rules are applied for the
  highlight colour that are used for the normal background
  colour.

- Added tablet input event processing. So far only one
  tablet input device is reported to work properly. If there
  are more, I would like to know :)

- Rewrote the central menu event processing loop. MagicMenu
  should now snap out of an Intuition deadlock much faster
  than before. I also removed the global menu timeout,
  which is now redundant.

- The menus no longer pop out of multicolour style back
  into old 3D style without warning.

- Starting the main program twice now brings up the
  prefs editor, just like with any other commodities tool.

- Made the drop shadow a little smaller (4x4 instead of 6x6).

- Fixed two fatal bugs in the bitmap initialization code.

- Added two more patches, this time for OffMenu() and OnMenu().

- MagicMenu now supports menu lending. For this to work, I had
  to remove the "screen with the active menu pops to front"
  feature. This screen depth arrangement also got into trouble
  with child screens, which would always get popped to the
  background.

- The prefs program now sports a "Test" button. If the main
  program is not yet running when you hit this button, the
  prefs program will try to launch it.

MagicMenu 2.9 (beta)
====================

- Moved the default imagery back into chip RAM.

- When in keyboard control mode, [Shift]+[Esc] will abort
  the entire menu operation, no matter which menu hierarchy
  you are in.

- The 3D multicolour mode now requires that the menu font is
  at least nine pixels tall. If it is smaller, you will get
  the old 3D mode.

- Reworked the menu imagery and made sure that the MX and
  checkmark images match in size.

- Multicolour ghosted text no longer gets rendered over and
  over again when moving the mouse across it.

- When running out of pens for the gradient fill slider
  the prefs editor should now back out gracefully rather than
  end up trying to load colour register -1 with greyscale
  data. This could have been the reason for the inexplicable
  prefs editor crashes.

MagicMenu 2.10 (beta)
=====================

- Default prefs project icon images now come from ENV:sys/def_pref.

- The multicolour 3D look mode no longer requires a font of a least
  nine pixels. Now eight pixels will do (and don't you complain if the
  imagery is taller than the menu font).

- The menu image remapping code should now be much smarter for images
  that use the PlanePick/PlaneOnOff option.

- Removed the "Remove" option from the prefs program.

- The prefs editor could copy too many colours when updating the
  the program settings.

- Added a "precision" slider to the prefs program. With this slider
  you can select how much error the colour allocation routine will
  tolerate when selecting the colours for the multicolour menu.

- Rewrote all the colour management routines to be as careful as
  possible when releasing the allocated pens. It looks as if not
  all system configurations will treat pen #-1 as a no-op.

MagicMenu 2.11 (beta)
=====================

- The preferences editor now opens a custom screen with 32 colours
  if it cannot get that many from the requested public screen.

- When MagicMenu has taken control of the menus, window depth
  arrangement calls are no longer ignored, but delayed until
  the menu closes. Actually, I would have to add a whole lot
  more patches to make operation halfway safe, but for now
  I'm just plastering patches onto routines which are somewhat
  likely to get triggered while the menus are up.

MagicMenu 2.12 (beta)
=====================

- The window depth arrangement patches were not installed correctly,
  causing them to do nothing in most cases.

- In non-blocking mode and when running under Kickstart 2.04 you
  would still get the shadow borders. As the drop shadows do not
  work under V37 this was not really sensible.

- Added Mario's new 4 colour images. Note that you will get these
  only with high resolution screens and fonts >= 11 points. I also
  changed the old 3D look ghosting style.

MagicMenu 2.13 (beta)
=====================

- Added a new option to the prefs program. If "Prefer screen colours"
  is enabled, MagicMenu will make up the menu colours from the screen
  user interface colours rather than using the colours you selected
  from the palette.

- Updated the WindowToFront/WindowToBack patches to delay window
  operations on the screen the MagicMenu menus are active on.

- Added more patches to SetWindowTitles and RefreshWindowFrame.
  I should also patch RefreshGList, RefreshGadget, NewModifyProp,
  ModifyProp, and about seven other routines but I guess I'd
  rather stop here. All these routines can cause deadlocks which
  MagicMenu will back out of safely. This can be a little annoying,
  but you can still change the MagicMenu operating mode for
  normal operation.

- The keyboard control commands now consistenly work when they are
  enabled and go out of the way when the are disabled.

- Changed the settings file format, this time hopefully for the last
  time.

- Added a new option to have menus open slightly delayed, or put
  another way, when you have stopped moving the mouse.

- Added another new option to turn off the frames drawn around the
  active menu item.

MagicMenu 2.14 (beta)
=====================

- Corrected the placement of the Command images in menu items and
  subitems.

- For the "old look" menus the separator bars now render in the
  correct colour.

- The old 3D look menu borders are now just one pixel thick,
  regardless of the display mode used.

- Changed the preferences data exchange interface between the
  prefs editor and the main program. Now the system should no
  longer crash if you try to change the settings of the new
  program with an old prefs editor.

- Drop shadows are now restricted to pop-up menus.

- Some menus will open faster now in non-blocking mode, as they

will be "promoted" to simple refresh windows if possible.

- Some of the patches MagicMenu installs to protect itself are no longer active in non-blocking mode. This will help Workbench and other applications which would otherwise get caught in their display update work.

- Changed the alignment rules for the checkmark, Amiga key and submenu arrow images to match those Intuition uses (or would use if it could).

- With "mark submenus" disabled, MagicMenu would cease to render submenu indicators in multicolour 3D mode. It now works again as it should.

MagicMenu 2.15
==============

- An invalid hotkey specification no longer keeps the main program from functioning. It will complain and the feature connected with the hotkey will be unavailable, but the program will continue to run.

- Changed the default colours for the preferences editor fallback screen.

- Added more patches for screen depth arrangement, opening and closing.

- Added Mario's new artwork.

MagicMenu 2.16 (internal)
=========================

- The size of the menu bar is now taken from the Screen data structure rather than made up from the font height.

MagicMenu 2.17 (internal)
=========================

- The layers.library patches now jump into action only if the non-blocking mode and the drop shadows are enabled.

MagicMenu 2.18
==============

- Removed all the layers.library patch code. The drop shadows are now rendered using a much more robust technique that does not require any black magic at all. As such, it now also works under Kickstart 2.04 and is no longer restricted to non-blocking mode (yes kids, that's what you always wanted).

```
MagicMenu 2.19
==============

- The changes in 2.18 introduced bugs into the menu refresh code.
  This is what I had to fix in this version.
```

1.16 translations

Very early during the big MagicMenu rework Ole Friis, coordinator of the Amiga Translators Organization, approached me and offered to help in getting the program text translated. In the "catalogs" drawer you will find the result of the collaboration. I am deeply indebted to all the people who spent their time translating the program text:

Czech:
Translator: Vit Sindlar, xsindl00@fik.dcse.fee.vutbr.cz
Proofreader: -

Danish:
Translator: Flemming Steffensen, fsteff@dannug.dk
Proofreader: Ole Friis, ole_f@post3.tele.dk

Dutch:
Translator: Leon Woestenberg
Proofreader: -

Finnish:
Translator: Marko Honkanen, marko.honkanen@mail.suomi.net
Proofreader: Ville Pispa, brainvpp@compost.fipnet.fi

French:
Translator: Jérôme Chesnot, jchesnot@pratique.fr
Proofreader: Vincent Oneto, voneto@instn.saclay.cea.fr

Greek:
Translator: Pantelis Kopelias, leestar@acropolis.net
Proofreader: Manos Konstantiniadis, konem@acropolis.net

Italian:
Translator: Francesco Mancuso (mcfrank@mediatel.it)
Proofreader: Vincenzo Gervasi (gervasi@di.unipi.it)

Polish:
Translator: Konrad Dubiel, konrad@inet.com.pl
Proofreader: Marcin Orîowski, carlos@inet.com.pl

Spanish:
Translator: Juan Antonio Ramirez, goliath@nether.net
Proofreader: Arturo Roa, aroa@redestb.es

Swedish:
Translator: Thomas Andersson, steiner@kd.qd.se
Proofreader: Magnus Holmgren, cmh@lls.se

1.17 authors

Martin Korndörfer wrote the original MagicMenu program. He is no longer involved in the development of the program.

The 3D multicolour menu artwork and design were created by Mario 'padrino' Cattaneo.
It was him who brought MagicMenu back to life after development had come to a standstill with the 1.29 version.

This documentation, the preferences editor and the development work following MagicMenu 2.3 were done by Olaf 'Olsen' Barthel.

1.18 registration

The original MagicMenu used to be freeware, you would use it and were not expected to do something in return for using it. This eventually changed with MagicMenu 2.13. We decided that as a MagicMenu user you should be encouraged to make a contribution. There are no usage restrictions, unnerving reminders in this program that force you to register with us, nor is there a special program version available you would receive upon registering. If you decide to make a contribution you will encourage us to continue development and to make enhancements to the program.

It's up to you to decide what you want to contribute and whom you want to send it to:

Martin Korndörfer created the original MagicMenu and developed all versions up to and including V2.3:

Martin Korndörfer
Pommernstraße 15
D-86916 Kaufering, Germany

Mario 'padrino' Cattaneo is responsible for the artwork used in the program since V2.0:

*** NEW >>>

Mario Cattaneo
Paul-Marien-Str. 6
66111 Saarbrücken, Germany

<<< NEW ***

Olaf Barthel picked up development with V2.4, he is responsible for all the changes, enhancements, bug fixes and new bugs in this version:

Olaf Barthel
Brabeckstraße 35
30559 Hannover, Germany

Any contribution makes sense, large or small.

1.19 support

You can find the most current version of MagicMenu either on Aminet or
on the official home page, as maintained by Mario 'padrino' Cattaneo:

   http://fsinfo.cs.uni-sb.de/~cattaneo/magicmenu/index.html

For feature requests or bug reports please contact <olsen@sourcery.han.de>.