RAVEN MENU FORMAT V2.01

BY RAVEN SOFTWARE

1.0 Overview

The scripted menus allow for easy manipulation of console variables and configuration files from within a GUI environment. They are interpreted as loaded and create their own layout. No compilation is necessary to change a menu or HUD. They sit on top of the console and have no direct access to any data members outside of their own classes. The page is laid out as a bunch of dirty rectangles. The system was designed to be as easy to use as HTML, but extended to take full advantage of the Quake style configuration files and console.

1.1 Useful Console Variables

ui_precache – if this cvar is set to 1, all the menu and HUD files are loaded once on game start up and never freed. While developing menus and HUDs, set this to 0 so that any changes made will be reflected without having to quit the game. **tip_millisecPerChar**, **tip_millisecPerLine** and **tip_millisecMin** can be used to tweak the duration of the tooltips that appear on screen.

2.0 Menu Basics

The first thing you should do is familiarise yourself with the console and its intricacies. The following documentation presumes a thorough knowledge of how the console works and how to get the most from it.

The only requirements for a menu file are –

- The extension is “rmf”
- The file resides in “base/menus” or a subfolder
- The file contains a “<stm>” to start and a “</stm>” to end.

A description of the internal layout of the code will help define some of the terminology used. There is a controlling class (menu system), which is created when the menus are displayed. This contains all global information required by the menu system and every interface required to fully utilise it. There can be only one menu system class at any one time.

Attached to the menu system class is a linked list of menus. Menus are defined as one complete set of frames. Each menu defines one complete menu screen (e.g. the video settings screen). A menu contains all the information global to that page and grabs anything more from the parent menu system. There can be as many menu pages attached to the menu system as memory will allow.

Each menu page has a linked list of frames attached to it. The frame starts out the full size of the menu page, and frames are cut from it. Frames are named and can be referenced directly. A frame does not necessarily have to have content. There can be as many frames as memory will allow. An example of a frame is the scrollable area in the video settings menu.

Each frame may or may not have a list of pages associated with it. One frame can have many pages attached to it, but the page on the top of the page stack is the only one

displayed. There can be an unlimited number of pages attached to any one frame, but, in the vast majority of cases, it is only one. Frames are the hardest thing to get a grasp on, and once that is done, the rest of the system should be fairly easy to understand.

Each page has a list of areas attached to it. These define the actual content of the whole page (text, models, images etc). Again, there is no limit to the number of areas that can be used as content. Extensive error checking is performed on each and every area as it is created and loaded. Should anything fail, a developer warning will be issued and the area will simply not appear.

There is a stack of all the above, created once and only destroyed when the menus are exited. So, if a frame or page is revisited, it is not reconstructed, just redisplayed. Pages are only constructed when a page is first visited.

All paths have a presumed base and extension. For example, all images used in the menu system reside in “base/gfx/menus/” directory and have the extension “jpg” or “png”.

If you examine the existing files you will see a great amount of text in the format of “&MENU_XXXX&”. Anything within “&” is directed through a string package for localisation. Unless you wish to translate into many different languages, please ignore this feature. These string packages are created and edited with StripEd.

There is a file “defines.rmf” which is always loaded whenever the menu system is active. This is used to precache any fonts and shaders that may be used, and to equate colours to names as well as defining constant values, which are used throughout other menu files.

3.0 HUD Basics

The only requirements for a HUD file are –

- The extension is “hud”
- The file resides in “base/huds” or a subfolder
- The file contains “<hud>” to start and a “</hud>” to end

HUDs are extremely similar to menus is most ways, the main conceptual difference between HUDs and menus is that HUDs do not have a strict hierarchy; HUDs can be forced as topmost or bottommost. Some menu features are not available in the HUDs and vice versa.

The HUD system uses defines.hud for its predefined constants and shaders.

4.0 Terminology

action – a command that is normally bound to a key and used in game e.g. +attack
align – either center, left, normal or right.
tint - a value representing a 32 bit RGBA color.
command – a command that can be issued directly from the game console e.g. disconnect

cvar – a console variable name e.g. r_fullbright
list - a list of items separated by commas e.g. “Armor,Flash Pak,Grenade”
text - a generic text string.
value - a number, normally an integer but can be positive or negative and sometimes a fraction. E.g. 2.4
menu - the name of an rmf or HUD file
frame - the name of a frame within a screen.
keyname - the name used to bind a key e.g. KP_DEL, SEMICOLON, A, B or C

5.0 Game Console Commands For Menus

The menus are activated through a series of commands issued from the console.

menu <menu> [frame]
This constructs a menu page or redisplays it if it has already been constructed. E.g. “menu main” brings up the main menu. If a frame name is specified, the page will be built in that frame.

intermission <menu>
Intermission is identical to the menu command, with the exception that the created page cannot be exited.

confirm <cvar> <config1> <config2>
If the cvar is non-zero execute config1, otherwise execute config2.

reloadall
Marks all pages in the stack to be reloaded when they are next visited. Used to change languages.

refresh
Refreshes the topmost page. For example, if a displayed cvar has changed by another process, you will have to refresh the screen to see it in the menu.

popmenu
Pops the most recent page off the top of the stack as long as there is a valid page behind it. If the most recent page was the only page on the stack, then the menu will only be popped if the game is active.

killmenu
Identical to popmenu, but always pops the page irrelevant of the validity of the page behind or the availability of the console.

return
Destroys the whole menu system provided there is an active game running in the background. i.e. returns to the game.

menuoff
Destroys the whole menu system including all pages currently on the stack.

select <cvar>
Used to select items. Used exclusively with the selection area.

checkpass <password> <pass menu> <fail menu>
Checks **password** against the parental lock password and displays the **pass menu** if they match or the **fail menu** if they don't.

changepass <oldpass> <newpass> <verify newpass> [pass menu] [fail menu]
If **oldpass** matches the parental lock password, and **newpass** is the same as **verify newpass**, then the parental lock password will be changed. If **pass menu** is set, then this will be displayed if everything was ok, and if **fail menu** is set, then this will be displayed if any error occurred.

freshgame
Starts a fresh game, bringing up a requester asking if you wish to exit the current game if you are in one.

ui_restart
Shuts down and then reloads every menu and HUD file from scratch

6.0 Game Console Commands For HUDs

hud <name> [frame]
Brings up a fullscreen HUD if frame is not defined, or bring up a HUD in the requested frame. If the requested HUD is already active then no action is taken.

rehud <name> [frame]
This is the same as HUD, but always reinitialises the existing HUD if it is already active.

pophud <name> [frame]
Destroys the requested HUD if it is anywhere in the HUD display list.

hudxfade <name> <time> [frame]
Cross-fades two HUDs together. If the frame is specified, it will be purely in that frame.

7.0 Control Word Details

One quick note about control commands; whenever optional keywords are given, they go inside the bracket. For example: <stm global>

<stm></stm>
This defines the start of data to be parsed in an rmf file. There must be a </stm> somewhere later in the file for the file to be valid. The </stm> defines the end of the data

to be parsed. Anything before the <stm> or after the </stm> will be ignored completely. There can only be one <stm> and one </stm> per rmf file.

Optional Keywords

nopush – this stm is not pushed onto the page stack therefore it cannot be revisited.
waitanim – the page waits for the Ghoul animation to finish before allowing any input. (e.g. on the animating main page)
window – designates the page as not being fullscreen
fragile – Any key press will instantly exit this screen. (E.g. On the startup animation)
global - disable the default scissoring to clip the contents of this page to its bounds.
command <command> – Execute a command (normally to set up data on the page) before creating the page. (E.g. the game statistics page)
resize <width> <height> - Resize the current frame to a new width and height and center on the screen. Holding down the right mouse button can move around a resized menu on it. (e.g. any requestor type menu).
complete <command> – the command to run when the scrolling completes or the text timeout hits.

<hud></hud>

This defines the start of data to be parsed in an HUD file. There must be a </hud> somewhere later in the file for the file to be valid. The </hud> defines the end of the data to be parsed. Anything before the <hud> or after the </hud> will be ignored completely. There can only be one <hud> and one </hud> per rmf file.

Optional Keywords

resize <width> <height> - Resize the current frame to a new width and height and center on the screen.
complete <command> - The command to add to the console buffer when the scrolling wraps once or the text hits a timeout.
window – Defines the HUD as non fullscreen
translucent – This makes HUDs underneath this one render first (makes this HUD an overlay)
topmost – Always force this HUD to the top (e.g. for subtitle printing)
backfill <cvar> <tint> - Uses color as the backfill and sets the cvar to that value
escapable <command> - The command to add to the console buffer when escape is pressed with this HUD up

<frame name width height>

Frames define the layout of a page. The “default” frame is the full size of the page (which may have been resized), and successive frames are cut from the default frame or the named frame. All frames are scaled to the resolution so the whole page fills the screen.

Optional Keywords

page <name> - Defines the stm or HUD file associated with this frame.
cpage <cvar> - The parser finds the contents of the cvar and uses that as the name of the page to load.
cut <name> - Defines the name of the frame from which this frame is cut.

border <width> <line width> <line tint> - This defines a border which fits inside the frame. Normally used to outline requesters and normally has no page attached to it.
perimeter <left> <top> <right> <bottom> - This defines a non uniform border with no line or backfill color
backfill <tint> - Defines the background color of the frame. Mainly used for frames with no page attached.

<key keyname command>
When keyname is pressed when this page is active, command will be added to the console exec buffer. The keynames are the same as the bind names (E.g. A, B, ENTER, KP_END etc.). There is also a special keyname “any” that refers to any key press.

E.g. <key any “popmenu”> will pop the current menu off the menu stack if any key is pressed.

<ikey action command>
When any key bound to action is pressed while the page is active, command will be added to the console exec buffer.

E.g. <ikey +attack “echo hello”>
If mouse1 and lctrl are bound to +attack, then when either of these keys are pressed, “hello” will be echoed to the console.

<ckey keyname cvar falsecommand truecommand>
When keyname is pressed anywhere on the screen, cvar is checked, if it is zero, then falsecommand is added to the command buffer, if it is non zero, then true command is added to the command buffer.

E.g. <ckey enter console "play sound/menus/invalid.wav" "menuoff">
If enter is pressed and the console is enabled, then the menus will all be turned off. If the console is disabled (zero), then the invalid sound will play.

<comment any text here>
Used to add comments to the rmf or HUD page. Completely ignored by everything.

E.g. <comment The quick brown fox jumped over the lazy dog>

<includecvar cvar>
Includes the text contained in the cvar directly as source. The cvar text can have control sequences and layout keywords in it.

E.g.
<set menu_text “ <font type title>Hello!<normal>”>
<includecvar menu_text>

<cincludecvar cvar1 cvar2>
As includecvar, but only includes the data of cvar2 if the contents of cvar1 are non-zero.

<cnincludecvar cvar1 cvar2>
As includecvar, but only includes the data of cvar2 if the contents of cvar1 are zero.

<exincludecvar cvar1 cvar2 cvar3>
As includecvar, but includes cvar2 if the contents of cvar1 are non zero, otherwise includes cvar3.

<einclude cvar1 value cvar2>
As includecvar, but only includes the data of cvar2 if the contents of cvar1 are exactly equal to value.

<include page>
Includes a different page of source into this page. This is a raw text inherit. An rmf extension is automatically added and the page must reside in the “base/menus” folder.

E.g. <include globe> will include globe.rmf into the current page always.

<cinclude cvar page>
Checks the value of cvar, and includes page if, and only if, the cvar is non zero.

E.g. <cinclude window_avail m_fullscreen_toggle> will include m_fullscreen_toggle.rmf into the current page if, and only if, the contents of the window_avail cvar are non-zero.

<cninclude cvar page>
Only includes page if cvar has a value of zero.

E.g. <cninclude no_won m_public_server> will include m_public_server.rmf if, and only if, no_won is zero.

<exinclude cvar page1 page2>
Checks the value of cvar and includes page1 if it is zero, otherwise it includes page2.

E.g. <exinclude mail_avail blank letter> will include letter.rmf if mail_avail is non-zero or blank.rmf if mail_avail is zero.

<einclude cvar value page>
Includes page if the cvar exactly equals value.

E.g. <einclude mapname pra2 pra2mission> will include menus/pra2mission.rmf if the mapname is pra2.

<alias command actions>
This works exactly like an alias on the console, it strings a list of commands together and gives them simple name.

E.g. <alias taunt “say Come get some loser!”>

<set cvar value>
Sets cvar to value immediately. It does not wait for the page to be loaded; it is evaluated when the command is parsed.

E.g. <set r_mode 3> will set the screen mode to 640x480

<cbuf config>
Executes a command on the console after the page is loaded.

E.g. <cbuf "exec default"> will exec default.cfg at the next opportune moment.

<config config>
This defines a config file that is run whenever the page is created or revisited.

E.g. <config video> will exec menus/video.cfg whenever the page is visited.

<exitcfg config>
This defines a config file to be run whenever a page is destroyed or left.

E.g. <exitcfg applysettings> will exec menus/applysettings.cfg whenever the page is left.

<timeout seconds command>
This defines a command to be run after a certain count of seconds has expired.

E.g. <timeout 5.0 "menu intro"> will start up the intro menu after 5 seconds have passed since creation of the page.

<light x y z intensity tint>
Used to illuminate Ghoul2 models with different lights. The default is purely linear ambient light.

<frametimeout timeout>
After the specified timeout in seconds, this frame will be destroyed.

<enterset cvar value>
Sets a cvar whenever this page is entered or visited

<exitset cvar value>
Sets a cvar whenever this page is exited

8.0 Layout Keyword Details

<tint name 0xABGR>
Equates a 32-bit RGBA longword color to a name. The format is little endian, so the components appear in the order ABGR. It is a layer of abstraction between raw numbers and names. The color name can be used anywhere a color can be, and a color does not

have to be named. For example, both “red” and “0xff0000ff” are both perfectly valid and usable in any situation. The default color names are defined in defines.rmf.

E.g.
<tint red 0xff0000ff>
<tint blue 0xffff0000>
<tint gray 0xff808080>
<tint darken 0x60000000>
<tint shadeblue 0x60ff0000>

<font keyword value keyword value etc>

Changes the current font on the page. Any text printed after a font command will be in the new font unless another font command is given.

Keywords

type – this can be either hud, title, medium, credtitle, credmed, lcd, lcdsmall or small. View the different fonts by type “menu fonts” at the console.
tint <tint> – defines the default color for all following text
atint <tint> - defines the default highlight color for all following text.
dropshadow – the text gets a simple and subtle drop shadow
blink – makes the text blink on and off 4 times a second
typematic – makes the text print one character at a time (just like the printing of the location)
reset – This resets the typematic printing time so multiple areas of one page appear at once.
speed <cps> – This defines the speed of the typematic printing in characters per second

E.g. <font type medium tint normaltext atint hilitetext>

A carriage return. This moves the current “next area position” down the height of the current font and to the left of the frame. If this is not a valid position, it will continue moving down until it is. Whenever the layout control is changed (i.e. a <right> or keyword is encountered), the “next area position” is set to the top of the page.

<hbr>

A carriage return. This works exactly the same as
 with the exception that the “next area position” is not reset to the top on a layout change. All following areas appear below the previous area.

<chbr cvar>

Acts as <hbr>, but only if the contents of cvar are non-zero.

A layout command that forces all following areas to appear centered on the screen.

<normal>
A layout command that places all following areas to the right of the previous area until no more will fit, and then a carriage return is applied. This is the default layout state, all areas appearing from left to right just like on a page of text.

<left>
This layout command places all the following areas starting at the left of the page going right. The only difference between this and <normal> is that <left> starts from the next available place on the left of the page and not the current “next area position”.

<right>
A layout command that places the following areas at the right of the page working left.

<cursor value>
Defines the cursor to be used in this frame. Current valid values are 0 for no cursor and 1 for the standard cursor.

E.g. <cursor 0> would turn off the cursor in this frame.

<defaults keyword value keyword value etc>
Allows easy definition of a standard set of defaults, and allows configuration of page wide items.

Keywords

noborder – All following areas are created without a border unless specifically requested.
border <width> <linewidth> <tint> – All following areas are created with the border defined here.
tiptint – Defines the color of the text in the tooltip popups.
tipbtint – Defines the color of the background of the tooltip popups. The light edge is this color 33% brighter; the dark edge is this color 33% darker.
tipfont – Defines the font to be used in the tooltips.

E.g. <defaults border 32 2 red tiptint green>

<autoscroll value>
Defines the speed at which the page vertically scrolls. Used on the credits screen to make the credits scroll slowly by. The lower the value of autoscroll, the slower the scroll speed. If all the text on the page can fit on one screen then the page will not scroll at all.

Keyword

timed <cvar> - makes the scroll take the amount of time specified in the cvar; used for subtitles.

E.g. <autoscroll 1.0> or <autoscroll .70 timed hud_subtitle_timer>

<tab value>
Defines a list of tab stops for layout in pixels. Normally used for one tab stop, but can define an unlimited amount. For example, this is used on the key-binding screen. One tab is defined (<tab 200>) and then the following areas have a keyword tab inside them that tells them to use that tab stop. All tab stops are scaled to the resolution, and the rendering to that tab stop is clipped to the following one.

E.g. <tab 100 200 300>

<bghoul value>
This loads in a ghoul model as a background so as to allow other areas to be bolted to it. See the definition of the ghoul area for details on this.

<backdrop keyword image keyword value>
Defines the background of the page. This can be formatted in many ways.

Keywords

tile <image> – tiles the following image to fill the frame (or the whole screen if the global flag is set)
stretch <image> – stretches the following image to fill the frame.
center <image> – centers the following image in the frame. Does not scale at all.
left <image> – puts the following image flush to the left of the screen
right <image>– puts the following image flush to the right of the screen.
bgcolor <tint> – defines the background color to be used.
cbgcolor <cvar> – defines the cvar that contains the background color to be used
setimage <cvar> - defines the cvar that contains the name of the image to use

E.g. <backdrop center “misc/logo” bgcolor green> would center “gfx/menus/misc/logo” on the frame and fill the rest of the screen green.

9.0 Area Keyword Details

Areas define the actual content of the screen. They are rectangular and lay themselves out automatically depending on the current control status. Each area is designed to have a good set of defaults, or at least make a good guess as to what it should be to minimize the number of parameters that need to be passed and/or parsed.

The following keywords can be used with all area types, although some have no meaning, are redundant or it would be silly to use them with some area types.

If an area is grayed out it cannot be changed. This is used to display the contents of game settings while in game and can be used to conditionally allow some settings depending on other settings. E.g. cannot enable Quincunx filtering if not in FSAA mode.

9.1 Common keywords

key <key> <command> - when this area is highlighted, and key is pressed, then command is added to the exec buffer.

ckey <key> <cvar> <false command> <true command> - when this area is highlighted, and key is pressed, if cvar contains zero, then false command is added to the exec buffer otherwise true command is added.
ikey <action> <command> - if any key bound to action is pressed while this area is highlighted, then command is added to the exec buffer.
tint <color> - the area will have the tint of the color. E.g. “tint blue” will filter out all red and green components
atint <color> - when this area is highlighted (i.e. the cursor is over the area) the area will be tinted by color.
btint <color> - some areas require/can have a backfill color – this is it.
ctint <color> - some areas need additional highlighting/differentiating colors – this is one of them.
dtint <color> - some areas need additional highlighting/differentiating colors – this is the other.
noshade – some areas automatically shade different parts of their components. noshade disables this feature.
noscale – bolt offsets are automatically scaled to the resolution. Noscale disables this feature.
border <width> <line width> <line color> - adds a border to the current area. The border is width wide, and in the center is a line of color line color that is line width wide.
width <value> - overrides the default width of the area. Required for a very few areas.
height <value> - overrides the default height of the area. Required for some areas.
cvar <cvar> - associates a cvar with this area. Used to pass data to and from areas.
cvaralpha <cvar> - Use the contents of this cvar to define the overall alpha of this area
inc <value> - specifies the step value to be used when keyboard shortcuts are used.
mod <value> - specifies the modulus of the area - the value at which the cvar associated with the area will wrap around.
archive – makes the cvar referenced by this area archived; ie remembered between game sessions
tip <text> - specifies the tooltip for this area.
itip <index> <tip> - defines the tooltip to be displayed when the selection equals the index
xoff <value> - specifies the horizontal offset from its defined position of this area. Normally used with bolted areas.
yoff <value> - as above, but vertical offset.
boltx <value> – specifies the horizontal offset as xoff, but can be used in addition to xoff
bolty <value> - as above, but vertical offset
tab - use a tab stop when laying out this area.
noborder – if a border is defined in the defaults, then this turns off borders for this area.
align <align> - either left, center or right. Used as an additional layout tool to override the current page settings.
scale <value> - scales both axes of the current area
scalex <value> - scale just the x axis
scaley <value> - scale just the y axis
overlay – specify this image is an overlay to a previously defined baseimage
backoverlay – specify this image overlays the background image
link <name> - a way of associating areas so that affects are propagated throughout areas with the same link name
iflt <cvar> <value> - only process this area if cvar is less than (<) value
ifgt <cvar> <value> - only process this area if cvar is greater than (>) value
ifle <cvar> <value> - only process this are if cvar is less than or equal (<=) value

ifge <cvar> <value>- only process this are if cvar is greater than or equal to (>=) value
ifne <cvar> <value>- only process this are if cvar is not equal to (!=) value
ifeq <cvar> <value>- only process this are if cvar is equal to (==) value
ifsame <cvar> <string>- only process this are if cvar contains string
ifnotsame <cvar> <string>- only process this are if cvar does not contain string
ifblank <cvar> - only process this are if cvar is blank
ifnotblank <cvar>- only process this are if cvar is not blank
ifset <cvar> <bit>- only process this are if bit is set in cvar
ifclr <cvar> <bit>- only process this are if bit is not set in cvar
graylflt <cvar> <value> - gray out the area if cvar is less than (<) value
graylfgt <cvar> <value> - gray out the area if cvar is greater than (>) value
graylfle <cvar> <value> - gray out the area if cvar is less than or equal (<=) value
graylfge <cvar> <value> - gray out the area if cvar is greater than or equal to (>=) value
graylfne <cvar> <value> - gray out the area if cvar is not equal to (!=) value
graylfeq <cvar> <value> - gray out the area if cvar is equal to (==) value
graylfsame <cvar> <string> - gray out the area if the cvar is the same as string
graylfnotsame <cvar> <string> - gray out the area if the cvar is not the same as string
graylfblank <cvar> - gray out the area if the cvar is blank
graylfnotblank <cvar> - gray out the area if the cvar is not blank
graylfset <cvar> <bit> - gray out the area if bit is set in cvar
graylfclr <cvar> <bit> - gray out the area if bit is not set in cvar
clickable – define the area as clickable i.e. Has a command attached to it.
change <time> - if the cvar associated with this area changes, then alter the alpha of this area (and linked areas) over time.

9.2 Area Keyword Definitions

<blank width height>
A blank area used for layout and collision purposes. For example, if you need a margin down the left hand side, just define a blank area of the margins width, very high and place on the left hand side of the screen.

E.g.
<blank 16 1024> would create a 16 wide margin

<hr width x tint color>
This command does a <hbr> command, displays a horizontal rule, and then another <hbr>. This is a good way of separating out different topics on the same page. It defaults to white and the full width of the frame.

E.g.
<hr width 200 tint gray>

<vbar tint color>
Creates a vertical scroll bar down the right hand side of the screen. It automatically sizes to the size and position of the page. An area with the **variable** keyword defined can change size – this is how things like the server list dynamically adjust the vbar properties.

E.g. <vbar tint vbargray>

<text text keyword value etc>

This creates an area containing a word that has properties (e.g. it is clickable). It uses the current font, font color and highlight color unless overridden. Regular words typed into the page format themselves to the dimensions of the page according to the layout controls, however they cannot be made clickable. The text area enables this.

Keywords

regular – use the normal text layout for this area
atext – defines an additional string to be displayed before this (like in list)

E.g. <text “DISCONNECT” key mouse1 “disconnect”> disconnects the game when the word is clicked on.
E.g. <text “&MENU_CREDITS_DESIGNER&” border 16 1 red> puts the contents of the strip token with a 1 pixel thick red border around it.

<ctext cvar keyword value etc>

This creates an area containing the contents of the cvar. It uses the current font, font color and highlight color unless overridden. This works in very much the same way as text, with the exception that the text is directed through a cvar.

Keywords

atext – text to be displayed to the left of the contents of cvar
invisible – if the cvar contains no info, do not display the atext
integer – display only the integer portion of the cvar

E.g. Your GL Driver is <ctext gl_driver>

<image imagefile keyword value etc>

This creates an area the size of the image loaded.

Keywords

alt <image> - Sets the alternate highlight image of this area. When the cursor moves over this area, the original image is changed to this one. This image will scale to the dimensions of the original.
hflip - Flips the image horizontally.
vflip - Flips the image vertically.
setimage <cvar> - the image displayed is named in the cvar string. Used for the crosshair.
proportional – this makes the image scale so it takes up the same amount of screen real estate as if it were displayed on a 640x480 screen
baseimage – this means subsequent areas with the overlay command will appear at an offset to this area
saving – used to define this as a savegame screenshot – if saves are not allowed at this point, then a page with this image will fail to create.

E.g. <image "misc/gold_star" scale .5 proportional> will scale the image to half its size proportionally.

<list liststuff keyword value etc>

This is the general workhorse area. You give a list of valid options to go through and a left mouse click will go forward through them, a right mouse click backwards. If no matching string is given, then the index of the item is the value put into the cvar. A caveat to watch out for is having two entries of the same value.

Keywords

atext <text> - defines text to be placed before the item.
match <list> - defines a parallel list of values that the items in the original list refer to.
bitmask <bit number> - works on the bit number of the cvar value rather than the whole cvar. Used for deathmatch flags. More than 2 items in the list could cause undefined results.
files <root> <base> <ext> - grabs the list of files from "root/base.ext" and allows the user to select through the files. Only the base part of the filename is displayed.
lock <index> <cvar> <value> - if the list gets set to index, then cvar will be set to value. Multiple locks can be attached to one area. This is used for mutually exclusive but independent settings.

E.g.
<list "640x480,800x600,1024x768" match "3,4,6" cvar gl_mode> would be awkward as every time you click on this box, the video system would be restarted.
<list "56k,Cable,DSL,T1" match "3500,10000,10000,20000" cvar rate> will not work past the Cable option because it will set rate to 10000, re-lookup the value when DSL is selected and think it is Cable.

<slider cvar value keyword value etc>

This creates a generic slider area, which defaults to horizontal, but can be vertical. The max (far right or top) must be greater than the min (far left or bottom). The default min is 0.0f and the default max is 1.0f. While highlighted, pressing the arrow keys will modify the slider value by the step value (which defaults to 0.1f). Buttons at either end of the slider bar are created automatically and also modify the cvar by the step value.

Keywords

min <value> - the value the cvar will be set to when the button is at the far left of the slider. Defaults to 0.0f.
max <value> - the value the cvar will be set to when the button is at the far right of the slider. Defaults to 1.0f.
cvarmax <cvar> - overrides the max value to the contents of cvar.
step <value> - modifies the step value from the default 0.1f.
vertical – makes the slider bar a vertical one.
horizontal - make the slider bar a horizontal one (default)
bar <image> - overrides the default bar graphic to your custom one.
button <image> - overrides the default button graphic.
cap <image> - overrides the default end cap graphic.

E.g. <slider cvar r_gamma> will create a horizontal slider bar ranging from 0.0f to 1.0f and directly modify the r_gamma cvar.

<ticker text width value keyword value etc>

Creates an item of scrolling text to the width you define.

Keywords

speed – the speed at which the text scrolls
delay – the delay before the text starts to scroll into the rectangle

E.g. <ticker “&MENU_CREDITS_DESIGNER&” width 200 speed 10>

<input width value cvar value keyword value etc>

Defines a text input box of a specified width. The input is stored in the specified cvar.

Keywords

global - any text input on this page will be captured by this area. Only one global input area is allowed per page.
hidden – replace the echoing of the typed data with repeats of *. Used for password input.
maxchars <value> - the maximum number of characters that can be input into this area. The default is however many can be displayed in the area.
history – allows the input area to remember previously entered lines. Cursor up retrieves the previous line etc.

E.g. <input cvar ar_seed_text width 160 border 12 1 text global tint clickable archive maxchars 20>

<setkey action keyword value etc>
Examines the following command and displays up to two keys that are bound to that command.

Keywords

atext <string> - the text associated with the key to bind
nobind – this key is not bindable, just displayable.

E.g. <setkey +forward atext "&MENU_KEYS_FORWARD& : " tab btint clickable> will create something like:

Forward Mouse3, w

<selection image width height>

Selection creates a box that allows the selection of different types of items. Use the console command **select** to pick items; if a selected cvar has the pistol flag set, then it will be put in the selection box marked with pistol. The size limits, associated ammo and other data are defined in the cvar appended with _info.

E.g.
setp wp_m1911a1 "0"
set wp_m1911a1_info "weapon_renders/m1911a1,1/7/0/0,am_045acp,7,none,0,none"

Use “select wp_m1911a1” to put this pistol in the selection box. The info cvar is a comma separated list defined as -
“weapon_renders/m1911a1” – image to display when selected
“1/7/0/0” – Pick 1 at once, it comes with a clip of 7, it has no secondary weapon and no accessories.
“am_045acp” – The used ammunition type
“7” – Clip size of 7
“none” – No secondary ammunition type.
“0” – No secondary ammunition
“none” – there is no possibility of accessories.

Keywords

backimage <image> - defines the background image to tile when there is no selection
proportional – makes the selection box proportional to the screen resolution
rifle – defines the type of selection as rifle
pistol– defines the type of selection as pistol
item– defines the type of selection as item
ammo– defines the type of selection as ammo
misc– defines the type of selection as misc
grenade– defines the type of selection as grenade

<ghoul modelpath keyword value etc>

This loads and displays a ghoul2 3d model and can optionally play an animation.

Keywords

yaw <cvar> – defines a cvar to control the yaw of the model
pitch <cvar> – defines a cvar to control the pitch of a model
roll <cvar> – defines a cvar to control the roll of a model
gscale <value> – scales a ghoul2 model in addition to the regular area scaling.
oneshot – only play the ghoul2 anim once
animate – animate this model
anim – the name of the animation to play on this model
rotate <roll> <pitch> <yaw> - the angular velocity of the model
rotation <roll> <pitch> <yaw> - the initial angles of the model
end <command> - the command to add to the console when the model animation completes

E.g. <ghoul "weapons/AK74_no_acc_MS/AK74_no_acc_MS" border 1 0 smokey height 74 scale 6 anim "ak74_no_acc_ms" rotation 180 0 0>

<filebox root base ext>

Displays a list of files from a given directory from which the user can select one.

Keywords

backfill <tint> - defines the background color for this rectangle
sort <sort> - sort can be name, size, time, rname, rsize or rtime. This is the criterion used to sort the files displayed.

<filereq root base ext>

A basic file requester.

Keywords

backfill <tint> - defines the background color for this rectangle
sort <sort> - sort can be name, size, time, rname, rsize or rtime. This is the criterion used to sort the files displayed

<loadbox keyword value etc>

Displays a list of savegame files with images, titles and dates.

Keywords

backfill <tint>- defines the background color for this rectangle
sort <sort> - sort can be name, size, time, rname, rsize or rtime. This is the criterion used to sort the files displayed
saving – this is a saving screen, so do not allow if not in a game or dead
spacing <value> - the number of pixels separating each savegame snapshot

E.g. <loadbox width 400 cvar menu_save key mouse1 "confirm cl_confirmations loadconfirm loadgame" key mouse2 "confirm cl_confirmations wipeconfirm wipe" spacing 16>

<loadlist keyword value etc>

Displays a list of savegame files without images, but with a title, difficulty and date. When a savegame is selected the menus upload the image to be displayed a set the image name in the associated cvar.

Keywords

backfill <tint>- defines the background color for this rectangle
sort <sort> - sort can be name, size, time, rname, rsize or rtime. This is the criterion used to sort the files displayed
saving – this is a saving screen, so do not allow if not in a game or dead
diffcvar <cvar> - where to put the difficulty level of the currently selected savegame
timecvar <cvar> - where to put the time and date of the currently selected savegame
auto – defines whether to list auto savegames or in game saves.

E.g. <loadlist width 200 btint clickable cvar menu_save timecvar menu_load_time diffcvar menu_load_diff auto>

<hbar width height>

This creates a horizontal progress bar. If the width and height values are numeric, they define the width and height of a uniform colored bar. If they are shader names, then

width is the name of the shader that acts as a background and height is the name of a shader that repeats for the foreground.

Keywords

invisible – if the associated cvar contains zero, then don't display an empty bar
invert – draws the big background image on top of the repeating foreground image

E.g. <hbar "misc/load_clip" "misc/load_bullet" cvar ldg_progress>

<meter width height keyword value etc>

A pseudo random bar graph based on the contents of a cvar – used in the PADD.

E.g. <meter 80 36 overlay xoff 14 yoff 12 tint hudgray ctint paddyellow cvar hud_padd atint red>

Appendix A: Menu Examples

m_keys.rmf is the menu page that allows you to change your key bindings. It is pretty much the same command over and over with different values. However, you will see basic formatting rules used in every menu. Here is an excerpt from the file with some of the setkey commands cut out for space.

```
<stm>

<vbar tint vbartint>

<tab 225>
<center>
<font type title tint text dropshadow>
<hbr>

<text "&MENU_KEYS_MOVEMENT&">
<hr width 256 tint text>
<font type medium tint clickable atint reversed>
<left>
<hbr>

<setkey +forward atext "&MENU_KEYS_FORWARD& : " tab btint clickable><hbr>
<setkey +back atext "&MENU_KEYS_BACK& : " tab btint clickable><hbr>
<setkey +left atext "&MENU_KEYS_LEFT& : " tab btint clickable><hbr>
<setkey +right atext "&MENU_KEYS_RIGHT& : " tab btint clickable><hbr>
<setkey +moveleft atext "&MENU_KEYS_STEP_LEFT& : " tab btint clickable><hbr>
<setkey +moveright atext "&MENU_KEYS_STEP_RIGHT& : " tab btint clickable><hbr>
<setkey +moveup atext "&MENU_KEYS_UP& : " tab btint clickable><hbr>
<setkey +movedown atext "&MENU_KEYS_DOWN& : " tab btint clickable><hbr>
<setkey +prone atext "&MENU_KEYS_PRONE& : " tab btint clickable><hbr>
<setkey +speed atext "&MENU_KEYS_RUN& : " tab btint clickable><hbr>
<setkey +strafe atext "&MENU_KEYS_STRAFE& : " tab btint clickable><hbr>
<setkey +leanleft atext "&MENU_KEYS_LL& : " tab btint clickable><hbr>
<setkey +leanright atext "&MENU_KEYS_LR& : " tab btint clickable><hbr>
<setkey +lookup atext "&MENU_KEYS_LOOK_UP& : " tab btint clickable><hbr>
<setkey +lookdown atext "&MENU_KEYS_LOOK_DOWN& : " tab btint clickable><hbr>
<setkey centerview atext "&MENU_KEYS_CENTER& : " tab btint clickable><hbr>
<setkey +mlook atext "&MENU_KEYS_MOUSE_LOOK& : " tab btint clickable><hbr>
</stm>
```

Here's an example of a setup for other menu files. This contains all the frames that are used for the objectives. Then, each frame references another menu file or sub page. This is the standard setup for all menus.

```
<stm>

<ikey "menu objective" popmenu>

<set m_obj_info "m_obj_info">

<frame back 0 1 page b_objective>
<frame objective 0 0 perimeter 32 30 30 80>

<frame top 0 104 cut objective border 2 1 black>
<frame obj_main_title 0 20 page m_obj_main_title cut top>
<frame obj_main 0 78 page m_obj_main cut top border 4 0 clear>

<frame middle 0 104 cut objective border 2 1 black>
<frame obj_sub_title 0 20 page m_obj_sub_title cut middle>
<frame obj_sub 0 78 page m_obj_sub cut middle border 4 0 clear>

<frame bottom 0 124 cut objective border 2 1 black>
<frame obj_info_title 0 20 page m_obj_info_title cut bottom>
<frame obj_info 0 84 cpage m_obj_info cut bottom border 4 0 clear>
<frame obj_aboutinfo 0 18 page m_obj_aboutinfo cut bottom border 4 0 clear>

<frame linebreak 0 6 cut objective>
<frame topbuttons 0 0 page m_obj_buttons cut objective>

</stm>
```

Here's an example of a HUD. There's really no difference between a menu and a HUD except for when they're called. With a HUD, the game is still running.

```
<hud>
<backdrop bgcolor black>

<left>

<font type credmed tint gray>
<hr tint clear>
<text "&MENU_CREDITS_CREATIVE_DIR&"><br>
<font type credmed tint darkred>
<text "&NAMES_BRAFFEL&"><br>

<font type credmed tint gray>
<hr tint clear>
<text "&MENU_CREDITS_PROJECTLEAD&"><br>
<font type credmed tint darkred>
<text "&NAMES_JZUK&"><br>

<font type credmed tint gray>
<hr tint clear>
<text "&MENU_CREDITS_LEADART&"><br>
<font type credmed tint darkred>
<text "&NAMES_JKOBERSTEIN&"><br>

<font type credmed tint gray>
<hr tint clear>
<text "&MENU_CREDITS_LEADPROGRAM&"><br>
```

```
<font type credmed tint darkred>
<text "&NAMES_DKRAMER&"><br>

<font type credmed tint gray>
<hr tint clear>
<text "&MENU_CREDITS_MPLEAD&"><br>
<font type credmed tint darkred>
<text "&NAMES_RJOHNSON&"><br>

</hud>
```