

## **Curses**

**COLLABORATORS**

	<i>TITLE :</i> Curses		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		August 26, 2022	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>Curses</b>	<b>1</b>
1.1	Curses implementation for AmigaTalk© 1998: . . . . .	1
1.2	Control of the Curses Package: . . . . .	2
1.3	Displaying of Text for the Curses Package: . . . . .	6
1.4	Retrieving of Text for the Curses Package: . . . . .	6
1.5	Insertion of Text for the Curses Package: . . . . .	7
1.6	Removal of Text for the Curses Package: . . . . .	8
1.7	Miscellaneous Curses methods: . . . . .	9

---

# Chapter 1

## Curses

### 1.1 Curses implementation for AmigaTalk© 1998:

The Curses interface for the AmigaTalk system is documented herein & is only a subset of the complete Curses package utilized by the rest of the world. Most of the methods are accessed from primitive 124, with the exception of the printAt: method, which uses primitive 126.

The following Curses functions are NOT implemented by AmigaTalk, since they use variable arguments & it will take someone with more than my miniscule knowledge of smalltalk to utilize them. They can be faked by using other parts of the AmigaTalk system & the Curses package, so in a sense they are redundant anyway:

```
int printw( char *fmt, ... );
```

```
int wprintw( WINDOW *win, char *fmt, ... );
```

```
int mvprintw( int line, int col, char *fmt, ... );
```

```
int mvwprintw( WINDOW *win, int line, int col, char *fmt, ... );
```

```
int scanw( char *fmt, ... );
```

```
int wscanw( WINDOW *win, char *fmt, ... );
```

```
int mvscanw( int line, int col, char *fmt, ... );
```

```
int mvwscanw( WINDOW *win, int line, int col, char *fmt, ... );
```

Methods available for the Curses package are:

[Curses Control](#)

[Text Display](#)

[Text Retrieval](#)

[Text Insertion](#)

[Text Removal](#)

[Miscellaneous](#)

---

## 1.2 Control of the Curses Package:

new

Initialize the Curses Class.

initialize

Open the Curses screen (equivalent to `initscr()`).

closeDown

Kill the Curses environment (equivalent to `endwin()`)

initializeWithColors: depth

Open the Curses screen & use colors (equivalent to:  
`StartColor()` [non-standard Curses] followed by `initscr()`).

initWithStdColors

Open the Curses screen & use colors (equivalent to:  
`start_colors()` followed by `initscr()`).

openWindow: xStart yStart: y width: w height: h

Open a new Curses window (equivalent to `newwin()`).

openSubWindow: parent xStart: x yStart: y width: w height: h

Open a Curses sub-window (equivalent to `subwin()`).

refreshScreen

Refresh the Curses environment (equivalent to `refresh()`).

refreshWindow: winNumber

Refresh a single Curses window (equivalent to `wrefresh()`).

closeWindow: winNumber

Close a Curses window (equivalent to `delwin()`, followed by  
`refreshScreen`).

moveWindow: winNumber x: x y: y

Move a Curses window (equivalent to `mvwin()`, followed by  
`refreshWindow:`).

cBreak: status

Either set (status = TRUE: call `cbreak()`) or reset  
(status = FALSE: call `ncbreak()`) the control-break of the  
Curses system.

enableClear: winNumber status: status

Either enable (`clearok( status = TRUE )`) or disable  
(`clearok( status = FALSE )`) the clear function for the given  
window.

enableCursor: winNumber status: status

Either enable (`leaveok( status = TRUE )`) or disable  
(`leaveok( status = FALSE )`) the cursor for the given window.

---

newlineMap: status

Either enable (nl(), status = TRUE) or disable

(nonl(), status = FALSE) the mapping of CRLF to CR function for the given window.

echo: status

Either enable (echo(), status = TRUE) or disable

(noecho(), status = FALSE) the echo function for the given window.

enableDelay: winNumber status: status

Either enable (nodelay( status = FALSE )) or disable

(nodelay( status = TRUE )) the delay function for the given window.

setColor: number red: r green: g blue: b

Change the color register number to the new RGB values supplied then perform refreshScreen.

setTextPenColor: colornum

Change the text color rendering pen to register colornum.

NOTE: Non-standard Curses.

setBackPenColor: colornum

Change the background color rendering pen to register colornum.

NOTE: Non-standard Curses.

setDrawMode: mode

Change the drawing mode to mode. Allowable modes are:

0 = JAM1

1 = JAM2

2 = COMPLEMENT

4 = INVERSEVID

NOTE: Non-standard Curses.

enableScroll: winNumber status: status

Either enable (scrollok( status = TRUE )) or disable

(scrollok( status = FALSE )) the scroll function for the given window.

enableKeyPad: winNumber status: status

Either enable (keypad( status = TRUE )) or disable

(keypad( status = FALSE )) the following key values to be sent to the given window:

KEY\_BACKSPACE 0010 /\* backspace character \*/

KEY\_DC 0177 /\* Delete character \*/

KEY\_DOWN 0400 /\* The down arrow key \*/

KEY\_UP 0401 /\* The up arrow key \*/

KEY\_LEFT 0402 /\* The left arrow key \*/

KEY\_RIGHT 0403 /\* The right arrow key \*/

KEY\_HELP 0404 /\* Help Key \*/

KEY\_F0 0405 /\* Function keys \*/

KEY\_F(n) (KEY\_F0+(n))

scrollWindow: winNumber

Scroll the given window (scroll()), followed by refreshWindow:).

setScrollRegion: top Bottom: bott

Setup the scroll region for the Curses system (equivalent to setscreg(), followed by refreshScreen).

setWindowScrollRegion: winNumber top: top Bottom: bott

Setup the scroll region for the given window (equivalent to wsetscreg(), followed by refreshWindow:).

windowNeedsRefresh: winNumber

Tell Curses to add the given window to the refresh list.

setAttributes: attrs

Setup the Curses system attributes, where attrs is one or more of the following:

NORMAL = 0

INVERSE = 16

UNDERLINE = 32

BOLD = 64 Follow up with a call to refreshScreen.

setWindowAttributes: winNumber attr: attrs

Setup the given window attributes, where attrs is one or more of the following:

NORMAL = 0

INVERSE = 16

UNDERLINE = 32

BOLD = 64 Follow up with a call to refreshWindow:.

addAttributes: attrs

Add a Curses system attribute, where attrs is one or more of the following:

NORMAL = 0

INVERSE = 16

UNDERLINE = 32

BOLD = 64 Follow up with a call to refreshScreen.

addWindowAttributes: winNumber attr: attrs

Add an attribute to the given window, where attrs is one or more of the following:

NORMAL = 0

INVERSE = 16

UNDERLINE = 32

BOLD = 64 Follow up with a call to refreshWindow:.

removeAttributes: attrs

Remove a Curses system attribute, where attrs is one or more of the following:

NORMAL = 0

INVERSE = 16

UNDERLINE = 32

BOLD = 64 Follow up with a call to refreshScreen.

removeWindowAttributes: winNumber attr: attrs

Remove an attribute from the given window, where attrs is one or more of the following:

NORMAL = 0

INVERSE = 16

UNDERLINE = 32

BOLD = 64 Follow up with a call to refreshWindow:.

invertColors

Perform a Curses standout() command, followed by refreshScreen.

invertWindowColors: winNumber

Perform a Curses wstandout() command, followed by refreshWindow:.

revertColors

Undo the invertColors (standout()) command by performing a standend() Curses command, followed by refreshScreen.

revertWindowColors: winNumber

Undo the invertWindowColors (wstandout()) command by performing a wstandend() Curses command, followed by refreshWindow:.

moveCursorTo: aPoint

Move the cursor from the current location to the given point (equivalent to move(), followed by refreshScreen).

moveWindowCursor: winNumber to: aPoint

Move the cursor from the current location in the given window to the given point (equivalent to wmove(), followed by refreshWindow:).

addToRefreshList: winNumber

Add the given window to the refresh list (equivalent to wnoutrefresh()).

updateWindows

Update all windows on the refresh list (equivalent to doupdate()).

flushKeys

Throw away all keystrokes in the system buffer (equivalent to flushinp()).

moveCursorFrom: aPoint to: newPoint

Move the cursor from the given point to the newPoint (equivalent to mvcur(), followed by refreshScreen).



### 1.3 Displaying of Text for the Curses Package:

For primitive 126, defined within the String Class, printAt:

is:

`printAt`: The argument must be a Point which describes a location on the Curses screen. The string is printed at the specified location.

`printChar`: c

Print a character c at the current cursor location (equivalent to `addch()`, followed by `refreshScreen()`).

`printWindowChar`: winNumber char: c

Print a character c at the current cursor location in the given window (equivalent to `waddch()`, followed by `refreshWindow()`).

`printChar`: c at: thePoint

Print a character c at the given location (equivalent to `mvaddch()`, followed by `refreshScreen()`).

`printWindowChar`: winNumber char: c at: thePoint

Print a character c at the given location in the given window (equivalent to `mvwaddch()`, followed by `refreshWindow()`).

`printString`: string

Print string at the current cursor location (equivalent to `addstr()`, followed by `refreshScreen()`).

`printWindowString`: winNumber string: string

Print a string at the current cursor location in the given window (equivalent to `waddstr()`, followed by `refreshWindow()`).

`printString`: string at: thePoint

Print a string at the given location (equivalent to `mvaddstr()`, followed by `refreshScreen()`).

`printWindowString`: winNumber string: string at: thePoint

Print a string at the given location in the given window (equivalent to `mvwaddstr()`, followed by `refreshWindow()`).

### 1.4 Retrieving of Text for the Curses Package:

`getChar`

Return the character at the current cursor location (equivalent to `getch()`).

`getWindowChar`: winNumber

Return the character at the current cursor location in the given

window (equivalent to wgetch()).

getCharAt: aPoint

Return the character at the given location (equivalent to mvgetch()).

getWindowChar: winNumber at: aPoint

Return the character at the given location in the given window

(equivalent to mvwgetch()).

getString: buffer

Return a string at the current cursor location (equivalent to getstr()).

getWindowString: winNumber buffer: string

Return a string at the current cursor location for the given window

(equivalent to wgetstr()).

getStringAt: aPoint buffer: string

Return a string at the given location (equivalent to mvgetstr()).

getWindowString: winNumber at: aPoint buffer: string

Return a string at the given location for the given window

(equivalent to mvwgetstr()).

readChar

Wait for the user to type in a character (equivalent to inch()).

readWindowChar: winNumber

Wait for the user to type in a character in the given window

(equivalent to winch()).

readCharAt: aPoint

Wait for the user to type in a character at the given location

(equivalent to mvinch()).

readWindowChar: winNumber at: aPoint

Wait for the user to type in a character at the given location in

the given window (equivalent to mvwinch()).

## 1.5 Insertion of Text for the Curses Package:

insertChar

Insert a character at the current cursor location (equivalent to insch(), followed by refreshScreen).

insertWindowChar: winNumber

Insert a character at the current cursor location in the given window (equivalent to winsch(), followed by refreshWindow:).

insertCharAt: aPoint

Insert a character at the given location (equivalent to mvinsch(), followed by refreshScreen).

`insertWindowChar: winNumber at: aPoint`

Insert a character at the given location in the given window (equivalent to `mvwinsch()`, followed by `refreshWindow:`).

`insertLine`

Insert a line at the current cursor location (equivalent to `insertln()`, followed by `refreshScreen`).

`insertWindowLine: winNumber`

Insert a line at the current cursor location in the given window (equivalent to `winsertln()`, followed by `refreshWindow:`).

## 1.6 Removal of Text for the Curses Package:

`emptyScreen`

Erase the Curses system display (equivalent to `erase()`, followed by `refreshScreen`).

`emptyWindow: winNumber`

Erase the given window display (equivalent to `werase()`, followed by `refreshWindow:`).

`clearScreen`

Clear the Curses system display (equivalent to `clear()`, followed by `refreshScreen`).

`clearWindow: winNumber`

Clear the given window display (equivalent to `wclear()`, followed by `refreshWindow:`).

`clearScreenToBottom`

Clear the Curses system display from the current location to the bottom (equivalent to `clrrobot()`, followed by `refreshScreen`).

`clearWindowToBottom: winNumber`

Clear the given window display from the current location to the bottom (equivalent to `wclrrobot()`, followed by `refreshWindow:`).

`clearScreenToEOL`

Clear the Curses system display from the current location to the end of the current line (equivalent to `clrtoeol()`, followed by `refreshScreen`).

`clearWindowToEOL: winNumber`

Clear the given window display from the current location to the end of the current line (equivalent to `wclrtoeol()`, followed by `refreshWindow:`).

`deleteChar`

---

Remove a character at the current cursor location from the Curses system display (equivalent to `delch()`, followed by `refreshScreen`).

`deleteWindowChar`: winNumber

Remove a character at the current cursor location from the given window display (equivalent to `wdelch()`, followed by `refreshWindow`).

`deleteCharAt`: aPoint

Remove a character at the given cursor location from the Curses system display (equivalent to `mvdelch()`, followed by `refreshScreen`).

`deleteWindowChar`: winNumber at: aPoint

Remove a character at the given cursor location from the given window display (equivalent to `mvwdelch()`, followed by `refreshWindow`).

`deleteLine`

Delete the current line the cursor is on (equivalent to `deleteln()`, followed by `refreshScreen`).

`deleteWindowLine`: winNumber

Delete the current line the cursor is on in the given window (equivalent to `wdeleteln()`, followed by `refreshWindow`).

## 1.7 Miscellaneous Curses methods:

`drawBorder`: winNumber hChar: hc vChar: vc

Draw a box around the given window (equivalent to `box()`, followed by `refreshWindow`).

`beep`

Send a Ctrl-G (ASCII Bel) to the system.

`flash`

Flash the Curses screen.

`hasColors`

Return the status of the Curses screen color usage.

---