VidMgr 1.3 documentation
Revision 1.1
Written by Andrew Clarke

The following document describes version 1.3 of the VidMgr library as written by the author, Andrew Clarke, in October 1996. VidMgr provides a set of public domain screen drawing, cursor and keyboard routines for text mode DOS, OS/2 and 32-bit Windows 9x/NT applications. Full source code is included in the ZIP archive (named *vidmgr13.zip).*

You may contact the author via Internet e-mail at randy@zws.com for any comments or questions about this document. Corrections and additions are welcome.

`void vm_init(void);`

This initialises the VidMgr subsystem. This function should be called before any other function contained in the VidMgr library. If you previously called the *vm_done()* function and wish to continue using the functions from the VidMgr library, you should call *vm_init()*. *No value is returned from the function, so it is assumed that the call succeeded.*

`void vm_done(void);`

Exits the VidMgr subsystem. This function should be called when your application exits, or often when you wish to run an external application (eg. before calling *system()*). Success is assumed and no value is returned.

`char vm_getscreenwidth(void);`

Returns a value equal to the width of the screen in columns. On DOS, OS/2 and Windows systems the value returned is commonly 80. *(Note that screens wider than 255 characters, while uncommon, are not supported.)*

`char vm_getscreenheight(void);`

Returns a value equal to the height of the screen in rows. On DOS, OS/2 and Windows systems the value returned is commonly 25. (*Note that screens deeper than 255 rows, while uncommon, are not supported.)*

short vm_getscreensize(void);

Returns a value equal to the amount of memory required to hold all the information currently contained on the screen. For a screen 80 columns wide and 25 rows (or lines) high, the value returned is 4,000. *This function is deprecated and may be removed from future versions of VidMgr. In any case, it can be calculated using:*

```
size = vm_getscreenwidth() * vm_getscreenheight() * 2;
```

void vm_gotoxy(char x, char y);

Positions the text cursor at column *x*, row *y*. A coordinate of *1, 1* points to the upper-left hand side of the screen. No value is returned.

char vm_wherex(void);

Returns a value equal to the X (column) position of the text cursor. A value of 1 points to the far-left hand side of the screen. *Columns beyond 255 are not supported.*

char vm_wherey(void);

Returns a value equal to the Y (row) position of the text cursor. A value of 1 points to the top of the screen. *Rows beyond 255 are not supported.*

int vm_kbhit(void);

Returns a non-zero value if a key on the keyboard was pressed. This may exclude some shift (Ctrl, Alt, Shift) and other function keys on some systems. In DOS programs running under DESQview, Windows or OS/2, this function releases "timeslices" to the system while it is idle to prevent it stealing CPU cycles from other applications running concurrently. Support for timeslicing under Novell Netware and DoubleDOS is also provided, but this is untested.

int vm_getch(void);

Waits for a key to be pressed on the keyboard and returns its value. See Table 1.1 and Table 1.1.1 for a list of keycode values returned by this function.

Table 1.1: Keycode values returned by *vm_getch()*.

Keycode	Description or Value (hex)
0x01	*Ctrl+A*
0x02	*Ctrl+B*
0x03	*Ctrl+C*
0x04	*Ctrl+D*
0x05	*Ctrl+E*
0x06	*Ctrl+F*
0x07	*Ctrl+G*
0x08	*Ctrl+H, Backspace*
0x09	*Ctrl+I*
0x0a	*Ctrl+J, Ctrl+Enter*
0x0b	*Ctrl+K*
0x0c	*Ctrl+L*
0x0d	*Ctrl+M, Enter*
0x0e	*Ctrl+N*
0x0f	*Ctrl+O*
0x10	*Ctrl+P*
0x11	*Ctrl+Q*
0x12	*Ctrl+R*
0x13	*Ctrl+S*
0x14	*Ctrl+T*
0x15	*Ctrl+U*
0x16	*Ctrl+V*
0x17	*Ctrl+W*
0x18	*Ctrl+X*
0x19	*Ctrl+Y*
0x1a	*Ctrl+Z*
0x1b	*Ctrl+[, Escape*
‘A’ .. ‘Z’	*A .. Z*
‘a’ .. ‘z’	*a .. z*
‘0’ .. ‘9’	*0 .. 9*
‘`’ ‘~’ ‘!’ ‘@’ ‘#’ ‘$’ ‘%’ ‘^’ ‘&’ ‘’ ‘(‘ ‘)’ ‘-‘ ‘_’ ‘=’ ‘+’ ‘[‘ ‘{‘ ‘]’ ‘}’ ‘|’ ‘\’ ‘;’ ‘:’ ‘\’’ ‘”’ ‘<’ ‘>’ ‘,’ ‘.’ ‘?’ ‘/’*	*` ~ ! @ # $ % ^ & * ( ) - _ = + [{] } | \ ; : ’ ” < > , . ? /*

Table 1.1.1: Extended keycode values returned by *vm_getch()*.

The following are 'extended' keycodes and may not be available on all systems:

Keycode (hex)	Description
0x1c	*Ctrl+*
0x1d	*Ctrl+]*
0x1e	*Ctrl+^*
0x1f	*Ctrl+-*
0x7f	*Ctrl+Backspace*
0x0300	*Ctrl+@*
0x3b00	*F1*
0x3c00	*F2*
0x3d00	*F3*
0x3e00	*F4*
0x3f00	*F5*
0x4000	*F6*
0x4100	*F7*
0x4200	*F8*
0x4300	*F9*
0x4400	*F10*
0x8500	*F11*
0x8600	*F12*
0x5400	*Shift+F1*
0x5500	*Shift+F2*
0x5600	*Shift+F3*
0x5700	*Shift+F4*
0x5800	*Shift+F5*
0x5900	*Shift+F6*
0x5a00	*Shift+F7*
0x5b00	*Shift+F8*
0x5c00	*Shift+F9*
0x5d00	*Shift+F10*
0x8700	*Shift+F11*
0x8800	*Shift+F12*
0x5e00	*Ctrl+F1*
0x5f00	*Ctrl+F2*
0x6000	*Ctrl+F3*
0x6100	*Ctrl+F4*
0x6200	*Ctrl+F5*
0x6300	*Ctrl+F6*
0x6400	*Ctrl+F7*
0x6500	*Ctrl+F8*

0x6600	*Ctrl+F9*
0x6700	*Ctrl+F10*
0x8900	*Ctrl+F11*
0x8a00	*Ctrl+F12*
0x6800	*Alt+F1*
0x6900	*Alt+F2*
0x6a00	*Alt+F3*
0x6b00	*Alt+F4*
0x6c00	*Alt+F5*
0x6d00	*Alt+F6*
0x6e00	*Alt+F7*
0x6f00	*Alt+F8*
0x7000	*Alt+F9*
0x7100	*Alt+F10*
0x8b00	*Alt+F11*
0x8c00	*Alt+F12*
0x5200	*Insert*
0x4700	*Home*
0x4900	*Page Up*
0x5300	*Delete*
0x4f00	*End*
0x5100	*Page Down*
0x4b00	*Left arrow*
0x4d00	*Right arrow*
0x4800	*Up arrow*
0x5000	*Down arrow*
0x9200	*Ctrl+Insert*
0x7700	*Ctrl+Home*
0x8400	*Ctrl+Page Up*
0x9300	*Ctrl+Delete*
0x7500	*Ctrl+End*
0x7600	*Ctrl+Page Down*
0x7300	*Ctrl+Left arrow*
0x7400	*Ctrl+Right arrow*
0x8d00	*Ctrl+Up arrow*
0x9100	*Ctrl+Down arrow*
0xa200	*Alt+Insert*
0x9700	*Alt+Home*
0x9900	*Alt+Page Up*
0xa300	*Alt+Delete*
0x9f00	*Alt+End*
0xa100	*Alt+Page Down*
0x9b00	*Alt+Left arrow*
0x9d00	*Alt+Right arrow*

0x9800	*Alt+Up arrow*
0xa000	*Alt+Down arrow*

#define vm_mkcolor(fore, back)

A macro that generates video attribute values compatible with VidMgr, based on a foreground and background color pair, eg.

vm_mkcolor(LIGHTCYAN, BLUE)

Refer to Table 1.2 for a list of video attributes that can be used with this function. *vm_mkcolour() i*s a synonym for *vm_mkcolor()* for non-American English programmers.

Table 1.2: Video attributes that may be used with *vm_mkcolor().*

Name	**Value**
BLACK	*0x00*
BLUE	*0x01*
GREEN	*0x02*
CYAN	*0x03*
RED	*0x04*
MAGENTA	*0x05*
PURPLE	*MAGENTA*
BROWN	*0x06*
LIGHTGRAY	*0x07*
LIGHTGREY	*LIGHTGRAY*
DARKGRAY	*0x08*
DARKGREY	*DARKGRAY*
LIGHTBLUE	*0x09*
LIGHTGREEN	*0x0a*
LIGHTCYAN	*0x0b*
LIGHTRED	*0x0c*
LIGHTMAGENTA	*0x0d*
LIGHTPURPLE	*LIGHTMAGENTA*
YELLOW	*0x0e*
WHITE	*0x0f*

#define vm_fore(attr)

A macro that returns the foreground color of a video attribute pair, eg.

vm_fore(vm_mkcolor(LIGHTCYAN, BLUE)) / returns LIGHTCYAN */*

#define vm_back(attr)

A macro that returns the background color of a video attribute pair, eg.

vm_back(vm_mkcolor(LIGHTCYAN, BLUE)) / returns BLUE */*

void vm_paintclearbox(char x1, char y1, char x2, char y2, char attr);

Fills a box on the screen with the *attr* video attribute and clears its contents. The location of the text cursor remains fixed. The value of the video attribute can be generated using the *vm_mkcolor()* macro, eg.

vm_paintclearbox(20, 5, 60, 20, vm_mkcolor(LIGHTCYAN, BLUE));

will place a box on the screen with a light cyan foreground and a blue background and clear its contents.

void vm_paintclearscreen(char attr);

Fills the screen with the *attr* video attribute and clears its contents. The location of the text cursor remains fixed.

void vm_paintclearline(char row, char attr);

Fills a row on the screen with the *attr* video attribute and clears its contents. The location of the text cursor remains fixed.

void vm_paintcleareol(char row, char attr);

Fills a row on the screen beginning at the current X location of the text cursor with the *attr* video attribute and clears its contents. The location of the text cursor remains fixed.

void vm_paintbox(char x1, char y1, char x2, char y2, char attr);

Fills a box on the screen with the *attr* video attribute. The location of the text cursor remains fixed. *vm_attrib()* is a synonym for *vm_paintbox()*.

void vm_paintscreen(char attr);

Fills the screen with the *attr* video attribute. The location of the text cursor remains fixed.

void vm_paintline(char row, char attr);

Fills a row on the screen with the *attr* video attribute. The location of the text cursor remains fixed.

void vm_painteol(char row, char attr);

Fills a row on the screen beginning at the current X location of the text cursor with the *attr* video attribute. The location of the text cursor remains fixed.

void vm_clearbox(char x1, char y1, char x2, char y2);

Clears a box on the screen using the current color video attribute. The location of the text cursor remains fixed.

void vm_clearscreen(void);

Clears the screen using the current color video attribute. The location of the text cursor remains fixed.

void vm_clearline(char row);

Clears a row on the screen using the current color video attribute. The location of the text cursor remains fixed.

void vm_cleareol(char row);

Clears a row on the screen beginning at the current X location of the text cursor using the current color video attribute. The location of the text cursor remains fixed.

void vm_fillbox(char x1, char y1, char x2, char y2, char ch);

Fills a box on the screen with the character *ch* using the current color video attribute. The location of the text cursor remains fixed.

void vm_fillscreen(char ch);

Fills the screen with the character *ch* using the current color video attribute. The location of the text cursor remains fixed.

void vm_fillline(char row, char ch);

Fills a row on the screen with the character *ch* using the current color video attribute. The location of the text cursor remains fixed.

void vm_filleol(char row);

Fills a row on the screen with the character *ch* beginning at the current X location of the text cursor using the current video attribute. The location of the text cursor remains fixed.

void vm_clrscr(void);

Clears the screen using the current video attribute and moves the text cursor to the upper-left hand corner of the screen.

void vm_clreol(void);

Clears all characters on the current row beginning at the current X location of the text cursor using the current video attribute. The location of the text cursor remains fixed.

char vm_getchxy(char x, char y);

Returns the character on the screen at the coordinate *x*, *y*.

char vm_getattrxy(char x, char y);

Returns the video attribute on the screen at the coordinate *x*, *y*.

`void vm_xgetchxy(char x, char y, char *attr, char *ch);`

Returns both the video attribute and character on the screen at the coordinate *x*, *y*.

`void vm_setattr(char attr);`

Sets the current video attribute to be used to be used next, for functions where no video attribute value may be specified (eg. *vm_putch()*).

`void vm_putattr(char x, char y, char attr);`

Sets the video attribute on the screen at the coordinate *x*, *y*. The character at that location, and the location of the text cursor both remain fixed.

`void vm_putch(char x, char y, char ch);`

Displays a character on the screen at the coordinate *x*, *y*. The previous character at that location is overwritten. The location of the text cursor remains fixed.

`void vm_puts(char x, char y, char *str);`

Displays a string on the screen at the coordinate *x*, *y*. The previous text at that location is overwritten. The location of the text cursor remains fixed.

`void vm_printf(char x, char y, const char *format, ...);`

Displays a string on the screen at the coordinate *x*, *y* using formatted output. The previous text at that location is overwritten. The location of the text cursor remains fixed. *This function is deprecated and may be removed from future versions of VidMgr.*

`void vm_xputch(char x, char y, char attr, char ch);`

Displays a character on the screen at the coordinate *x*, *y* using the video attribute *attr*. The previous character at that location is overwritten. The location of the text cursor remains fixed.

`void vm_xputs(char x, char y, char attr, char *str);`

Displays a string on the screen at the coordinate *x*, *y* using the video attribute *attr*. The previous text at that location is overwritten. The location of the text cursor remains fixed.

*void vm_xprintf(char x, char y, char attr, const char *format, ...);*

Displays a string on the screen at the coordinate *x*, *y* using the video attribute *attr* and formatted output. The previous text at that location is overwritten. The location of the text cursor remains fixed. *This function is deprecated and may be removed from future versions of VidMgr*.

*void vm_gettext(char x1, char y1, char x2, char y2, char *dest);*

Copies the text and video attributes from a box on the screen to memory. The memory required to store a copy of the box is equal to *width x height x 2*. For a box 80 columns wide and 25 rows (or lines) high, 4,000 bytes of memory are required.

*void vm_puttext(char x1, char y1, char x2, char y2, char *srce);*

Copies the text and video attributes from memory to the screen.

void vm_setcursorstyle(int style);

Sets the size of the text cursor, where supported. See Table 1.3 for a list of cursor style values that may be used with this function.

Table 1.3: Cursor style values that may be used with *vm_setcursorstyle()*.

Name	Description
CURSORHIDE	*Hides the text cursor*
CURSORNORM	*Resets the text cursor to whatever size it was when vm_init() was called*
CURSORHALF	*Set the cursor size to half that of CURSORFULL*
CURSORFULL	*Set the cursor size to a solid block*

```
void vm_printfcenter(char row, const char *format, ...);
void vm_printfbetween(char x1, char x2, char row, const char *format,
...);
void vm_xprintfcenter(char row, char attr, const char *format, ...);
void vm_xprintfbetween(char x1, char x2, char row, char attr, const
char *format, ...);
void vm_horizline(char x1, char x2, char row, char attr, char ch);
void vm_vertline(char y1, char y2, char col, char attr, char ch);
void vm_frame(char x1, char y1, char x2, char y2, char attr, char
*frame);
```

These functions are currently undocumented. *These functions are deprecated and may be removed from future versions of VidMgr*.