V6Lib Manual

Version 0.5
20 January, 1998

Jason C. Penney
jpenney@chelmsford.com

Introduction

This document is not yet complete. Only the Technical Reference for V6Lib is near finished form. This section will contain enough information to get you started with V6Lib, but will later be rewritten to be more useful.

The V6 Z-machine offers support for 8 text windows, images, sounds, menus and user stacks. Currently V6Lib doesn't support V6 menus and user stacks, but it will in the future. For a better understanding of the Z-Machine's V6 Screen model, see section 8.6 of "The Z-Machine Standards Document".

Both this manual and V6Lib are still under development.

Getting Started

The basic steps to getting started are simple. Make sure that the V6Lib files are in your include path. Include V6Defs before Parser, and include V6 after VerbLib. There are some constants that can be defined to change the behavior of V6Lib (`NOZPIC`, `NOZSND`, `MAXZPICS`, `MAXZSNDS`). They must be defined before including V6Defs.

Here is a commented shell file (based on the old shell.inf by Graham Nelson [I think]).

```
!!!! Start of file
Switches v6;
Release 0;

Constant Story "V6Shell";
Constant Headline "^An Interactive Emptiness^\
             by Jason C. Penney.^";

!Constant NOZPIC; !Uncomment if you don't want image support
!Constant NOZSND; !Uncomment if you don't want sound support

!Constant MAXZPICS x; !Uncomment if you want a dynamic image object
                      !pool.  Replace x with the size of the pool

!Constant MAXZSNDS x; !Uncomment if you want a dynamic sound object
                      !pool.  Replace x with the size of the pool


Include "V6Defs";
Include "Parser";
Include "VerbLib";
Include "V6";

Object V6Land "V6Land"
       with description "A gigantic void."
       has  light;

[ Initialise;
       V6Init(); !you can pass a ZWinStyle object to V6Init
                 !to be used as the startup style.
       location = V6Land;
       print "^^^^^Welcome...^^";
];

Include "Grammar";
end;
!!! end of file
```

Technical Reference

Class Documentation

ZPic

The ZPic class provides an interface for displaying images. To create an instance of a ZPic object, it is necessary to provide with the private property `picnum`, which should be equal to the image's number in the resource file.

```
ZPic   TitlePic
 private
      picnum 1;
```

If the constant `MAXZPICS` is defined before the inclusion of V6Defs.h a pool of dynamic ZPic objects will be created equal to `MAXZPICS`. This can be useful when an image is only going to be needed for a short time. The images number should be passed to the ZPic via `create()` (NOTE: Inform does not (at this time) handle the passing of arguments to `create` as one might expect, to overcome this, call `create` again on the new object with the intended parameters).

```
DrawTitle[ TitlePic;
      TitlePic = ZPic.create();
      TitlePic.create(1);
      . . .
      TitlePic.Draw();
      ZPic.destroy(TitlePic);
      . . .
];
```

GetWidth(), GetHeight()

Returns the images width or height (respectively) in units.

```
height = TitlePic.GetHeight();
```

Draw([y, x])

Draws the image within the active ZWindow. If no parameters are given the image displays at the current cursor position.

```
TitlePic.Draw();
```

The optional `y` and `x` parameters give an offset within the window.

```
TitlePic.Draw(1,1);
```

Erase([y, x])

Paints an area the size of the ZPic in the active ZWindow with the background colour. If no parameters are given the area is painted at the current cursor position.

```
TitlePic.Erase();
```

The optional `y` and `x` parameters give an offset within the window.

```
TitlePic.Erase(1,1);
```

ZSnd

The ZSnd class provides an interface for handling sound effects (and possibly music). To create an instance of a ZSnd object, it is necessary to provide with the private property `sndnum`, which should be equal to the sound's number in the resource file.

```
ZSnd   Creak
 private
       picnum 3;
```

If the constant `MAXZSNDS` is defined before the inclusion of V6Defs.h a pool of dynamic ZSnd objects will be created equal to `MAXZSNDS`. This can be useful when an sound is only going to be needed for a short time. The sound's number should be passed to the ZSnd via `create()` (NOTE: Inform does not (at this time) handle the passing of arguments to create as one might expect, to overcome this, call create again on the new object with the intended parameters).

```
PlayCreak[ Creak;
       Creak = ZSnd.create();
       Creak.create(3);
       . . .
       Creak.Play();
       ZSnd.destroy(Creak);
       . . .
];
```

V6Lib provides two ZSnd objects automatically, `HighBeep` and `LowBeep`. They play a high or low pitched beep sound respectively.

GetVolume()

Returns the ZSnd's current volume setting.
```
    vol = Creak.GetVolume();
```

SetVolume(vol)

Sets the volume of the ZSnd to `vol`. `vol` should be an integer from 0 to 8 or `V_MAX`. 0 is the softest the sound can be played and `V_MAX` is the loudest.
```
    Creak.SetVolume(V_MAX);
```

Play([repeats, routine])

Called with no arguments it will play the sound once.
```
    Creak.Play();
```
If `repeats` is greater than zero, the sound will repeat that many times. To repeat forever (or until stopped) pass in `RP_FOREVER` for `repeats`.
```
    Creak.Play(RP_FOREVER);
```
`routine` should be the address of a routine that will be called with no arguments once the sound has finished playing.
```
    Creak.Play(0,AfterCreak);
```

Stop()

Stops the sound if it is currently playing. (NOTE: Currently the Z-Machine only supports one sound playing at a time. Calling stop on any ZSnd objects will stop all sound. This may change in future). Currently does not stop sounds that are Fading.
```
    Creak.Stop();
```

Prepare()

Asks the interpreter to pre-load the sound into memory.
```
Creak.Prepare();
```

Finish()

If the sound has been pre-loaded into memory, it is flushed out.
```
Creak.Finish();
```

Fade(sv, fv, steps)

Fades a sound in or out. The starting volume, given as `sv`, is faded to the final volume, given as `fv`. It repeats the sound `steps` number of times to get from `sv` to `fv`.
```
Creak.Fade(5,1,12);
```

ZWindow

The ZWindow class provides an interface for controlling the windowing features of the V6 Z-Machine's screen model. V6Lib declares 8 ZWindow objects: `MainWin`, `StatusWin`, `ZWin2`, `ZWin3`, `ZWin4`, `ZWin5`, `ZWin6`, and `ZWin7`.

Activate()

Sets the ZWindow as the Active ZWindow. All output will go to the Active ZWindow.
```
MainWin.Activate();
```

GetYLoc(), GetXLoc()

Returns the current Y or X location on the screen (respectively) of the ZWindow.
```
xloc = MainWin.GetXLoc();
```

GetYSize(), GetXSize()

Returns the height or width (respectively) of the ZWindow in units.
```
height = StatusWin.GetYSize();
```

GetYCursor(), GetXCursor()

Returns the cursor's current Y or X position (respectively) within the ZWindow.
```
ycursor = StatusWin.GetYCursor();
```

GetLMargin(), GetRMargin()

Returns the Left or Right margin setting (respectively) of the ZWindow in units.
```
left_margin = MainWin.GetLMargin();
```

GetNewLineIntRoutine()

Returns the address of the ZWindow's newline interrupt routine.
```
nlroutine = ZWin3.GetNewLineIntRoutine();
```

GetIntCount()

Returns the ZWindow's interrupt count.
```
int_count = ZWin3.GetIntCount();
```

GetFontStyle()

Returns the ZWindow's current font style (NOTE: On some interpreters a combination of styles is possible, so the value may not directly equal a specific style. It is probably best to treat the value as a bitfield.)

```
    if (StatusWin.GetFontStyle() & ST_BOLD)
    {
            print “Bold is active^”;
    }
```

GetBGColour(), GetFGColour()

Returns the current background or foreground colour (respectively) of the ZWindow.

```
    colour = MainWin.GetBGColour();
```

GetFont()

Returns the current font of the ZWindow.

```
    font = MainWin.GetFont();
```

GetCharHeight(), GetCharWidth()

Returns the height or width (respectively) of a character in the ZWindow's current font, in units. For proportional fonts the interpreter should return the width of the '`0`' (zero) character.

```
    cwidth = StatusWin.GetCharWidth();
```

GetLineCount

Returns the line count of the ZWindow. (see section 8 of "The Z-Machine Standards Document" Version 1.0 for more info about the line count)

```
    lc = MainWin.GetLineCount();
```

SetLoc(row, col)

Set the ZWindow's new location on the screen to be (`row`,`col`). If either argument is 0 then only the other will be changed. The screen's top left z-pixel is (1,1).

```
    MainWin.SetLoc(1,(StatusWin.GetYSize()+1));
```

Move(rowoff, coloff)

Move the ZWindow `rowoff` units vertically and `coloff` units horizontally from it's current position.

```
    MainWin.Move(0,-3);
```

SetSize(height, width)

Set the ZWindow's size to be `height` units tall and `width` units across.

```
    StatusWin.SetSize(20,640);
```

SetCursor(row, col)

Set the ZWindow's cursor's location within the ZWindow to be (`row,col`). If either argument is 0 then only the other will be changed. The ZWindow's top left z-pixel is (1,1).

```
    StatusWin.SetCursor(1,1);
```

MoveCursor(rowoff,coloff)

Move the ZWindow's cursor `rowoff` units vertically and `coloff` units horizontally from it's current position.

```
    MainWin.MoveCursor(-5,2);
```

HideCursor(), UnHideCursor()

Makes the ZWindow's cursor invisible or visible (respectively).

```
    StatusWin.HideCursor();
```

SetMargins(lmar,rmar)

Sets the ZWindow's left margin to be `lmar` units and it's right margin to be `rmar` units.

```
    MainWin.SetMargins(5,5);
```

SetColours(fg, bg), SetBGColour(bg), SetFGColour(fg)

Controls the colour settings of the ZWindow. The first form sets the ZWindow's foreground colour to `fg`, and it's background colour to be `bg`.

```
    MainWin.SetColours(CL_WHITE, CL_BLACK);
```

The other two forms change either the background or foreground colour (respectively).

```
    StatusWin.SetBGColour(C_UNDERCUR);
```

SetFont(font)

Sets the ZWindows current font to `font.`

```
    StatusWin.SetFont(FN_COURIER);
```

SetFontStyle(style)

Sets the ZWindow's current font style to `style.` (NOTE: On some interpreters a combination of styles is possible, so the value may not directly equal a specific style. It is probably best to treat the value as a bitfield.)

```
    StatusWin.SetFontStyle(ST_BOLD | ST_ITALIC);
```

Scroll(num)

Scrolls the ZWindow `num` units. `num` may be positive or negative. (NOTE: This has no bearing on the windows scrolling attribute. See WinStyleSet.)

```
    MainWin.Scroll(MainWin.GetCharHeight()*15);!Scroll 15 lines.
```

SetNewlineIntRoutine(routine)

Sets the ZWindow's newline interrupt routine to `routine`. This routine will be called when the ZWindow's interrupt is about to be set to zero. This may be used to cause text to print around images.

```
    MainWin.SetNewlineIntRoutine(ResetMargins);
```

SetIntCount(num)

Sets the ZWindow's interrupt countdown to `num`. By default this value is zero. When set to a non zero value, it is decremented each time the ZWindow newlines. Upon reaching zero again, the newline interrupt routine will be called.

```
    !Set interrupt count for printing around SomePic
    MainWin.SetIntCount(SomePic.GetXSize()/MainWin.GetCharHeight());
```

SetLineCount(num)

Set's the ZWindow's line count to `num`. This can be used to manipulate when [MORE] is printed.

```
    MainWin.SetLineCount(25);
```

WinStyleSet(flags)

Changes the ZWindow's window style settings by overwriting the ZWindows current style settings with `flags`.

```
    ZWin2.WinStyleSet(WS_TRANSCRIPT | WS_BUFFER);
```

WinStyleOn(flags)

Changes the ZWindow's window style settings by turning on the styles passed in by `flags`, leaving all other styles alone.

```
    ZWin2.WinStyleOn(WS_SCROLL);
```

WinStyleOff(flags)

Changes the ZWindow's window style settings by WinStyleOff() turns off the styles passed in by `flags`, leaving all other styles alone.

```
    ZWin2.WinStyleOff(WS_WRAP);
```

WinStyleToggle(flags)

Changes the ZWindow's window style settings by flipping the current settings of the styles passed in by `flags`, leaving all other styles alone.

```
    ZWin2.WinStyleToggle(WS_TRANSCRIPT | WS_WRAP);
```

Erase()

Clears out the ZWindow with its current background colour.

```
    StatusWin.Erase();
```

DrawPic(pic, y, x)

Draws ZPic `pic` in the ZWindow. The `y` and `x` parameters behave exactly like the ones for ZPic.Draw(`y`,`x`).

```
    StatusWin.DrawPic(StatusBanner,1,1);
```

ErasePic(pic, y, x)

Erases ZPic `pic` in the ZWindow. The `y` and `x` parameters behave exactly like the ones for ZPic.Erase(`y`,`x`).

```
    StatusWin.ErasePic(CRose);
```

ResizeToPic(pic)

Sets the size of the ZWindow to be that of the ZPic `pic`.

```
    ZWin3.ResizeToPic(Portrait);
```

ZWinStyle

The ZWinStyle class provides an interface for controlling the layout of the screen. The library provides on ZWinStyle instance called `DefaultZWinStyle`, which emulates the Inform Library status line behavior. Each ZWinStyle should provide it's own Finish(), Init() and Update().

Activate()

Calls the active style's Finish(), then sets the the ZWinStyle as the active style, and finally, calls the new active style's Init().

```
    CRoseStyle.Activate();
```

Init()

This should be provided by the programmer. It should set up the windows and margins used by the style. It may wish to draw any background image that will reside on screen while this style is active.

Update()

This should be provided by the programmer. It will be called each turn when DrawStatusLine would have been called.

Finish()

This should be provided by the programmer. It will be called before this style is retired in favor of another. It should set all windows back to their previous positions, undo any margin changes, and such that may get in the way of the new style.

Library Defined Information

Constants

Colours

The following colours are defined as constants by the library: `C_BLACK`, `C_RED`, `C_GREEN`, `C_YELLOW`, `C_BLUE`, `C_MAGENTA`, `C_CYAN`, `C_WHITE`, `C_GREY`. `C_UNDERCUR` is also provided, it refers to the colour under the cursor. (NOTE: There has been no consensus on whether modern interpreters should use Infocom's Dos or Amiga palette. Until this is decided upon, `C_GREY` is actually a Global, to make the library work similar on both platforms. I'm pushing for the Amiga palette myself).

Fonts

The following fonts are defined as constants by the library: `FN_NORMAL`, `FN_PICTURE`, `FN_GRAPHIC`, `FN_COURIER`. To the best of my knowledge most V6 interpreters will only display `FN_NORMAL` and `FN_COURIER`. `FN_PREV` will switch to the previously used font.

Text Styles

The following text styles are defined as constants by the library: `ST_ROMAN`, `ST_REVERSE`, `ST_BOLD`, `ST_ITALIC`, `ST_FIXED`. On some interpreters a combination of styles may be possible. In these cases, setting the style to `ST_ROMAN` should clear all other styles.

Window Styles

The following constants are defined by the library: `WS_WRAP` (wrap text), `WS_SCROLL` (scroll the window if text doesn't fit), `WS_TRANSCRIPT` (text in this window should be included in the transcript, `WS_BUFFER`(buffer text before printing).

Appendix

Thanks!

I want to take this time/space to thank the following people for there
help and support and such with V6Lib:

Matt Ackeret, Neil Brown, Charlie Cole, John Holder, Stefan Jokisch, Patrick Kellum, Vincent Laviano, Howard Liu, Denise Nedley, Graham Nelson, Gunther Schmidl, Miron Schmidt, and Brian Waite.
(Sorry if I'm forgetting anybody, feel free to remind me...)

Internet Resources

There's a homepage for V6Lib at:
http://www.chelmsford.com/home/jpenney/V6Lib/

V6Lib programming questions can be asked on `rec.arts.int-fiction`. I read posts there regularly, and post there semi-regularly. By posting your questions on the newsgroup others may be able to offer assistance as well.

Bug reports, comments, questions or whatever can be sent directly to me via email. My address is `jpenney@chelmsford.com`