PowerWindows

Georg Steger

COLLABORATORS			
	TITLE : PowerWindows		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Georg Steger	July 20, 2024	

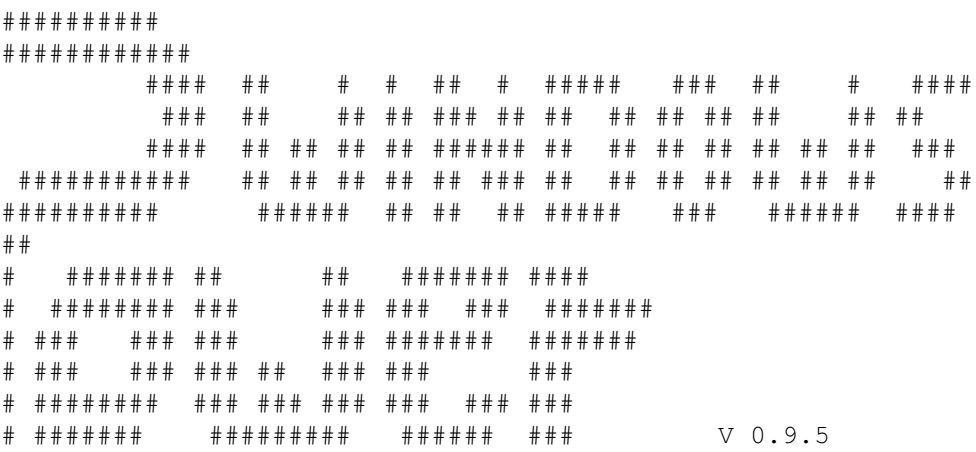
REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PowerWindows

1.1 PowerWindows V 0.9.5 Developper-Docs (C) 1997 by Georg Steger



```
                                  Introduction
                                  Important Things

                                  Screen-Checker
                                  Window-Checker
                                  Icon-Renderer

          Examples
```

1.2 Introduction

PowerWindows starting with V 0.9.5 allows you to use external Routines for doing the following things:

- Check whether a Window shall get an Iconify-Gadget or not, specify the Position of the Gadget (if you want) and render the Gadget (if you want).

- Check whether a Screen shall be patched (to allow Out-Of-Screen

Movings and Iconify-Operations) or not.

- Render the Icons (Mini-Windows) created by PowerWindows when iconifying a Window.

The first two are useful if it's impossible to specify with the Options provided by PowerWindows which Screens/Programs you want to be patched and which not.

An external Icon-Renderer allows you to change the look of the Icons, if you don't like the one "designed" by me.

1.3 Important things you have to know!

To write an external Routine for PowerWindows you should use a C-Compiler. If you know the way C-Compilers pass the Parameters to the Functions it should not be very difficult writing a Routine using an Assembler or even other Languages ("E","Pascal").

Non C-Programmers should take a look at the Source of the Startup-Codes.

Notes: o Create your Program with "NO STARTUP-CODE"

- Don't use SMALL-DATA (A4/A5 relative addressing of Data)!
- If necessary tell your Compiler to access ExecBase directly on Address 0x00000004, because you can't use it's standard Startup-Code and so a Variable ExecBase might not exist (depends on the Compiler) or might not be initialized.
- Your Routine should not eat too much Stack!
- Depending on what you are going to write, one of the Startup-Codes in the "Startup"-Directory must be linked together with your Program. This Startup-Code must have the highest Link-Priority.
- Assembler-Programmers should use the Startup-Codes too, so if in a future Release something changes, you are prepaired!
- The Prototypes of the Functions are in PWDevelopper.h .
- Assembler-Guys: Don't trash Registers D2-D7/A2-A6!

1.4 External Screen-Checkers

Read the Important Things if you haven't done it yet!!
Take a look at PWDevelopper.h !

First PowerWindows checks whether the new Screen is O.K. to be patched by

doing all the Checks described in the Config-File. If it is O.K. then it calls your Routine to do a further Check. The Routine has the following Prototype (taken from PWDevelopper.h !):

```
    WORD CheckScreen(Msg msg);
```

Everything is very similiar to the BOOPSI-Dispatcher Concept. First you have to check msg->MethodID. Depending on what you find there msg will be one of the Structures defined in the Includes. A Screen-Checker will get the following MethodIDs:

```
  CS_INIT  : msg will be struct csInit
  CS_EXIT  : msg will be struct csExit
(*) CS_CHECK : msg will be struct csCheck
```

To access msg you must cast it to the right Structure. There are some nice Macros in "PWDevelopper.h" doing this for you. Assembler-Guys will get msg at 4(sp).

Depending on what MethodID you get the following must be done:

CS_INIT : Do your Initialisation (Open libs etc.). Return CSR_OK if everything went fine, otherwise return return CSR_FAIL.

Note: You should also check the Version-Fields! Return CSR_FAIL if your Routine requires a newer Version!

CS_EXIT : Do your Cleanup (Close libs etc.). You can return anything.

CS_CHECK (*) : Return CSR_OK if you want the Screen Scr to be patched, otherwise return CSR_FAIL!

Notes: o You get CS_INIT only ONCE (when PowerWindows is starting)!

- o You get CS_EXIT only ONCE (when PowerWindows quits)!
- o (*) This Methods are multi-threaded. Watch out and don’t use any DOS Functions here.

1.5 External Window-Checkers

Read the Important Things if you haven't done it yet!!
Take a look at PWDevelopper.h !

First PowerWindows checks whether the new Window is O.K. to get an Iconify-Gadget by doing all the Checks described in the Config-File. If it is O.K. then it calls your Routine to do a further Check. The Routine has the following Prototype (taken from PWDevelopper.h !):

```
    WORD CheckWindow(Msg msg);
```

Everything is very similiar to the BOOPSI-Dispatcher Concept. First you have to check msg->MethodID. Depending on what you find there msg will be one of the Structures defined in the Includes. A Window-Checker will get the following MethodIDs:

```
  CW_INIT    : msg will be struct cwInit
  CW_EXIT    : msg will be struct cwExit
(*) CW_CHECK   : msg will be struct cwCheck
(*) CW_RENDERIG: msg will be struct cwRenderIG
```

To access msg you must cast it to the right Structure. There are some nice Macros in "PWDevelopper.h" doing this for you. Assembler-Guys will get msg at 4(sp).

Depending on what MethodID you get the following must be done:

CW_INIT : Do your Initialisation (Open libs etc.). Return CWR_OK if everything went fine, otherwise return return CWR_FAIL.

Note: You should also check the Version-Fields! Return CWR_FAIL if your Routine requires a newer Version!

CW_EXIT : Do your Cleanup (Close libs etc.). You can return anything.

CW_CHECK (*) : Return CWR_OK|CWR_ICONIFYGAD if you want the Window Win to get an Iconify-Gadget, otherwise return CWR_FAIL! You can also set the Coordinates of the Gadget by filling in the Gad_LeftEdge, Gad_TopEdge, Gad_Width, Gad_Height Entries of the Message you get. Gad_Flags must contain the Flags for the Gadget. Actually only

```
        GFLG_RELRIGHT and GFLG_RELBOTTOM are supported. Don't
        look at the Gad_?? values when you get the Message,
        because they are not initialized. PowerWindows will use
        the Coordinates only if you OR CWR_ICONIFYPOS to the
        Return-Code. Simply return:

          CWR_OK|CWR_ICONIFYGAD|CWR_ICONIFYPOS.


CW_RENDERIG (*)   : If you don't want to do the Rendering of the Iconify-
        Gadget simply return CWR_FAIL and PowerWindows will do
        it for you. If you want to do it: GadImage points to
        the Image of the Gadget (which is an IMAGECLASS Boopsi
        Object). DrawImageMsg points to an IM_DRAW Message
        (struct impDraw). It's best you take a look at the
         Example ! Don't forget to return CWR_OK!


------------------------------------------------------------------------------
Notes:  o You get CW_INIT only ONCE (when PowerWindows is starting)!

  o You get CW_EXIT only ONCE (when PowerWindows quits)!

  o (*) This Methods are multi-threaded. Watch out and don't use
    any DOS Functions here.
```

1.6 External Icon-Renderers

```
Read the  Important Things  if you haven't done it yet!!
Take a look at  PWDevelopper.h !

------------------------------------------------------------------------------

You can create a Routine that does the Icon-Rendering for PowerWindows.
You should at least print the Window-Title somewhere, otherwise it's hard
to say which Window a certain Icon belongs to. This is the Prototype:


    WORD CheckScreen(Msg msg);


Everything is very similiar to the BOOPSI-Dispatcher Concept. First you have
to check msg->MethodID. Depending on what you find there msg will be one
of the Structures defined in the Includes. A Screen-Checker will get the
following MethodIDs:

  RI_INIT       : msg will be struct riInit
  RI_EXIT       : msg will be struct riExit
  RI_INITICON   : msg will be struct riInitIcon
  RI_RENDERICON : msg will be struct riRenderIcon
(*) RI_EXITICON   : msg will be struct riExitIcon

To access msg you must cast it to the right Structure. There are some nice
```

Macros in "PWDevelopper.h" doing this for you. Assembler-Guys will get msg at 4(sp).

Depending on what MethodID you get the following must be done:

RI_INIT : Do your Initialisation (Open libs etc.). Return RIR_OK if everything went fine, otherwise return RIR_FAIL.

Note: You should also check the Version-Fields! Return RIR_FAIL if your Routine requires a newer Version!

RI_EXIT : Do your Cleanup (Close libs etc.). You can return anything.

RI_INITICON : You get this Message when PowerWindows creates an Icon. ParentWin points to the Window that is being iconified. IconWidth and IconHeight contains the Dimensions of the Icon-Window that is going to be opened. You can change IconHeight. Don't touch IconWidth. In UserData you can store something (for example a Pointer to something you alloced). UserData is not something global, but "local" to the Icon that will be created. The Message-Structures riRenderIcon and riExitIcon will contain the Value you have stored here. You can return anything. The Icon will be created regardless what you return!

RI_RENDERICON : Render the Icon. You must render into Icon->RPort! ParentWin points to the iconified Window. Icon points to the Icon-Window. UserData will contain what you have inserted there during RI_INITICON. You should at least do some minimal Rendering even if you were not able to alloc/load something during RI_INITICON. You get this message only once for each Icon. In some cases Power-Windows adds some invisible System-Gadgets to the Icon. For example when you have set ICONDEPTH to NORMAL, PW will add an invisible Depth-Gadget in upper right Edge of the Icon. To check this, you have to scan through the GadgetList of the Icon:

```
iwin=((struct riRenderIcon *)msg)->Icon;
gad=iwin->FirstGadget;
while (gad)
{
    if ((gad->GadgetType&GTYP_SYSTYPEMASK)==GTYP_WDEPTH)
    {
         /* we have found a Depth-Gadget */
         /* Now let's get the coordinates of it */
```

```
                /* and ... */

                x1=gad->LeftEdge;
                if (gad->Flags&GFLG_RELRIGHT)
                 x1+=(iwin->Width-1);
                y1=gad->TopEdge;
                if (gad->Flags&GFLG_RELBOTTOM)
                 y1+=(iwin->Height-1);
                x2=x1+gad->Width-1;
                if (gad->Flags&GFLG_RELWIDTH)
                 x2+=(iwin->Width);
                y2=y1+gad->Height-1;
                if (gad->Flags&GFLG_RELHEIGHT)
                 y2+=(iwin->Height);

                /* ... render something special there */

                RectFill(iwin->RPort,x1,y1,x2,y2);
            }
            gad=gad->NextGadget;
        }
```

You can also change the Position of the Gadgets if you like. To do this use SetGadgetAttrs().

You can return anything.

RI_EXITICON (*) : You get this Message before PowerWindows kills an Icon (when deiconifying a Window or quitting the Program). Free the Resources you have alloced during RI_INITICON and stored in UserData.

Notes: o You get RI_INIT only ONCE (when PowerWindows is starting)!

o You get RI_EXIT only ONCE (when PowerWindows quits)!

o (*) This Methods are multi-threaded. Watch out and don't use any DOS Functions here.

1.7 Example Source Codes

WindowChecker.c Very similiar to the internal Routine of PowerWindows
IconRenderer.c Simple Rendering. Icons of WB Windows look different