Wimp Topics - Starting up with a Title

Follow Nathan Micholey's tips to add a bit of style to your Wimp programs

There are many programs nowadays which start up displaying a decorative title page, which usually gives information on the program's authors and licensee. FireWorkz and Impression, for example, both have title pages which are displayed while they are initialising. In this month's Wimp Topics, a program to create and display a title page with fonts and sprite graphics is described which can be incorporated into your own programs so that they too can have an attractive start-up banner.

You will need first to design a largish sprite for your title page which is about 200 pixels square in mode 20, or 200*100 pixels in mode 12. Save this in a file called TitleSpr inside the application directory. Now design a template called Title using FormEd or another template editor. This should have at least two icons; one which contains the sprite and one which contains the name of your program in a large anti-aliased font of about 20 points.

It is worth remembering that if your program is to be used on other systems, you can only rely on the fonts Corpus, Homerton and Trinity being present, so you should restrict your font choice to one of these. If your program has to be RISC OS 2 compatible, you can't even rely on these being present, in which case the icons will appear blank (it's probably a good idea to check the OS version and use only the system font if RISC OS 2). Now that the Risc PC is available, which allows anti-aliased fonts to be used in place of the system font, it is good practice in general not to use anti-aliased fonts in templates, since they will override the user's preferred font on that machine. For a title page, though, we can make an exception since we want to use a larger font.

In order for your program to display the fonts in a template, you will need some extra code. Firstly, before we load templates, we need to call PROCinit_fonts which is defined as:

```
DEF PROCinit_fonts
LOCAL f%
DIM fontdata% &100
FOR f%=0 TO 252 STEP 4
 fontdata%!f%=0
NEXT
ENDPROC
```

This allocates a 256-byte array which is used by Wimp_LoadTemplate to record the fonts that are used by the template that has been loaded. When a font is encountered, a font handle (in the range 0-255) is allocated, and the byte at fontdata%+fonthandle is incremented. If several templates are loaded then they can all use the table pointed to by fontdata% which will contain the counts of all font handles used in the program's templates. Thus the command which loads a template will look like this:

```
SYS "Wimp_LoadTemplate",,block%,indir
%,indir%+ind%,fontdata%,a$
```

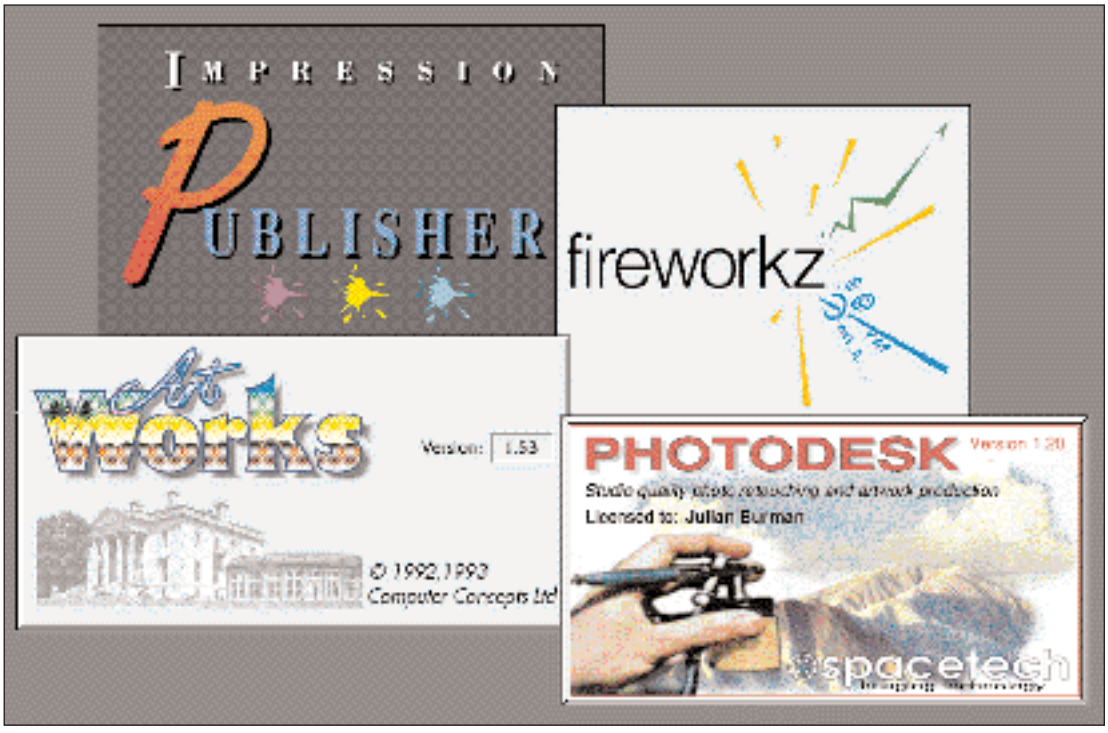
When your program exits you will need to call PROCclosefonts which is defined as:

```
DEF PROCclosefonts
LOCAL f%
FOR f%=0 TO 255
 WHILE fontdata%?f%>0
  fontdata%?f%-=1
  SYS "Font_LoseFont",f%
 ENDWHILE
NEXT
ENDPROC
```

This ensures that all the font handles that you have used are disposed of correctly.

For our example program we will assume that RISC OS 3 is being used, which gives us the opportunity to find out the storage requirements of a window and its indirected icons from the template file before loading it. This is done by calling Wimp_LoadTemplate with 0 in R1 instead of a pointer to the block. It then returns the window size and indirected data size in R1 and R2, so we can create a function:

```
DEF FNcreatewindow(a$,sprites%)
LOCAL hand%,ind%,indir%
SYS "Wimp_LoadTemplate",,0,indir%,0,f
ontdata%,a$ TO ,hand%,ind%
IF hand%>&300 ERROR 0,"Template  "+a$
```



Some examples of commercial start-up screens

```
+" is too big to load"
    DIM indir% ind%
    SYS "Wimp_LoadTemplate",,block%,indir
%,indir%+ind%,fontdata%,a$
    block%!64=sprites%
    SYS "Wimp_CreateWindow",,block% TO ha
nd%
    =hand%
```

which requires an array to be set up before it is used:

```
    DIM block% &300
```

Of course, your block can always be larger than this if any of your windows require it.

FNcreatewindow returns the window handle of the template whose name was passed in a$. It sets the window s sprite pointer to the value in sprites%, which should be -1 if you have no sprite pool. This allows us to use different sprite pools with each window.

A sprite pool can be loaded like this:

```
    DEF FNloadsprites(a$)
    LOCAL a%,l%,s%
    a%=OPENINa$
    l%=EXT#a%+32
    CLOSE #a%
    DIM s% l%:!s%=l%
    SYS  OS_SpriteOp ,&10A,s%,a$
    =s%
```

where a$ is the filename and the returned value is a pointer to the sprite pool created.

We can now create our title page like this:

```
    PROCinit_fonts
    sprites%=FNloadsprites("<TitlePage$Di
r>.TitleSpr")
    SYS "Wimp_OpenTemplate",,"<TitlePage$
Dir>.Templates"
    titlehand%=FNcreatewindow("Title",spr
ites%)
    SYS "Wimp_CloseTemplate"
```

We now need some quite lengthy procedures to open the window at the centre of the screen, regardless of the desktop screen mode:

```
    DEF PROCget_screen_size(RETURN x%,RET
URN y%)
    LOCAL xp%,yp%
```

```
    SYS "OS_ReadModeVariable",-1,4 TO ,,x p%
    SYS "OS_ReadModeVariable",-1,5 TO ,,y p%
    SYS "OS_ReadModeVariable",-1,11 TO ,, x%
    SYS "OS_ReadModeVariable",-1,12 TO ,, y%
    x%=(x%+1)<<xp%:y%=(y%+1)<<yp%
    ENDPROC

    DEF PROCget_window_size(block%,RETURN
    x%,RETURN y%)
    x%=block%!8-block%!0:y%=block%!12-blo
ck%!4
    ENDPROC

    DEF PROCset_window_position(block%,mi
nx%,miny%,maxx%,maxy%)
    !block%=minx%:block%!4=miny%:block%!8
=maxx%:block%!12=maxy%
    ENDPROC

    DEF PROCopen_window_at_centre(hand%)
    LOCAL x%,y%,wx%,wy%
    !block%=hand%
    SYS "Wimp_GetWindowState",,block%
    PROCget_screen_size(x%,y%)
    PROCget_window_size(block%+4,wx%,wy%)
    x%=x%>1:y%=y%>1
    wx%=wx%>1:wy%=wy%>1
    PROCset_window_position(block%+4,x%-w
x%,y%-wy%,x%+wx%,y%+wy%)
    block%!28=-1
    SYS "Wimp_OpenWindow",,block%
    ENDPROC
```

PROCget_screen_size uses OS_ReadModeVariable to get the Eigen values of the current screen mode and the number of pixels across and down. It then uses these to calculate the screen size in OS units. PROCget_window_size reads the window size from the block returned from Wimp_GetWindowState and PROCset_window_position positions it in the correct place before opening.

Now our window is open, we must leave it open for a predefined time and then close it. It is also useful to close the title page when it is clicked on to prevent it becoming a nuisance. This can be done by using Wimp_PollIdle with a suitable delay specified in R2. If it is called initially with bit zero of the mask unset, then after the delay period a null event will be returned. At this point (or earlier if you receive a mouse click event over the title window), you can close the window and set bit zero of the mask, so that in future the call will act exactly as Wimp_Poll.

We begin by finding the current monotonic time and adding a short delay in centiseconds to it:

```
    SYS "OS_ReadMonotonicTime" TO time%
    time%+=500
```

This code is a little simplistic as it fails to deal with wrap-around, where incrementing the timer will make the value of time% greater than the maximum allowable integer. We will overlook this for now as the probability of this actually happening is tiny. The poll loop will then look something like this (assuming mask% has been initialised with bit zero clear):

```
    REPEAT
    SYS "Wimp_PollIdle",mask%,block%,ti
me% TO e%
    CASE e% OF
    WHEN 0:PROCclose_title
    WHEN 6:IF block%!12=titlehand% PRO
Cclose_title ELSE PROCdo_poll(e%,blo
ck%)
    OTHERWISE:PROCdo_poll(e%,block%)
    ENDCASE
    UNTIL quit%=TRUE
```

You should create the procedure PROCdo_poll yourself to deal with any events your program needs to receive. Its parameters are e%, which is the event code and block% which points to the block returned from Wimp_Poll.

You will also need the following procedure:

```
    DEF PROCclose_title
    !block%=titlehand%
    SYS "Wimp_DeleteWindow",,block%
    titlehand%=0:mask%=mask% OR 1
    PROCclosefonts
    ENDPROC
```

to get rid of the title page window and set bit zero of the poll mask. If you do use fonts in other templates, you should move the call to PROClose_fonts to another part of your program.

As it stands, this implementation has some shortcomings; for example, it does not free the memory used by the title page's sprite. If you are feeling adventurous, you could try adapting it to use Alan Wrigley's heap manager (Wimp Topics 7:5) so that you can free this memory for other purposes. This implementation also makes it difficult for you to use null events for other purposes. The best way around this would be to leave bit zero of the poll mask unset,

and to set a flag, say titleopen%, during the period the title window is open. Use a CASE statement to call Wimp_PollIdle as above if this flag is set, or Wimp_Poll if not. Then use the same flag on receiving null events to determine whether to call PROCclose_title or some other procedure.

This month's magazine disc contains an application illustrating the principles described in this article.