Technical Note PR22

pIdle Proc (or how to let users know what's going on during print time...)

CONTENTS

This Technical Note discusses how to defensively program a `pIdle` procedure to work with the majority of print drivers in existence today, and how to install it at print time.

[Apr 01 1991]

Introduction

When using a `pIdle` procedure at print time, there are a few things that one should remember to be compatible with the printer drivers that are available today. This Technical Note discusses installing a `pIdle` procedure at the right time and the things to remember when writing one.

Back to top

Installing The pIdle Proc

Let's start by installing the `pIdle` procedure at the right time. You must install your `pIdle` procedure into the print record before calling `PrOpenDoc`. If you do not install your `pIdle` procedure before your call to `PrOpenDoc`, the printer driver does not give the application's `pIdle` procedure any time. The following code fragments demonstrate installing the `pIdle` procedure in the right place:

MPW Pascal

```
    << more print loop would appear above, see Technote #161 for details >>

    {**  Install a pointer to your pIdle proc into your print record. **}

    PrintingStatusDialog := GetNewDialog(257, NIL, POINTER(-1));
    thePrRecHdl^^.prJob.pIdleProc := @checkMyPrintDialogButton;

    thePrPort := PrOpenDoc(thePrRecHdl, NIL, NIL);
```

MPW C

```
    << more print loop would appear above, see Technote #161 for details >>

/**  Install a pointer to your pIdle proc into your print record.  **/

PrintingStatusDialog = GetNewDialog(257, nil, (WindowPtr) -1);
(**thePrRecHdl).prJob.pIdleProc = checkMyPrintDialogButton;

thePrPort = PrOpenDoc(thePrRecHdl, nil, nil);
```

For a complete printing loop that handles errors at print time and makes all of the right calls to the Printing Manager, refer to Technical Note #161, A Print Loop That Cares....

Back to top

Things To Remember About pIdle Procedures

It is extremely important to design and code your `pIdle` procedure as defensively as possible, thereby making sure that it works with as many printer drivers as possible. This section details a few things to remember when creating `pIdle` procedures.

Saving And Restoring The Current Port

It is extremely important to save the printer driver's `GrafPort,` upon entry to your `pIdle` procedure and restore it upon exit. Why? If you do not, the printer driver would draw into the `GrafPort` of your dialog box instead of its `GrafPort,` which will cause some bad results. To save the printer's `GrafPort`, you should call `_GetPort` when entering your procedure. Before you exit your procedure, you would call `_SetPort` to set the port from your dialog box back to the printer driver's `GrafPort` (i.e., the one you saved with `_GetPort`).

Saving And Restoring The Printer Driver's Resources

If the application changes the resource chain within it's `pIdle` procedure, you want to save and restore the printer driver's resource chain. Why? Some printer drivers assume that their resource chain does not change, but this may not be true when the driver calls the `pIdle` procedure installed by the application at print time. To accomplish this task, call `_CurResFile`, saving the ID of the printer driver's resource file at the beginning of your `pIdle` procedure. When you exit from your `pIdle` procedure, restore the resource chain back to the printer driver's resource chain with a call to `_UseResFile`.

At this point, you might be wondering what might change the resource chain. If you called `_OpenResFile` or `_UseResFile` (anything that would change the value of the low memory global `TopMapHdl`) within the application's `pIdle` procedure, the chain would be changed. If you are not changing the resource chain, these calls would not be needed.

Handling Errors From Within A pIdle Procedure

You should avoid calling `PrError` within your `pIdle` procedure; errors that occur while it is executing are usually temporary, and serve only as internal flags for communication within the printer driver - they are not intended for the application. If you absolutely must call `PrError` within your idle procedure, and an error occurs, never abort printing within the idle procedure itself. Wait until the last called printing procedure returns, then check to see if the error still remains. Attempting to abort printing within an idle procedure is a guarantee of certain death.

Canceling Or Pausing The Printing Process

If you install a procedure for handling requests to cancel printing, with an option to pause the printing process, beware of timeout problems when printing to the LaserWriter. Communication between the Macintosh and the LaserWriter must be maintained to prevent a job or a wait timeout. If there is not any communication for a period of time (over two minutes), the printer times out and the print job terminates due to a wait timeout. Or, if the print job requires more than three minutes to print, the print job terminates due to a job timeout. Since, there is not a good method to determine to what type of printer an application is printing, it is probably a good idea to document the possibility of a LaserWriter timing out for a user who chooses to select "pause" for over two minutes.

Some Printer Drivers Do Not Support pIdle Procedures

Some printer drivers do not support `pIdle` procedures, as they prefer to handle the `pIdle` procedure in their own manner without giving an application's `pIdle` procedure any time. This situation should not be a problem as long as you do not assume that your `pIdle` procedure is always called at print time. Therefore, you should only create your `pIdle` procedure to display the dialog box and respond to a user pausing, continuing, or canceling a print job.

Back to top

Conclusion

When installing your `pIdle` proc, it must be installed before the application calls `PrOpenDoc`. You want to make sure that you save and restore the `GrafPort`, upon entry and exit of your `pIdle` procedure, to make sure that the printer driver will image into the correct port during the print job. Finally, if you are changing the resource chain by calling `_OpenResFile` or `_UseResFIle`, you want to make sure that you save and restore the resource chain.

Back to top

References

Inside Macintosh , Volume II, The Printing Manager

Technical Note M.IM.PrintLoop -- A Print Loop That Cares...

Back to top

Downloadables

Acrobat version of this Note (K). Download
