Using the Colour Picker

David Spencer shows how your programs can use the new RISC OS 3.5 Colour Picker.

RISC OS 3.5 includes a standard Colour Picker which is used by the likes of Draw to select colours. There are four main reasons why your own Wimp applications should also use this in preference to any other method:

1. It gives a professional feel to an application.

2. It is more economic in terms of code space and development time.

3. It provides a consistent user interface across applications.

4. It will automatically cope with future extra colour models such as a Pantone colour selector.

The purpose of this article is to show just how the Colour Picker can be utilised by programmers, and give a simple example. Whilst the information is relevant to any programming language, the emphasis is towards use with Basic, as this is still the language used by most non-professional developers.

THE BASIC THEORY

The Colour Picker is invoked from a host application by issuing a SYS call to open the picker dialogue box. It then registers itself as a Wimp filter which means that it can intercept Wimp_Poll calls and carry on its business independently of the host until something interesting happens (such as the user picking a colour). When a colour is chosen, the choice is returned to the user via a Wimp message. A second Wimp message is used to request that the host issues a SYS call to close the picker box, for example after the user has clicked Select on OK. It is also possible to monitor colour changes as the user makes them via a third message. The host can choose to receive details of the current colour selection either each time the user finishes dragging one of the bars, or continuously during a drag. Obviously, which method you choose will depend on the application. For example, when setting the background colour of a DTP document frame then you probably don t need to know the colour at all until the user clicks on OK, whilst a drawing package may want to respond as the user plays around with the colour selection.

Despite the fact that the Colour Picker supports several colour models, such as RGB and HSB, the colour returned to the host are always given as RGB values, because these are the most commonly used form within RISC OS. However, to allow the Colour Picker to be re-opened in the exact state it was left in, it also returns a set of parameters that are specific to the colour model in use, and can be treated as a black box by the host. The collection of RGB colour and these additional parameters is termed a colour block , and takes the following form:

?block	0
block?1	Red component (0-255)
block?2	Green component (0-255)
block?3	Blue component (0-255)
block!4	Length of rest of block
block!8	Colour model number
block!12	Model specific data

-

Although the host application is only likely to need the red, green and blue components, it should save the rest of the block for next time the Colour Picker is called. The total length that needs to be saved is given by the value at (block!4) + 8 (this to allow for the first two words). The first time the Colour Picker is called in a given context, it is sufficient simply to pass a starting RGB value, and set the word at block!4 to zero. This will default to the RGB colour model.

DOWN TO WORK

Having described the principles, and the format

of colour blocks, it is time to move on to the actual calls.

A Colour Picker dialogue box is opened by the call:

```
SYS "ColourPicker_OpenDialogue",
      flags,block TO handle
```

Currently, flags can only be set to 0 to indicate a static dialogue box, or 1 for a transient box, which is closed automatically when OK or Cancel are clicked on. Typically, transient boxes are used when they form part of a menu tree. The variable block points to a parameter block with the following format, detailing the dialogue box:

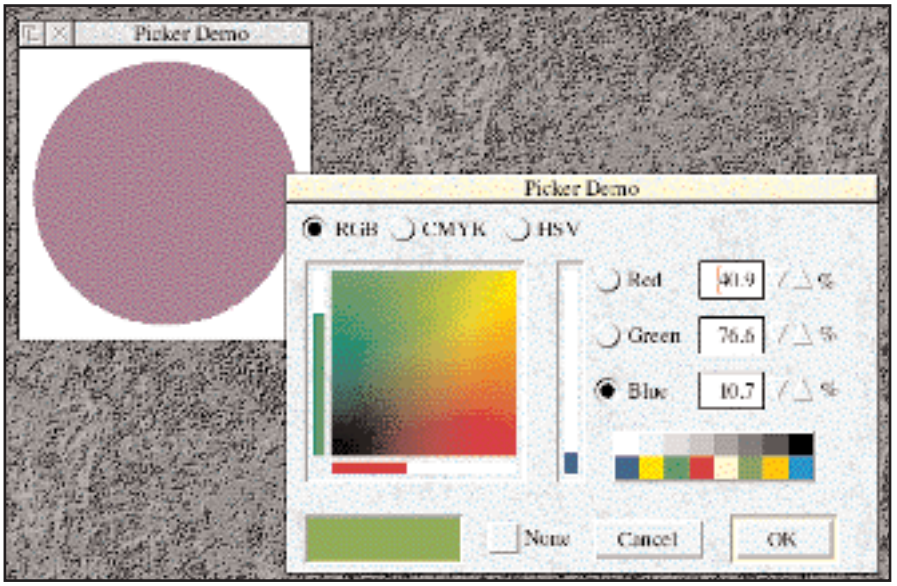
```
!block      option flags
block!4     pointer to title (0 for default)
block!8     x0
block!12    y0
block!16    x1
block!20    y1
block!24    xscroll
block!28    yscroll

block!32    the colour block starts here
..
```

The option flags should not be confused with the flags already described. If bit 0 of the option flags is set, then the picker box will have a none option, and if bit 1 is set, then this none option will be selected by default. Bits 2 and 3 define how often feedback is passed to the host. If both bits are zero it means that the host is only ever informed when OK is clicked. If bit two is set it will inform the host whenever the colour selection is made, whilst setting bit 3 will inform the host whenever the colour is changed, including in the middle of a drag. All the other option flags must be set to zero.

The pointer to the picker box title should be self explanatory - with 0 using the default of Colour choice . The next six parameters determine the position of the picker box in normal Wimp terms, although in practice, x0 and y1 should be set to the coordinate of the top-left corner, y0 should be &80000000, x1 should be &7FFFFFFF, and both xscroll and yscroll should be 0. This will ensure that the picker box is fully visible. Finally, the block contains the arbitrary length colour block described above.

Once the picker box has been created, a handle is returned to the host. This can be used to identify the picker box in the case of one application allowing several picker boxes to be open at once.



Running the example program.

When the user finally chooses a colour by clicking on OK, the host is sent the message number &47700. This is a standard Wimp message block, with the following data:

```
polldata+20     picker box handle
polldata+24     flags
                (bit 0 set if none chosen)
polldata+28     colour block starts here
..
```

As already said, the RGB colour is read from the first word of the colour block (polldata+28). If the host has opted for continuous updates as the user chooses, these are returned in a similar way to the final choice above, except that the message number is &47701, and the

flags are extended so that bit 1 is set if a drag of a colour bar is currently in progress.

The host can at any time find the state of a picker box using the call:

```
SYS "ColourPicker_ReadDialogue",handle,
block
```

which will fill the buffer pointed to by block with data in the same format used to create the picker box. However, the colour block returned will be that at the present time, and not when the picker box was created. If block is set to zero, then the call instead returns in register R1 the length that the buffer needs to be.

The host can also update the picker box whilst it is open using the call:

```
SYS "ColourPicker_UpdateDialogue",
changes,handle,block
```

The setting of bits in changes determines what attributes are changed:

- Bit 0 Change presence of none button
- Bit 1 Change state of none button
- Bit 2 Change continuous report option
- Bit 3 Change position
- Bit 5 Change title
- Bit 6 Change RGB colour
- Bit 7 Change whole colour model

handle specifies the picker box to alter, and block points to a description block as used to create the picker box, except that only the items being changed need to be present.

A picker box can be closed at any time using the call:

```
SYS "ColourPicker_CloseDialogue",0,handle
```

where handle is as returned when the picker box was created. When the user clicks Select on OK or Cancel of a static picker box, the Colour Picker sends the host a message &47702, with polldata+20 containing the picker box handle. The host should then make the above call to close the picker box. Transient picker boxes are closed automatically.

AN EXAMPLE PROGRAM

Listing 1 is a fully self contained example program, from which it should be easy to see how the Colour Picker is used. When run, it opens a fixed size window containing a circle. Clicking anywhere in the window will open the colour picker allowing the colour of the circle to be changed, although any changes are only reflected when you click on OK. If None is selected then an outline circle is shown. The application quits as soon as the main window is closed.

This month's magazine disc contains a slightly modified version of the example program which implements the colour picker as part of a menu structure, and also allows for continuous update of the circle's colour.

```
 10 REM > !RunImage
 20 REM Version   A 1.0
 30 REM Author    David Spencer
 40 REM RISC User July 1994
 50 REM Program   Subject to Copyright
 60 REM           Not Public Domain
 70 :
 80 DIM taskid% 4:$taskid%="TASK"
 90 DIM messages% 12,cblock% 100
100 DIM name% 12:$name%="Picker Demo"
110 colour%=-1:pick%=0:messages%!0=&47
700
120 messages%!4=&47702:messages%!8=0
130 !cblock%=0:cblock%!4=0
140 SYS "Wimp_Initialise",300,!taskid%
,name%,messages%
150 DIM q% 256,buffer% &200:main%=FNcw
indow
160 !q%=main%:SYS "Wimp_OpenWindow",,q
%
170 ONERROR PROCerrorbox:SYS "Wimp_Cre
ateMenu",,-1
180 REPEAT
190   SYS "Wimp_Poll",1,q% TO A%
200   CASE A% OF
210   WHEN 1:PROCredraw_window(!q%)
220   WHEN 2:SYS "Wimp_OpenWindow",,q%
230   WHEN 3:PROCfinish:END
240   WHEN 6:PROCopenpicker(!q%,q%!4)
250   WHEN 17,18:PROCreceive(q%)
260   ENDCASE
270 UNTIL FALSE
280 :
290 DEF PROCopenpicker(x,y)
300 IF pick%=0 THEN
310 IF colour%=-1 !q%=3 ELSE !q%=1
320 q%!4=name%:q%!8=x:q%!12=&80000000
```

```
 330 q%!16=&7FFFFFFF:q%!20=y:q%!24=0:q%
!28=0
 340 FOR F%=0TOcblock%!4+7
 350 q%?(32+F%)=cblock%?F%:NEXT
 360 SYS "ColourPicker_OpenDialogue",0,
q% TO pick%
 370 ENDIF
 380 ENDPROC
 390 :
 400 DEF PROCreceive(q%)
 410 CASE q%!16 OF
 420   WHEN 0:PROCfinish:END
 430   WHEN &47700:PROCchoice(q%)
 440   WHEN &47702:SYS "ColourPicker_Cl
oseDialogue",0,q%!20:pick%=0
 450 ENDCASE
 460 ENDPROC
 470 :
 480 DEF PROCfinish
 490 IF pick% SYS "ColourPicker_CloseDi
alogue",0,pick%
 500 SYS "Wimp_CloseDown"
 510 ENDPROC
 520 :
 530 DEF PROCchoice(q%)
 540 FOR F%=0TOq%!32+7
 550 cblock%?F%=q%!(28+F%):NEXT
 560 IF (q%!24 AND 1) THEN
 570 colour%=-1
 580 ELSE
 590 colour%=!cblock%
 600 ENDIF
 610 SYS "Wimp_ForceRedraw",main%,0,-40
0,400,0
 620 ENDPROC
 630 :
 640 DEF FNcwindow
 650 q%!4=400:q%!8=400:q%!12=800:q%!16=
800
 660 q%!20=0:q%!24=0:q%!28=-1:q%!32=&87
000002
 670 q%?36=7:q%!37=2:q%!38=7:q%!39=0:q%
!40=0
 680 q%!44=0:q%!48=-400:q%!52=400:q%!56
=0
 690 q%!60=9:q%!64=&3000:q%!68=0:q%!72=
0
 700 $(q%+76)="Picker Demo":q%!88=0
 710 SYS "Wimp_CreateWindow",,q%+4 TO
H
```