Debug

Paul Manias

COLLABORATORS

	TITLE : Debug		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Paul Manias	July 26, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Debug

1.1 Debug Documenation

```
MODULE DOCUMENTATION
Name:      DEBUG
Version:   0.9 Beta.
Date:      December 1997
Author:    Paul Manias
Copyright: DreamWorld Productions, 1996-1997.  All rights reserved.
Notes:     The Debug functions are located in the DPKernel.  They are
           only activated when the debugger "IceBreaker" is loaded.

CHANGES V0.9B
Renamed: ErrorMessage() to ErrCode()

Edited:  ErrCode()
         DebugMessage() "see also"

Added:   DPrintF()
```

1.2 Debugging Functions

```
 FUNCTIONS
 These functions are mapped in the dpkernel.

 DebugMessage()
 DPrintF()
 ErrCode()
 StepBack()
```

1.3 Debug: DebugMessage()

```
FUNCTION
Name:     DebugMessage()
Short:    Send a message to the debugger.
Synopsis: void DebugMessage(LONG Type [d7], char *String [a5]);
```

```
DESCRIPTION
Sends  a  message  to the debugger if it is active.  If the debugger is not
active then this function does nothing.

This  function  is intended for use in the system modules, but you may also
use  it  in  standard programs to send your own debug messages.  If this is
the case then you will want to use the DBG_Message type and supply a String
pointer.  The message will show up in bold text, so you can easily identify
it in the debug output.

MODULE PROGRAMMERS
You  will find that there is a debug type for all initialisation functions.
If you are going to use DebugMessage() inside your module it must be called
at  the  start  of each function, as the debugger will want to pick up your
parameters.   You  may also use the STEP flag in the Type parameter so that
the  debugger  can  provide tree-formatted output.  If this is the case you
will also need to call StepBack() at the end of your function.

INPUTS
Type   - One of the debug codes as described in games/debug.i.
String - Optional string used by some debug codes such as DBG_Message.

SEE ALSO
Debug: ErrCode()
```

1.4 Debug: DPrintF()

```
FUNCTION
Name:     DPrintF()
Short:    Send a formatted string to the debugger.
Synopsis: void DPrintF(const BYTE *Array [a5], ...);

DESCRIPTION
The DPrintF() function follows the same functionality and rules as the ANSI
PrintF().   The  only  difference  is  that it prints directly to the debug
window.   Due  to  internal limits your string is limited to a total of 256
bytes  output, although you should keep everything within 60 bytes to avoid
running  onto  a  second  row.   You  can  supply a maximum amount of 5 "%"
parameters to this function.

EXAMPLE
The  following  example  will print the default width of a Screen object to
the debug window.

void main(void)
{
   if (Screen = Get(ID_SCREEN)) {
      if (Init(Screen,NULL)) {
         DPrintF("The width of the screen is: %d", Screen->Width);
      }
      Free(Screen);
   }
}
```

```
INPUTS
Array - The first member of the array must be a string.  All other
        members of the array must be supplied according to the string
        formatting, 1 parameter for every % symbol that you have used.

SEE ALSO
Debug: DebugMessage()
```

1.5 Debug: ErrCode()

```
FUNCTION
Name:     ErrCode()
Short:    Send an error message to the debugger.
Synopsis: LONG ErrCode(LONG ErrorCode [d0]);

DESCRIPTION
Sends an error code to the GMS debugger, if it is currently active.  If the
debugger is not active then this function does nothing.

This  function  is intended for use in the system modules, but you may also
use  it  in  standard  programs to send your own error messages.  The error
will  show  up in bold text so that you can easily identify it in the debug
output.

INPUTS
ErrorCode - Standard GMS error code as described in dpkernel.i.

RESULT
Returns the same error code that you provided.

SEE ALSO
Debug: DebugMessage()
```

1.6 Debug: StepBack()

```
FUNCTION
Name:     StepBack()
Short:    Steps back the debugging tree in IceBreaker.
Synopsis: void StepBack(void);

DESCRIPTION
This  function  is  intended  for use by system modules.  You may use it in
your  own programs if you want to use IceBreaker’s tree feature to aid your
output.

StepBack()  has  to  be  used  in  conjunction  with  the  STEP flag in the
DebugMessage()  function.  See DebugMessage() for more information on this.
Any  time DebugMessage() is called with the STEP flag you will need to call
StepBack()  as  this  is  the only way to get the tree back to the position
before  you  altered  it.   If you forget to call StepBack() then the debug
tree will be permanently out of position.
```

```
SEE ALSO
Debug: DebugMessage()
```