Default

Paul Manias

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

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Default

1.1 IceBreaker V1.1

I C E B R E A K E R

VERSION 1.1

By Paul Manias

1.2 Introduction to IceBreaker.

INTRODUCTION

IceBreaker is the official debugger for GMS programs. Although very simple, it is one of the smallest and most powerful run-time debuggers that you can use on the Amiga. The power behind IceBreaker is that it does not have to patch any functions to tell you what your program is doing, instead the GMS functions are already written to support debug output! This removes the unfortunate kludginess of system debuggers like SnoopDOS. IceBreaker supports all functions except for those used in constant loops (eg Draw()) and those that are called so often it would become annoying (eg FindDPKTask()).

IceBreaker also supports detection of errorcodes which are shown in bold if they occur. If you run a program that doesn't work and you want to know why, try running IceBreaker to find out what is wrong. Often this will be enough for you to fix the problem.

If you've ever had problems attempting to debug hardware hacking games in the past, IceBreaker will be a great help to you. I hope you get a lot out

of its functionality.

1.3 IceBreaker requires...

REQUIREMENTS

IceBreaker requires the Games Master System, that's all. There's no special installation procedure, just run it from CLI. If you have KingCON then use the IceBreakerCon version.

1.4 How to use IceBreaker.

USAGE

There are two ways that you can run IceBreaker. The first is from a shell, simply by typing "GMSDev:Utils/IceBreaker/IceBreaker". The other way is to redirect the debug information to somewhere else. Here are some examples.

```
1.> IceBreaker >HD1:Game.dbg    <Output to a file on hard drive>

2.> IceBreaker >KCON:           <Output to a scrolling KCON: window>
```

Be aware that if your program crashes while IceBreaker is writing to a file on disk, it could invalidate the drive if the GMS program crashes. For this reason it might be better to write the output to RAM: or DF0:.

You can also redirect the output out to another machine that is networked to your computer, either on the parallel or serial ports. A printer is also another possibility if you think it will be useful! The advantage of remote output is that you can view what is happening as it occurs, even when the screen is taken over. Since most GMS programs are screen-based, it can be a great help.

If you have KingCON then it is highly recommended that you make use of this program to view output when using the CLI, as you will have a scroll bar to view debug history.

As all GMS programs are screen-based, you will have to use LEFT AMIGA+M to go back to Workbench if you want to view debug output.

1.5 How to program with IceBreaker.

PROGRAMMING

Programming with IceBreaker is EASY and I encourage you to use it. The best thing about it is that when you are not running IceBreaker all

messages go straight to an rts, so you do not lose any game speed, ie:

```
  moveq #ERR_FAILED,d0
  CALL  ErrorMessage
        ->       jmp Null
        -> Null: rts
  ...
```

As you can see your program will return within 2 instructions, so you can leave your debug messages in the code even when the program is released, and it runs at 99.99% efficiency. If something ever does go wrong, a user can activate IceBreaker and know the answer. This is very useful for beta testers.

DebugMessages and ErrorMessages

The two types of messages detected by IceBreaker are those sent by DebugMessage() and ErrorMessage(). A DebugMessage is very powerful and supports templates for all the various functions. When you see IceBreaker output something like this:

```
AllocMemBlock()   Size: 4040 (Data)
```

You know it is a DebugMessage. You can print out anything with DebugMessages, but you cannot send ErrorCodes to it.

To send out an ErrorCode (eg ERR_NOMEM) you need to use ErrorMessage(). These messages are not very versatile but are very simple and convenient to use. If you saw output like this:

```
AllocMemBlock()   Size: 4040 (Data)
Error:            NOMEM: No memory available.
```

You would know that AllocMemBlock() failed to allocate 4040 bytes of memory because there is no memory left in the system.

For detailed information on DebugMessage() and ErrorMessage(), look in the Debug autodoc.

The MESSAGE macro

A macro is specified in games.i that allows assembler programmers to easily send debug messages. Its syntax is:

```
  MESSAGE <MessageString>
```

Example:

```
  MESSAGE "This message will show up in IceBreaker."
```

You can use it anywhere in your program, it preserves all registers and requires the DPKBase in a6.

1.6 IceBreaker Copyright Notice.

COPYRIGHT