Restore

Paul Manias

COLLABORATORS

	TITLE : Restore	

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

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Restore **1**

1.1 Object: Restore 1

1.2 Object: Restore 1

1.3 Object: Restore 2

1.4 Object: Restore 2

1.5 Object: Restore 3

1.6 Restore: Activate() 3

1.7 Restore: Reset() 3

Chapter 1

Restore

1.1 Object: Restore

```
OBJECT DOCUMENTATION
Name:      RESTORE
Version:   0.9 Beta.
Date:      December 1997
Author:    Paul Manias
Copyright: DreamWorld Productions, 1996-1997.  All rights reserved.
```

1.2 Object: Restore

```
OBJECT
Name:    Restore
Module:  Blitter
Version: 1
Type:    Complex

DESCRIPTION
A  Restore  object  is  required  whenever you specify the CLEAR or RESTORE
flags in a Bob/MBob object.

Restore  objects are vital in double or triple buffered environments, as it
can be very difficult to keep track of the background areas that need to be
replaced.   By  using a Restore object and the CLEAR and RESTORE flags, you
will eliminate this problem completely.  A Restore object is list-based, so
you can use one Restore object for as many Bobs as you like.

The  Restore  object is made up of a number of entries, 1 for each Bob that
has  been  blitted to screen.  Each entry currently takes up about 28 bytes
of memory, multiplied by how many buffers there are (eg 56 bytes for double
buffering).   How many entries you require is dependent on the type of game
you  are  writing, but 50 entries will be plenty in most cases.  If you run
out  of entries, the Draw() functions will allocate extra space on the fly.
If  you  find  that this is occuring (your game may slow down when too many
objects  appear  on  screen),  we  recommend that you bump up the number of
entries you are allocating on initialisation.
```

There is a significant speed difference between restoring and clearing which you should be aware of. CLEARing is fast because the blitter is using fewer channels and the CPU can be used more effectively. RESTOREing is slower as the blitter has to move data between two different areas, plus it has to perform this action twice (once to save the background, and once to return the background). Use the CLEAR option whenever possible (if background is blank) otherwise you will probably have to use RESTORE.

ACTIONS
The Restore object supports the following actions:

```
   * Activate() Restores Bob backgrounds.
     Get()      Get a new Restore object.
     Init()     Initialises a Restore object to a Bitmap.
     Free()     Free a Restore object from the system.
   * Reset()    Reset a Restore object.
```

STRUCTURE
The Restore structure consists of the following public fields:

```
     Buffers Amount of screen buffers.
     Entries Amount of entries.
     Owner   Owner of the Restore object (must be a bitmap).
```

1.3 Object: Restore

```
FIELD
Name:        Buffers
Type:        WORD
Inheritance: Init() [Container: Screen]
Default:     1
On Change:   Cannot change after initialisation.
Status:      Read/IWrite
```

DESCRIPTION
This field specifies the amount of Bitmap buffers that the Restore object is based around. It is vital that this value matches the amount of buffers in the given screen. For example, if the screen is double buffered then this value will be 2. If triple buffered the value, will be 3.

If the value does not match you will get unpredictable results.

1.4 Object: Restore

```
FIELD
Name:      Entries
Type:      WORD
Default:   1
On Change: Cannot actively change after initialisation.
Status:    Read/IWrite
```

DESCRIPTION

This field specifes the amount of entries that are in the Restore object's list of Bob buffers.

When you first initialise your Restore object, you need to specify a generous amount of entries here -- 1 entry for every Bob that may be on screen at any time. If you under-estimate this value then everything will still work, but memory will need to be allocated on the fly to create enough space for all of your Bobs.

1.5 Object: Restore

```
FIELD
Name:        Owner
Type:        struct Bitmap *
Inheritance: Init()
On Change:   Cannot change after initialisation.
Status:      Read Only.
```

DESCRIPTION
This field references the Bitmap that your Restore object is linked to. It is initially inherited from the container object that you gave to Init(). If you provided a Screen as the container, a Bitmap will be derived from that.

1.6 Restore: Activate()

```
ACTION
Name:   Activate()
Object: Restore
Short:  Restores all buffered images by clearing or replacing backgrounds.
```

DESCRIPTION
This action will restore all the backgrounds that have been destroyed by Bobs that are using the RESTORE and CLEAR flags.

The positioning of this function is quite important - it should go BEFORE any Draw*() functions to be effective, otherwise you may get graphical glitches or even a system crash.

1.7 Restore: Reset()

```
ACTION
Name:   Reset()
Object: Restore
Short:  Resets a Restore object by clearing old image history.
```

DESCRIPTION
Resets a Restore object's image history so that any currently buffered images are no longer waiting to be restored. This is useful in circumstances such as completely changing the background imagery and no

longer blitting the previous set of images.

SEE ALSO
Restore: Flush()