# BServer

**COLLABORATORS**

| | | | |
|---|---|---|---|
| | *TITLE* : BServer | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | January 31, 2023 | |

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|---|---|---|---|
| | | | |

# Contents

# Chapter 1

# BServer

## 1.1 Bserver documentation

```
                                     BServer version 1.5
              Copyright © 1994, 1995 by Stefano Reksten
                     of 3AM - The Three Amigos !!!
```

CONTENTS OF THIS FILE:

```
               HOW TO CONTACT THE AUTHOR
```

## 1.2 Disclaimer

```
DISCLAIMER
---------------------------------------------------------------------------
   The author is *NOT* responsible for the suitability or accuracy of  this
documentation and/or the program(s)  it describes.  Any damage  directly or
indirectly caused by the use or  misuse of this  documentation  and/or  the
program(s) it describes is the sole responsibility of the user her/him self
---------------------------------------------------------------------------
```

## 1.3 This is for real!

```
                COPYRIGHT
---------------------------------------------------------------------------
BServer  v1.5, Copyright © 1994-95 by Stefano Reksten. All rights reserved.
This program may be distributed  non-commercially only  providing that  the
executable,  source  code,  documentation  and  copyright  notices  remain
unchanged and are included with the distribution.
The archive should contain the following directories/files:

 BServer_v1.5/
   Clients/
     8SVX/
       Balls.boing
       ChunLi.scream
       Clock.smp1
       Clock.smp2
       Lemming.OhNo!
       Lemming.Youpy
       Noise.snd
       Zangief.laugh
       Zangief.scream
     ILBM/
       DisplayIFF.BRS
       PT_Frames.BRS
     MISC/
       Fortune.data
     Balls
     Balls.info
     BlackScreen
     BlackScreen.info
     ChunLi
     ChunLi.info
     Clock
     Clock.info
     Curves
     Curves.info
     Dhalsim
     Dhalsim.info
     DisplayIFF
```

```
DisplayIFF.info
Eyes
Eyes.info
FadeScreen
FadeScreen.info
Fireworks
Fireworks.info
Fortune
Fortune.info
Hopalong
Hopalong.info
Invaders
Invaders.info
KillDMA
KillDMA.info
Lemming
Lemming.info
Life
Life.info
Lightning
Lightning.info
Majiks
Majiks.info
Mandelbrot
Mandelbrot.info
Maze
Maze.info
Melt
Melt.info
Multiline
Multiline.info
Noise
Noise.info
PacMan
PacMan.info
Plasma
Plasma.info
Poly
Poly.info
PT_Player
PT_Player.info
RandomDots
RandomDots.info
RastaParrot
RastaParrot.info
ScreenFall
ScreenFall.info
Shade
Shade.info
Shadows
Shadows.info
Shuffle
Shuffle.info
ShutScreen
ShutScreen.info
Spotlight
Spotlight.info
```

```
      StarField
      StarField.info
      TicTacToe
      TicTacToe.info
      World
      World.info
      Worms
      Worms.info
      Zangief
      Zangief.info
    Catalogs/
      Deutsch/
        BServer.catalog
      Français/
        BServer.catalog
      Italiano/
        BServer.catalog
      Norsk/
        BServer.catalog
      Svenska/
        BServer.catalog
    Libs/

              bitmap.library

              client.library
                  Sources/
      clients/
        ChunLi/
          ChunLi.c
          ChunLiData.c*
        Dhalsim/
          Dhalsim.c
          DhalsimData.c*
        Lemming/
          Lemming.c
          LemmingData.c*
        PacMan/
          PacMan.c
          PacManData.c*
        Shadows/
          Shadows.c
          ShadowsData.c*
        World/
          World.c
          WorldData.c*
          WorldData2.c*
        Zangief/
          Zangief.c
          ZangiefData.c*
        Balls.c
        BlackScreen.c
        Clock.c
        Curves.c
        DisplayIFF.c
        Eyes.c
        FadeScreen.c
    Fireworks.c
    Fortune.c
    Generic.c
    Hopalong.c
    Invaders.c
    KillDMA.c
    Life.c
    Lightning.c
    Majiks.c
    Mandelbrot.c
    Maze.c
    Melt.c
    Multiline.c
    Noise.c
    Plasma.c
    Poly.c
    PT_Player.c
    RandomDots.c
    RastaParrot.c
    ScreenFall.c
    Shade.c
    Shuffle.c
    ShutScreen.c
    Spotlight.c
    StarField.c
    TicTacToe.c
    Worms.c
  GFX/
    ChunLi.lha
    Dhalsim.lha
    invaders.img
    Lemming.lha
    PacMan.lha
    Shadows.lha
    World.lha
    Zangief.lha
  include/
    bitmap/
      bitmap.h
      bitmap_pragmas.h
    client.h
    client_pragmas.h
    server.h
  lib/
    libsources/
      bitmap.fd
      bitmap_chunky.s
      bitmap_library.c
      client.fd
      client_library.c
      Decrunch30.o
      PTPlay.s
    client.lib
  server/
    catalogs/
      BServer.cd
      BServer_Dansk.ct
```

```
          BServer_Deutsch.ct
          BServer_Español.ct
          BServer_Français.ct
          BServer_Italiano.ct
          BServer_Nederlands.ct
          BServer_Norsk.ct
          BServer_Portoguês.ct
          BServer_Svenska.ct
        askfiles.c
        bserver.c
        builtin_blanker.c
        commodity.c
        gadgets.c
        makefile
        modeid.c
        startclients.c
        window.c
    BServer
    BServer.info
    ClientList
    ClientList.info
    Clients.info
    Guide
    Guide.info
    Sources.info
  BServer_v1.5.info
```

(*) These files are obtained with BtoC and are not included due to their size.

Please note that the Sources directory comes in an archive.

ChunLi, Zangief and Dhalsim are Trademarks of CAPCOM,

Pacman is a Trademark of NAMCO,

Lemmings is a Trademark of DMA design,

RastaParrot is a Trademuck of Jamaikan of 3AM,

and Mandelbrot is a mathematician.

I did not know how to contact any of the first three in this list, so I could not ask for any type of permission to release the related clients. If any of them doesn't like to see his creations in my clients, I will remove the offending clients from this archive. Anyway I am not making any money from this work, so I hope they won't get upset 8-)

Of course Fred Fish is allowed to include this program in his library. I know there are other people doing like Fred: they are allowed to distribute BServer if the former conditions are respected.

This program is CARDWARE. If you use it you *MUST* send
               me
                a postcard from
your city/country.  He who uses this program not having sent me a postcard can be prosecuted by diarrhoea, data loss, expired driving licence. I aint

```
asking a lotta money from your pockets, just a stamp! :-)
---------------------------------------------------------------------------
```

## 1.4 What have we got?

```
                WHAT IS BSERVER?
---------------------------------------------------------------------------
1) HOW IT WAS BORN
Well, here's a program you can happily live without. It is a screen blanker
tha... hey! Wait!  Don't delete it immediately!  It's NOT a common blanker!
Infact, it is a SERVER for other blankers.  It was born because of two main
reasons: First, I somewhat liked a similar program for another machine: the
well-known AfterFart® for FuckinTrash® computer family (IS it a computer?),
but, luckily, I do not own a similar machine.  And second, I'm really bored
with the Blanker that comes along with Workbench.  So,  I thought to code a
blanker on my own. But I said, "Maybe other people want to build a blanker,
and maybe they're not so good at coding  or they're just (like me) too lazy
to start reading Includes and AutoDocs. Or maybe they just don't have them,
like my friend Luca Viola, who often asks me to lend them to him! >B-("
So I decided to program BServer. This program was made to help people build
their own screen blankers.

2) FEATURES (or: how does this program work)
Mainly, BServer does this:  it blanks the screen if the user does not press
any key, move the mouse, insert a disk and the computer does not pop up any
requester. (Everybody says "ooooh" ;-) BServer is a modular screen blanker;
it does not check for CPU usage, it runs its clients at a -5 priority.

3) CLIENTS
The magic is the way the screen is blanked.  Screen blanking can be done by
clients.  What are they?  A client is a program that having handshaked with
BServer is now blanking the screen and waiting BServer to tell him to stop.

Clients will be launched only when they are needed, and will quit when they
aren't needed any more.   This allows BServer to use a very small amount of
RAM. If a client fails another one will be choosen.  This can proceed up to
Builtin Blanker.
If you run a client (double-clicking on its icon or like that) when BServer
is active, this will immediately blank the screen. Note that the client may
fail if it needs some
               parameters
               , or it may refer to some default ones.

A running client may be stopped after a while and a new one started instead
(of course this amount of time can be changed).

Clients make use of their custom library,  client.library,  to reduce their
size.  Client.library keeps the code to play Protracker 3.00 modules & 8SVX
samples, clone screens,  and of course handle the communication between the
clients and BServer.

There is a  ModeID ListView gadget  in BServer's window  containing all the
screenmodes you can use. By default this is hidden, but if you press on the
'zip' gadget (that one near the window depth gadget) it will pop up.
```

```
Clients should use the selected one.  (At least MY clients try to do that!)
---------------------------------------------------------------------------
```

## 1.5 What do BServer need?

```
                WHAT DOES BSERVER NEED?
---------------------------------------------------------------------------
·  Kickstart 2.04 should be enough (if it is not, LET ME KNOW!)

·  Some clients to have fun with!
   I enclosed 40 clients;  it’s up to you to increase their number!  If you
   write some clients and you think they’re nice, stupid or crazy enough to
   be enclosed in this collection just send them to
               me
                and I will put them
   in future releases of this archive!
---------------------------------------------------------------------------
```

## 1.6 Installing BServer

```
                ---------------------------------------------------------------------------

To install this program, you have to

1) extract the archive (you have already done that ;-)

2) delete the Sources archive, if you are not interested in it (e.g. you
   are not a programmer or your programs are far better than mine ;-)

3) copy the drawer somewhere in your hard disk

4) Finally, the libraries part:
   More than a friend of mine a) does not like to put many custom libraries
   in the LIBS drawer or b) does not like to spread parts of a program in
   various drawers or partitions (i.e. they did not like to put BServer,
   or a mere copy of it, in the WBStartup, and the Catalogs in the LOCALE:
   directory). So
   · bitmap.library and client.library can be found in the Libs directory;
   · BServer itself can now be specified as a tooltype.
   You have to do the following:
   · if you want to start BServer at boot time, put a
               list
                of clients in
     the WBStartup drawer with BServer as default tool, with the DONOTWAIT
     tool specified and PRECISE PATHS to search clients;
   · install client.library and bitmap.library in LIBS:, or move them in
     the Clients drawer, as you prefer;
   · install BServer.catalog in your LOCALE: directory or leave it in the
     BServer drawer, as you prefer.
---------------------------------------------------------------------------
```

## 1.7 About languages...

```
                As I think it’s real fun and nice to have a program that speaks  ↩
                   my language
I enclosed some catalogs. If you see that your beloved language is not here
with the others, you can do the following actions:

· Get your language’s .ct file in the Sources/Catalogs directory.
· Fill in the entries.
· Send it back to
               me
                and I will compile and enclose it in the archive.

The following catalogs were compiled by:

Deutsch:   Marcel GRONER (Ironcode) gronm@info.isbiel.ch

Française: Alan GUILLEVIC (Kangourou) guillevi@andromede.u-bourgogne.fr
           25, Avenue du MAIL
           21240 TALANT

Italiano:  me! :-)

Norsk:     Haavard N. JAKOBSEN (Tittentei) haa_jako@spirea.gih.no

Svenska:   Richie OLSSON (richie) richie@medio.mh.se
           http://www.medio.mh.se/anvandare/richie/richie.html

Many thanks ppl!

PS: I had to do some little retouchs to fit them in the few space I had,
    and to add the new features every time I added a gadget...
    So if there’s any error it’s my fault! :-)
```

## 1.8 From CLI...

```
RUNNING BSERVER FROM CLI
---------------------------------------------------------------------------
If you want to launch  BServer from CLI  you should pass  him its arguments
according to its template that can of course be popped up with ’BServer ?’.
The template is: CX_PRIORITY/K/N,CX_POPUP/S,CX_HOTKEY/K,BLANKKEY/K,WLEFT/N,
WTOP/N,R=RANDOM/S,T=TIMEOUT/N/K,CT=CHANGETIME/N/K,D=DISPLAY/K,L=LIST/K,
B=BRIGHTNESS/K/N,V=VOLUME/K/N,NM=NOMOUSE/S,ND=NODISK/S

CX_HOTKEY:       The combination of keys needed to pop up BServer’s window.
                 Defaults to <lalt b>.

Here’s a brief explanation of the non-standard voices:

BLANKKEY:        A key that if pressed will tell BServer to start blanking.
```

```
                (This is to enable  the user to be able  to blank at will,
                without having to open BServer's window.)
                Default is <ctrl lalt b>.
WLEFT, WTOP:    Screen  relative  coordinates for the upper left corner of
                BServer's window. If they are not present, the window will
                be centered.
RANDOM:         Determines if BServer should choose casually from its list
                of clients.  If it's set then its builtin blanker will not
                be choosen.  (That is because it could be annoying to look
                at a black screen having other nice progs drawing birds or
                fishes or girls :-) etc...)  I decided to provide you with
                a client named  "BlackScreen" that (guess?) pops up only a
                black screen, just in case you can't live without it. ;-)
TIMEOUT=SECONDS: Determines the amount of time that  should pass before our
                server starts its activity.  BServer considers the 0 value
                as inactivity (anyway blanking actions may be started with
                the Blank gadget or with the blankkey).
CHANGETIME=SECONDS: Determines the amount of time before  the active client
                should be changed.  0 means no change.  THIS WORKS ONLY IF
                CLIENTS ARE CHOOSEN RANDOMLY
DISPLAY=ModeID: Determines the preferred display type;  it should be based
                upon the monitor and the language you use. Example: if you
                use English use "DISPLAY=NTSC:Super-High Res Laced" but if
                you use Italian, use "DISPLAY=NTSC:Super-Alta ris. inter."
LIST=filename:  The name of the list containing desired clients. This file
                will be loaded and kept in memory;  within its entries the
                server will choose a file and launch it.   Default file is
                ClientList.  *Be sure* you provide a correct path within
                clients' names if they are not in the curent directory, or
                BServer may not find them.
BRIGHTNESS=PERCENTAGE:   Specifies  the effective brightness  of the screen
                the user wants, from 0 (black) to 100 (normal).  So if you
                think  these blankers have  too shiny colors,  just reduce
                the brightness from here, not from your monitor.
                FadeScreen handles a BRIGHTNESS argument overriding this.
VOLUME=PERCENTAGE:  Specifies the volume level the user wants, from 0 (that
                is silence) to 100 (normal). So if you think some blankers
                are too noisy just reduce their volume from here, not from
                your monitor or hi-fi or whatever you use.
NOMOUSE:        Blanker won't be stopped by mouse movements (but will stop
                if a button is pressed or released).
NODISK:         Blanker won't be stopped by disk insertion or removal.

You can quit BServer by sending a CTRL_C to the program.
---------------------------------------------------------------------------
```

## 1.9 From Workbench...

```
                USING BSERVER FROM WORKBENCH
---------------------------------------------------------------------------
Just double-click on its icon.  You can specify  the CLI options by writing
them in its ToolTypes. (See
               RUNNING BSERVER FROM CLI
               .)
The popkey is by default lalt-b  (of course everything can be changed, just
modify the tooltypes of BServer).  In the program's interface you'll find a
lot of gadgets that control:
· the time that must pass before any blanking action,
· the time that must pass before a client is changed with another one,
· the desired brightness and volume level (in percentage),
· the random gadget (controls wether clients should be choosen randomly, or
  if the selected client should be used instead),
· the usual hide/quit gadget,
· the blank gadget (act immediately), has the same effect as he blankkey,
· the parameters gadget (useful to change parameters of the single clients
  without having to quit and restart),
· the add gadget (to add some clients to the list),
· the remove gadget (to remove the selected client from the list),
· the listview containing all the possible blankers the program can run,
· the listview containing the possible screenmodes.
In the menues you can find an Events voice, in which you can select/exclude
the input events that would normally stop the blankers (mouse movements and
disk insertion/removal) ...I did that mainly because my mouse is nasty! :-)
You can specify BServer as the default tool for a clients' list,  this will
prevent BServer from searching "ClientList" and force it to use your list.
Clicking on BServer,  holding shift-key and double-clicking on the icon has
the same effect (of course!).
---------------------------------------------------------------------------
```

## 1.10 ClientList file format

```
---------------------------------------------------------------------------
At launch time BServer will try to get a file called ClientList.  This file
contains a serie of rows, each containing "<NAME>: <PROGRAM> <PARAMETERS>".
These stand for the name a client should be known under,   the real name of
the program and the program's parameters. To understand this check the rows
in ClientList related to MultiLine 8-)
<NAME>s will appear in the listview gadget containing the clients names.
They must not be longer than 50 characters.   They are just symbolic names,
and may be different from the name of the client they refer to.
<PROGRAM>s will be called through SystemTagList(),  they must not be longer
than 150 chars; these are the actual clients (external programs)  that will
blank the screen.
<PARAMETERS> will be passed to programs after having opened a communication
channel with BServer. They must not be longer than 1024 chars. To help  you
passing filenames (or anything containing spaces) as parameters,  these can
be passed between a couple of  <'>  or  <">.  Parameters may be modified by
changing them in the parameters gadget, the one under the clients viewlist.

If while launching BServer you pass him another file as argument  (i.e., if
you click once on BServer's icon, hold shift-key pressed, then double-click
on another list's icon) this file will override ClientList. So you can just
create more lists and for example put one of these in the WBStartup drawer.
(In that case remember to put aldo the DONOTWAIT tooltype in the icon.)
The same result can be achieved if you specify  BServer as the default tool
of a clients list and then double-click on its icon.
---------------------------------------------------------------------------
```

## 1.11 More about clients

```
                CLIENTS AND ECS
---------------------------------------------------------------------------
Clients check which chip set is currently installed so they can adapt their
screens to the preferred screen mode.  So you can use all clients even with
ECS (but of course you can loose resolution, colors, etc... Get AGA!).
If a client cannot satisfy user's screen mode (e.g. 16 colors superhires on
an ECS machine), it will adapt the resolution to a displayable one (in this
case hires). If it is still not possible to open the screen the client will
report his failure to BServer, that will invoke another one.

Some clients MAY work with ECS,  it depends on the Display ID of the front-
most screen.   These clients actually need to clone it but adding SOME MORE
bitplanes. Thus it  may  be  possible that  they can't  open the new screen.
Within these,  Shadow and Spotlight  need just one more bitplanes, so it is
possible for them to work under ECS.
Lemming, Zangief, World and Dhalsim need at least 5 bitplanes (they *WON'T*
won't under ECS... unless the screen is a  LORES, 4  bitplanes maximum...).
World is like them but has a CUSTOMSCREEN parameter that allows him to open
on a custom LORES screen.
If Multiline is asked to work on the frontmost screen it will try to open a
screen  deeper for  color flashing;  if it is not possible  it will use the
same depth of the screen.
DisplayIFF  and  Plasma do not depend on the frontmost screen,  just on the
user's preferred screen mode.
---------------------------------------------------------------------------



CLIENTS AND PARAMETERS
---------------------------------------------------------------------------
The following clients accept some parameters:

Balls:
You can specify BALLS number - min. 1, max 8 (defaults to 5).

Clock:
You can specify FONTNAME (with trailing ".font") and FONTHEIGHT. This will
be the font used by this blanker.

DisplayIFF:
You can specify a PICTURE parameter that will contain the name if the ILBM
that will be used during screen blanking (loaded only when needed). Please
note that HAM mode is not supported by the
               bitmap.library
                scaling routine.
If use a very large picture you will notice that  it will be slow to scale
it so it will be slow to get back to Workbench :-)
The other parameter supported is MASK, boolean. That is, it have just to be
there. If specified your picture will be blitted using color 0 as transpar-
ent, *BUT* the last plane of the picture will be used for the mask,  so if
you use this, maximum number of colours will be 128 for an AGA machine.
(If you use a non-AGA machine you'll get the same 32 colors, 16 for hires.)
Defaults to the Clients/ILBM/DisplayIFF.BRS file.

FadeScreen:
```

This one accepts a BRIGHTNESS value overriding the one passed by BServer (the reason of this is easy to explain: a Brightness value of 100 passed by BServer would not blank at all... :-) and a DELAY for color fading that ranges from 0 to 100.

Fortune:
You can specify FONTNAME and FONTHEIGHT (see Clock). With TEXT you specify the text that will be printed. With FILE you specify where to retrieve the data. (Defaults to MISC/Fortune.data.) Data must be plain ascii text with cookies separed by '¤' (ASCII 164). You can specify up to 10 text lines using a '#' as a line separator. The program will check if every line fits in the screen, otherwise it will split them. Eventually in this process you could loose the lower lines.
If you specify a SCROLL parameter you'll get an unique line scrolling thru' the screen instead.

Mandelbrot:
You can specify the XRES and YRES of the rectangle (they will be cut to the screen's dimensions if required) and MAXITERS, the maximum number of iterations to go before drawing a 'black' pixel. Higher the number more precise will be the final result, but slower to achieve it. Standard value is 32.

MultiLine:
You can specify the number of GROUPS of lines (limited only by memory, but a high number will slow down the system) and the "tail"'s length (1 to 40, default 30); you can set XSIMM-etry and YSIMM-etry or XYSIMM-etry just by inserting these parameters (XYSIMM excludes the others).
Lines can be rendered in the frontmost screen instead of a custom one by specifying a FRONTMOSTSCREEN parameter.

PT_Player:
You must specify a Protracker3.0 module name. You may specify a FILE (an IFF ILBM) composed of two or more frames, each of the same size of the others, all put together to form a line. From this brush PT_Player will cut the frames to render its animation. In this case you must specify also the number of FRAMES (defaults to 6 frames found in Clients/ILBM/PT__Frames.BRS) and of VFRAMES (defaults to 1, it's the number of different lines of anim).
E.g. suppose you have a picture like this:

```
          +----------+----------+----------+----------+
          | frame1-1 | frame1-2 | frame1-3 | frame1-4 |
          +----------+----------+----------+----------+
          | frame2-1 | frame2-2 | frame2-3 | frame2-4 |
          +----------+----------+----------+----------+
          | frame3-1 | frame3-2 | frame3-3 | frame3-4 |
          +----------+----------+----------+----------+
```

In this case you have FRAMES=4 and VFRAMES=3.
VFRAMES are considered cyclical (i.e. they must be between 1 and 4, if they are less than 4 (1 vframe per voice) they will be repeated).
Remember frames are of the same size and not separed in any way.

RandomDots:
You can specify the SIZE of the small squares, from 1 to 32.

Shuffle:
You can specify a number of slices between 2 and 5; the higher this number the higher the squares' number the frontmost screen will be divided into.

```
World:
If you put  CUSTOMSCREEN  it will  open in  a custom screen instead of the
frontmost one.
---------------------------------------------------------------------------
```

## 1.12 How can I write a client?

```
                WRITING CLIENTS
---------------------------------------------------------------------------
If you decide to write a client  you should not worry too much about how to
handshake with BServer etc.,  (all the routines are included in client.lib)
but eventually you MUST tell the server you could not perform the required
blanking actions  (e.g. could not open a screen for low memory conditions).
Check if the machine you're running on has got AGA or not, and adapt to it.
Respect the user's preferred screenmode.  There are some standard sequences
of commands  in my sources to do that,  you can use them (check  expecially
Sources/Clients/Generic.c).

I have included with the distribution  the files needed to write a client:
client.h (to be included within a client source)  and client.lib  (to link
with the object). You can also look at a general client structure in one of
my clients, like StarField.c, BlackScreen.c, etc.
Read also the
               included autodocs
                to use client.library properly.

Many of my clients use
               bitmap.library
                to reduce their size.

The following clients have different authors:
Mandelbrot, Plasma, Hopalong, Life: Luca Viola and me.
RastaParrot: the concept is Copyright © by Massimo Capanni, coding by me.
KillDMA: From an idea of Gianluca Marcoccia... It's USELESS! :-) Maybe I
         should remove it from this archive...
The remaining: all by me.
---------------------------------------------------------------------------
```

## 1.13 bitmap.library

```
---------------------------------------------------------------------------
bitmap.library (currently version 1.4) is Copyright © 1994-1995 by me,
that is Stefano Reksten, and it's FREEWARE. You are free to use it in your
programs, but I retain the rights on it.

client.lib is useless if not used for BServer clients! :-)
Well, maybe some routines can be freely copied in other programs...
---------------------------------------------------------------------------
```

## 1.14 Once upon a day... what will the future bring

```
HISTORY:
---------------------------------------------------------------------------
May  5, 1994 : V1.0 - First release (alpha).

May 27, 1994 : V1.1 - ClientList and DisplayID were added (thanks must go
                      to Luca Viola for the idea), linking libraries were
                      debugged and transformed to shared to reduce clients'
                      size (client.library and bitmap.library).

Nov 25, 1994 : V1.2 - Some bugs removed (thanks to Enforcer, SegTracker and
                      FindHit!). Clients will be launched when needed, not
                      at start time. -> All clients rewritten (shorter!),
                      some BServer routines re-made from scratch.
Nov 27, 1994:         "Line" and "KillDMA" added.
Nov 29, 1994:         "Clock" added.
Jan  8, 1995:         Removed a bug in bitmap.library; Mandelbrot,Fireworks
                      and Balls reworked.
Jan 18, 1995:         CTRL-C added.

                      (V1.1 and V1.2 were internal releases - just ß-tests)

Jan 20, 1995: V1.3 -  Localized and made SMALLER than ever before!
                      "ScreenFall" and "Mandel881" added.
Jan 21, 1995:         Locale + Font adaptive.
Jan 23, 1995:         "Shade" added.
Jan 24, 1995:         Brightness added, "active" gadget removed (useless!).
Jan 29, 1995:         Some clients were made aware of AGA/non-AGA machines.
Jan 30, 1995:         A bug concerning TimeOut gadget removed.
Jan 31, 1995:         Menues added.  Now InputEvents can be "filtered" out.
Feb  6, 1995:         Bug preventing BServer to work under 2.0 fixed.
Mar  6, 1995:         Empty lines in client's list won't appear any more.
Mar  8, 1995:         Added "Hopalong"; improved "Life".
Mar  9, 1995:         Added "Plasma". (Thanks to Luca Viola for the plasma
                      source, but as ALWAYS >I< had to interface it! B-)
Mar 17, 1995:         Removed a bug preventing another client from starting
                      after first one failed.

Mar 20, 1995: V1.4 -  BServer can be a default tool; archive reworked.
Mar 22, 1995:         A bug in askfiles.c removed.
Apr 11, 1995:         A bug in the listview removed; "Text", "Multiline"
                      and "Eyes" added.
Apr 11, 1995:         "SpecuLine" and "ScrLine" added. Well I really LOVE
                      lines ;-)
Apr 26, 1995:         "Shuffle", "Noise" and "Majiks" added.
Apr 27, 1995:         8SVX support added. "TicTacToe" and "Worms" added.
Apr 28, 1995:         "Lightning" added. Size of many clients reduced.
Apr 29, 1995:         Added PlayAsynch8SVX() to client.library.
                      Autodocs for client.library moved in this guide.
Apr 30, 1995:         A bug in random clients' selection removed.
                      "Curves" added.
May  7, 1995:         "Scroller" added.
May  8, 1995:         "Spotlight" added. Shutscreen reworked.
Jun  6, 1995:         "World" added.
Jun  9, 1995:         "Lemming" added.
Jun 15, 1995:          "Hopalong881" added (well, just recompiled ;)
                       Enough stack given to Mandel881, to make it work! :-(

Jul  2, 1995: V1.5 -  QuadLine, SpecuLine, Line and ScrLine removed...
                       Don't worry, they are emulated via MultiLine :-)
                       Mandel881 and Hopalong881 removed, Mandelbrot and
                       Hopalong will check for 68881/68882/68040.
                     ->Parameters passing via ClientList instead of tooltype
                       reading in clients, parameters gadget added.
Jul 25, 1995:          Change time gadget added.
Sep  6, 1995:          All the icons support the NewIcon package by Nicola
                       Salmoria. Blank key handling improved.
Sep 23, 1995:          Protracker 3.0 module support and "PT_Player" added.
                       Added hotkey to blank immediately.
Oct 21, 1995:          "RandomDots" and "Invaders" added. "Majiks" enhanced.
Oct 23, 1995:          "Fortune" added (Text and Scroller absorbed in it).
Nov  5, 1995:          "Melt" and "Maze" added.
--------------------------------------------------------------------------

TO DO:
--------------------------------------------------------------------------
· Some other nice clients: Aquarium, Flying Things (Toilets, Toasters,
  RastaParrot, ZioPanubio®...), 3D starfield, Swarm, Rain, Tetris,
  Bubble Bobble, FracTunnel, 3D lines (configurable), Asteroids...

· Realize your ideas! (If it's not too difficult ;-)
--------------------------------------------------------------------------
```

## 1.15 Bugs :-(

```
                KNOWN BUGS:
--------------------------------------------------------------------------
PT_Player does not check if it is passed a proper Protracker3.0 module
(just checks for 'M.K.'). This may trash your memory.

I fixed all the bugs I've found or that have been reported.

I don't exclude there can be some other bugs left.  *BUT* you have also the
sources. So please, before writing me... if you can, check out what was it!
This is only a CARDWARE program...  And as I have NEVER received a postcard
from anywhere yet for this program... I won't correct any bug! >:-þ
OK, if you just can't find what the error is,  or have some brilliant ideas
or want to flame me, report something else, etc, etc... write
               me
               !
--------------------------------------------------------------------------
```

## 1.16 trouble

```
· T:After having extracted the archive and dragged it anywhere, BServer runs
  but the clients don't.
  S:Remember to copy the libraries in LIBS: (or in the Clients') directory.
```

-
  ```
  T:If I launch some clients from CLI, they don't blank!
  S:If the clients need any parameter, no way! You have to pass their parms
    through BServer's GUI.
  ```

-
  ```
  T:I pass a file as a parameter but the blanker won't accept it!
  S:If the file's name contains some spaces, you can pass it between <'> or
    <">. Example: "Data:Modules:mod.my preferred mod" or 'Data:mod.a b c d'
  ```

-
  ```
  T:There are not many clients!
  S:Why don't you make some? :-)
  ```

-
  ```
  T:BServer doesn't run under my PC clone with windows etc.
  S:Get a life.
  ```

## 1.17 That's me!!!

```
HOW TO CONTACT THE AUTHOR:
----------------------------------------------------------------------------
You can E-mail your messages (or clients!) at this address:

    rekststef@unisi.it

or snail mail to:

    Stefano Reksten c/o Naimi,
    viale Cavour, 40
    53100 Siena
    Italy

Have fun!!!
----------------------------------------------------------------------------
```

## 1.18 Autodocs

```
                                        Autodocs for client.library
                                    or
                           how to write a client


               OVERVIEW
                Here's the complete documentation of the function you can use to
interact with your server. (Internal only functions are not documented.)


               OpenCommunication

               CloseCommunication

               SendClientMsg
```

`GetServerCommand`

`WaitServerCommand`

`DISPLAYID`

`FILTEROUT`

`GET...RECT`

`GETBRIGHTNESS`

`GETVOLUME`

`RECTANGLEWIDTH`

`RECTANGLEHEIGHT`

`SpritesOff`

`SpritesOn`

`CheckAA`

`CloneFrontmostScreen`

`GetDeeperFrontmostScreen`

`DarkestColorIndex`

`BrightestColorIndex`

`Open8SVX`

`Close8SVX`

`Play8SVX`

`PlayAsynch8SVX`

`OpenModule`

`InitModule`

`PlayModule`

`StopModule`

`FreeModule`

`Decrunch30`

`GetArgString`

`GetArgInt`

`ObtainAnyFont`

## 1.19 What should a client do?

```
What’s to do:

When blanking is needed, BServer will choose a client and launch it.
So when you are launched, it’s supposed you will blank the screen, or
at least do something that will change the screens colors in order to
save your monitor’s phosphors.
OpenCommunication() will return you a DisplayIDInformation structure.
In this structure you’ll find the user’s preferred display id, overscan
dimensions, a parameters line, etc. (Its name lasts just for historical
reasons, seeing all the things this structure sontains ;)
OpenCommunication() *MUST* be called *ONCE* in order to establish a
channel between your program and BServer, that will tell him when to
stop. If anything goes wrong with memory allocation or this kind of stuff
SendClientMsg( ACTION_FAILED ) will tell BServer to run another client
from its list (or to use its builtin blanker). Check when to stop using
GetServerCommand() or WaitServerCommand(): when any input activity takes
place, BServer will issue you a COMMAND_QUIT. So long, you aren’t needed
anymore. Do your cleanup, CloseCommunication(<your DisplayIDInformation>)
and exit.
Have fun! :-)
```

## 1.20 opencommunication

```
                NAME
  OpenCommunication

SYNOPSIS
                          dinfo = OpenCommunication()
  struct DisplayIDInformation * = OpenCommunication()

FUNCTION
  Opens a connection between your blanker (client) and the server.
  Opens client.library . ClientBase is kept in client.lib, so you do
  not need to open it any more. client.library will be closed on exit.

RESULTS
  A DisplayIDInformation if communication was opened, NULL otherwise.

SEE ALSO

               CloseCommunication()
```

## 1.21 closecommunication

```
              NAME
  CloseCommunication

SYNOPSIS
       CloseCommunication( dinfo )
  void CloseCommunication( struct DisplayIDInformation * )

FUNCTION
  Closes communication between your client and the server. Frees the
  structure returned by OpenCommunication. Closes client.library.

SEE ALSO

             OpenCommunication()
```

## 1.22 sendclientmsg

```
              NAME
  SendClientMsg

SYNOPSIS
  success = SendClientMsg( action )
     BOOL = SendClientMsg( ULONG )

FUNCTION
  Sends a message to the server (waiting reply). Of course, your
  client should already have opened a communication with the server.

INPUTS
  action: an UBYTE describing the action you're performing. This may be:

        · ACTION_FAILED  (You couldn't start your blanking actions,
                         so you are telling the server to ask
                         another blanker.)

RETURNS
  TRUE if message was received, FALSE otherwise.

SEE ALSO

             OpenCommunication()
```

## 1.23 getservercommand

```
NAME
  GetServerCommand, WaitServerCommand

SYNOPSIS
  command = GetServerCommand()
  command = WaitServerCommand()

    UBYTE = GetServerCommand()
    UBYTE = WaitServerCommand()

FUNCTION
  It returns the command sent by the server. The difference between
  the two functions is that GetServerCommand checks the communication
  port searching for any message coming from the server, while
  WaitServerCommand will make your program wait until a command
  (different from COMMAND_IDLE) is received.
  The server may send you these commands:
  · COMMAND_IDLE:    the CommunicationPort is empty. This one can be
                     obtained only with GetServerCommand.
  · COMMAND_QUIT:    the server is asking you to quit.
  Note that GetServerCommand POLLS the port so don't use it out of any
  rendering (or something else) cycle. Use WaitServerCommand instead.

 There is a macro, STILL_BLANKING, that checks if the server is stopping
 us; an example of its usage may be:

 while( STILL_BLANKING )
    {
    RenderSomething();
    }

SEE ALSO
  any client's source
```

## 1.24 displayid

```
NAME
  DISPLAYID

SYNOPSIS
  DisplayID = DISPLAYID( dinfo )
      ULONG = DISPLAYID( struct DisplayIDInformation * )

MACRO
  This macro returns you an ULONG containing the screen mode choosen
  by the user for a client.
```

## 1.25 filterout

```
                NAME
  FILTEROUT

SYNOPSIS
  newDisplayID = FILTEROUT( oldDisplayID, flags_to_remove )
         ULONG = FILTEROUT( ULONG,        ULONG )

MACRO
```

```
  This macro removes all the unwanted flags from a DisplayID, e.g.:
  you are told to blank; the server sent you also the preferred
  screenmode (stored in dinfo). You want to filter SUPERHIRES mode
  and LACE flag but want a hires screen. You could pass as value
  for SA_DisplayID this value:
  FILTEROUT( DISPLAYID(dinfo) | HIRES, SUPERHIRES|LACE )

 SEE ALSO

               GetServerCommand(), WaitServerCommand()
```

## 1.26 getrect

```
 NAME
   GETSTANDARDRECT
   GETMAXOSCANRECT
   GETVIDEOOSCANRECT
   GETTXTOSCANRECT
   GETSTDOSCANRECT

 SYNOPSIS
           rectangle = GET...RECT( dinfo )

   struct Rectangle * = GET...RECT( struct DisplayIDInformation * )

 MACRO
   These macro return the address of a Rectangle structure containing
   a certain overscan value. (As ever, the DisplayIDInformation
   structure was allocated by your server.)
```

## 1.27 getbrivol

```
 NAME
  GETBRIGHTNESS
  GETVOLUME

 SYNOPSIS
  brightnes_level = GETBRIGHTNESS( dinfo )
     volume_level = GETVOLUME( dinfo )

           UBYTE = GETBRIGHTNESS( struct DisplayIDInformation )
           UBYTE = GETVOLUME( struct DisplayIDInformation )

 MACRO
  These two macro return you the desired level of brightness and volume
  the user wants to get. Of course you don’t have to respect them but in
  general you should. A program that does not respect GETBRIGHTNESS is
  FadeScreen (but it is due to its nature :-).
  Thus you can do for example

  bri = GETBRIGHTNESS(dinfo);
  volume = GETVOLUME(dinfo);
  /* ... */
  LoadRGB4( vp, 1, 5*bri/100, 12*bri/100, 15*bri/100 );
  /* ... */
  Play8SVX( my_sound, volume );
  /* ... */
```

## 1.28 rectdims

```
NAME
 RECTANGLEWIDTH
 RECTANGLEHEIGHT

SYNOPSIS
  width = RECTANGLEWIDTH( rect )
 height = RECTANGLEHEIGHT( rect )

  UWORD = RECTANGLEWIDTH( struct Rectangle * )
  UWORD = RECTANGLEHEIGHT( struct Rectangle * )

MACRO
 This macro returns the current width/height for a screen from a
 Rectangle structure.

 Example: OpenScreenTags( NULL,
             SA_Width, RECTANGLEWIDTH( GETTXTOSCANRECT(dinfo) ), ...
```

## 1.29 spritesoff

```
               NAME
 SpritesOff()

SYNOPSIS
 SpritesOff()

FUNCTION
 Disables sprite DMA, removes sprite from screen.
 Remember that performing a ScreenToFront action (when doublebuffering)
 or similar will activate sprite DMA again.

SEE ALSO

              SpritesOn()
              , Balls.c, any other client's source

BUGS
 Well sprites should remain deactivated...
```

## 1.30 spriteson

```
                NAME
 SpritesOn()

SYNOPSIS
 SpritesOn()

FUNCTION
 Enables sprite DMA.

SEE ALSO

               SpritesOff()
```

## 1.31 checkaa

```
NAME
 CheckAA()

SYNOPSIS
 aa_chipset = CheckAA()

 BOOL = CheckAA( void )

FUNCTION
 Tells wether AA chipset is installed or not, and if you can use it
 (that is, if you have kickstart 3.0+ installed). If this function
 returns FALSE, you have to fall back to 2.0, or to quit. As a general
 behaviour, a client should fall back to ECS.
```

## 1.32 clonefrontmostscreen

```
NAME
 CloneFrontmostScreen()

SYNOPSIS
          screen = CloneFrontmostScreen( brightness_level )
 struct Screen * = CloneFrontmostScreen( UBYTE )

FUNCTION
 Returns you a pointer to a Screen structure; this is a Screen that is
 identical to the frontmost screen, although it is automatically put
 to a Y=0 coordinate. It is up to you to free this screen when you're
 finished. Colors will be to a brightness_level percent of their original
 brightness. Valid range is 0-100, although no check is made on that.
```

## 1.33 getdeeperfrontmostscreen

```
               NAME
 GetDeeperFrontmostScreen()

SYNOPSIS
             scr = GetDeeperFrontmostsScreen( brightness_level, image_depth )
 struct Screen * = GetDeeperFrontmostScreen( UBYTE,            UBYTE )

FUNCTION
 Returns you a pointer to a Screen structure; this Screen is different
 from the original frontmost screen in terms of bitplane number (depth).
 The image_depth referes to an image that the user wants to be printed
 on the screen; thus (according to the fact that the screen and the image
 may - and surely will - have different colors in term of RGB components
 and number) the screen is deeper, it has MAX(image_depth,scr_depth) + 1
 bitplanes, the last of those will be filled with 1's. The lower and the
 upper halfs of the colormap will be loaded respectively from color 0 and
 from color (2<<scr_depth), with a color number that reflects that of the
 previous frontmost screen. It's up to the user to load properly the first
 colors with those of the image (from color 1 to color 2<<image_depth-1).
 This will produce a screen that looks like the previous but really has
 the colors "shifted" to the upper half of the table. The image can be
 now drawn using a cookie-cut function like BltMaskBitMapRastPort().

 brightness_level is used to reduce the brightness of the original screen
 that has been "cloned". Valid range is from 0 to 100, but no check is
 made.

SEE ALSO

              CloneFrontmostScreen()
              , any source of a
 client featuring animation.
```

## 1.34 darkestcolorindex

```
               NAME
 DarkestColorIndex()

SYNOPSIS
 index = DarkestColorIndex( screen )
 UBYTE = DarkestColorIndex( struct Screen * )

FUNCTION
 Returns you the index (pen number) of the darkest color in this screen's
 colormap. As a technical information, his function is based on the lowest
 sum of the squares of the single R, G, B components of the colors.

SEE ALSO

              BrightestColorIndex()
```

## 1.35 brightestcolorindex

```
                NAME
 BrightestColorIndex()

SYNOPSIS
 index = BrightestColorIndex( screen )
 UBYTE = BrightestColorIndex( struct Screen * )

FUNCTION
 Returns you the index (pen number) of the brightest color in this screen's
 colormap. As a technical information, his function is based on the highest
 sum of the squares of the single R, G, B components of the colors.

SEE ALSO

               DarkestColorIndex()
```

## 1.36 open8svx

```
                NAME
 Open8SVX()

SYNOPSIS
     svx = Open8SVX( name )
 Sound * = Open8SVX( char * )

FUNCTION
 Loads an IFF-8SVX sample from disk and returns you a pointer to a Sound.
 You actually don't need to know how this works (well if you're curious
 you can check it in /include/server.h), you just need the pointer.
 The 8SVX sample must be 1 octave wide and one shot only.

SEE ALSO

               Close8SVX()
               ,
               Play8SVX()
               ,
               PlayAsynch8SVX()
               ,
               GETVOLUME()
```

## 1.37 close8svx

```
                NAME
 Close8SVX()

SYNOPSIS
      Close8SVX( svx )
```

```
  void Close8SVX( Sound * )

 FUNCTION
  Frees all memory allocated for a Sound.

 SEE ALSO

               Open8SVX()
               ,
               Play8SVX()
               ,
               PlayAsynch8SVX()
               ,
               GETVOLUME()
```

## 1.38 play8svx

```
                NAME
  Play8SVX()

 SYNOPSIS
      Play8SVX( svx,     volume_percentage )
  void Play8SVX( Sound *, UBYTE )

 FUNCTION
  Plays a Sound (a loaded 8SVX sample). It is safe to pass a NULL pointer
  to this function. This function will fail if someone has allocated all
  audio channels. This function will return only when the sample has been
  completely played. If the sound can't be played (because audio channels
  have been allocated by someone else) this function will wait as if the
  sound could be played.

 SEE ALSO

               Open8SVX()
               ,
               Close8SVX()
               ,
               PlayAsynch8SVX()
               ,
               GETVOLUME()
```

## 1.39 playasynch8svx

```
                NAME
  PlayAsynch8SVX()

 SYNOPSIS
      PlayAsynch8SVX( svx,     volume_percentage )
  void PlayAsynch8SVX( Sound *, UBYTE )
```

```
FUNCTION
 Plays a Sound (a loaded 8SVX sample). It is safe to pass a NULL pointer
 to this function. This function will fail if someone has allocated all
 audio channels. This function returns immediately.

SEE ALSO

               Open8SVX()
               ,
               Close8SVX()
               ,
               Play8SVX()
               ,
               GETVOLUME()
```

## 1.40 openmodule

```
                NAME
 OpenModule()

SYNOPSIS
   module = OpenModule( filename )
 Module * = OpenModule( char * )

FUNCTION
 Loads from disk a music in protracker ('M.K.') module format.
 Also calls
               InitModule()
               .

BUGS
 None known.

SEE ALSO

               InitModule()
               ,
               PlayModule()
               ,
               StopModule()
               ,
               FreeModule()
```

## 1.41 initmodule

```
                NAME
 InitModule()

SYNOPSIS
```

```
      InitModule( module )
 void InitModule( Module * )

FUNCTION
 Prepares a module to be played. This is automatically called by

               OpenModule()
               , and it is meant to be called again to
 reset the module to its beginning.
 This function is safe to call with a (Module *) NULL parameter.

BUGS
 None known.

SEE ALSO

               OpenModule()
               ,
               PlayModule()
               ,
               StopModule()
               ,
               FreeModule()
```

## 1.42 playmodule

```
                NAME
 PlayModule()

SYNOPSIS
      PlayModule( module )
 void PlayModule( Module * )

FUNCTION
 Tries to start a module in background.
 This function is safe to call with a (Module *) NULL parameter.
 Uses Protracker 3.0 playroutine.

BUGS
 None known.

SEE ALSO

               OpenModule()
               ,
               InitModule()
               ,
               StopModule()
               ,
               FreeModule()
```

## 1.43 stopmodule

```
                NAME
  StopModule()

 SYNOPSIS
       StopModule( module )
  void StopModule( Module * )

 FUNCTION
  Stops a module that is being played. This function is called also by

               FreeModule()
               .
  This function is safe to call with a (Module *) NULL parameter.

 BUGS
  None known.

 SEE ALSO

               OpenModule()
               ,
               InitModule()
               ,
               PlayModule()
               ,
               FreeModule()
```

## 1.44 freemodule

```
                NAME
  FreeModule()

 SYNOPSIS
       FreeModule( module )
  void FreeModule( Module * )

 FUNCTION
  Calls
               StopModule()
               , then releases memory associated to a module.
  This function is safe to call with a (Module *) NULL parameter.

 BUGS
  None known.

 SEE ALSO

               OpenModule()
               ,
               InitModule()
               ,
               PlayModule()
               ,
               StopModule()
```

## 1.45 decrunch30

```
 NAME
  Decrunch30

 SYNOPSIS
  result = Decrunch30( dest_ptr, compressed_data_ptr )
    BOOL = Decrunch30( UWORD *,  UBYTE * )

 FUNCTION
  Decrunches data compressed with BtoC 2.5's BYTERUN2 or upper (a program
  I made to cut frames... ;-)
```

## 1.46 getargstring

```
                NAME
  GetArgString

 SYNOPSIS
  position = GetArgString( parm_line, string, store_buffer )
    char * = GetArgString( char *,    char *, char * )

 FUNCTION
  Checks if 'string' is contained in 'parm_line'. If it is found this
  function copies the next parameter in store_buffer (if it is not NULL)
  and returns a pointer to 'string' in the original 'parm_line'.

  The reason why the result is copied in a buffer and returned in a pointer
  is to handle spaces. A message can begin with ' or " (thus it MUST finish
  with ' or " ), if this happens everything between the '/"s will be copied
  (also spaces). If a message is not limited by spaces it will end as soon
  as a space is met.

  Tipically parm_line is taken from the di_Args field of the
  DisplayIDInformation structure.

 SEE ALSO

                GetArgInt
```

## 1.47 getargint

```
               NAME
 GetArgInt

SYNOPSIS
  value = GetArgInt( parm_line, string )
 char * = GetArgInt( char *,    char * )

FUNCTION
 Checks if 'string' is contained in 'parm_line'. If it is found this
 function returns the integer value of the following parameter.

 Tipically parm_line is taken from the di_Args field of the
 DisplayIDInformation structure.

SEE ALSO

              GetArgString
```

## 1.48 obtainanyfont

```
NAME
 ObtainAnyFont

SYNOPSIS
              font = ObtainAnyFont( dinfo )
 struct TextFont * = ObtainAnyFont( struct DisplayIDInformation * )

FUNCTION
 Tries to open any font specified in the DisplayIDInformation; tipically it
 contains the font specified for a certain client in the parameters gadget of
 BServer's main window. But if no font is specified it returns, in order,
 diamond/20, times/24 or topaz/8. The application has to close the font when
 finished.
```