back

512 Byte Democompo

The Results ?

Unfortunately some things went wrong this time. MD left to the TIG98 too early to fetch the competition entries in his mail. (He wasnt able to check his mail account from there). Thus the entries werent at the partyplace in time for the compo, so that no"real" voting could take place.

As far as I have heard there was a jury voted competition later. **Naos** contribution won this compo. But I didnt receive any further results.

Five entries were submitted:

Platon	Boing
RAY	Notime
Nao	512
Azure	Burn
Sniper	Jumparound

Here are the demos and the sources:

512b-demos.zip 13kb.

512bsource.zip 21kb.

The Mission

The 512 democompo is another "squeeze the last bit out of your code" compo. But unlike in most other competition with size restriction in this case not the size of the executable is counted, but the size of your code byself. A startup code providing you with functions for chunky screen output etc is given. This is done to avoid hacks which might prevent the demo from running on other configurations, than you use..

Download the archive with the startup code and this rules as text file here

Credits go to **Bluberry** for providing the init code and to **Piru** for some additional fixes.

The Rules

- You have to submit both a source and an executable of your demo(assembled with the given init)
- All sources will be published on http://come.to/amiga, after the competition is over. In case you dont want yours to be published, please state it clearly in your contribution.
- Your demos must use the given init. It contains everything you need. Take a look at the section below for further information on it. Your demo is not allowed to access any other resources (library-functions, rom data etc).
- Direct hardware access is only allowed for sound-playing, nowhere else. (in case you really want to squeeze sound in your demo :))
- Selfmodifying code (SMC) is only allowed in conjunction with a cache-clear, which is provided as a function of the init. No 030-specific hacks.
- No FPU-Usage.
- Your demo must not have an own exit routine. The init contains a build-in one within the refresh-function.
- The three "RTS" are included in the 512 bytes ! (so you may only add 506 bytes, unless you trick around with the labels)
- The startup-functions may be called with "BSR.W"

The Startup-Code

Place your code between the labels Code_Start and Code_End and your BSS definitions at the bottom of the source. You have to supply 3 functions:

Init:

Called once in the beginning. Registers are undefined.

VBlank:

Called every vblank after Init has finished. On first call, registersare as left by Init. On subsequent calls, registers are as left by last call.

Main:

Called once when Init has finished. Registers are as left by Init. If it terminates, the demo will exit.

Your code may only reference memory inside the code or in the BSS section and may only call the 6 supplied functions:

Update128x128:

Update the screen in 2x2 resolution using the supplied 128x128 linear chunky buffer. The sides will get color 0. Callable from Init and Main.

Input: A0 = Chunky buffer

Update160x128:

Update the screen in 2x2 resolutionusing the supplied 160x128 linear chunky buffer.Callable from Init and Main.

Input: A0 = Chunky buffer

Update256x256:

Update the screen in 1x1 resolutionusing the supplied 256x256 linear chunky buffer. The sides will get color 0. Callable from Init and Main.

Input: A0 = Chunky buffer

Update320x256:

Update the screen in 1x1 resolution using the supplied 320x256 linear chunky buffer. Callable from Init and Main.

Input: A0 = Chunky buffer

CacheClear:

Clear caches so written SMC will be valid. Callable from Init, VBlank and Main.

SetPalette:

Set the screen palette to the one supplied. Format is 256 longwords in $xxrrggbb. The upper byte of each longword is ignored. Callable from Init, VBlank and Main.

Input: A0 = Palette
