Subject: : Amiga Classic
Topic: : ALS, a new graphics system - RELEASED!
ALS, a new graphics system - RELEASED!
Author: : saimo
Date: : 2020/9/8 16:53:33
URL:

Recently I've been working on a new graphics engine.

VIDEO PREVIEW #1
VIDEO PREVIEW #2
VIDEO PREVIEW #3
VIDEO PREVIEW #4
VIDEO PREVIEW #5
FINAL VIDEO

OVERVIEW

"ALS" stands for "AMOS Layers System", as it turns the screens of AMOS Professional into layers that can be laid over one another, with complete control of order, opaqueness and colors, while keeping them renderable as usual.
It is easy, does not require much knowledge of the Amiga graphics hardware, does not need installation, does not depend on third-party extensions and comes as a collection of variables, arrays and procedures written in fully-commented AMOS code - it can be thought of as an AMOS source-level library.

ALS comes in two versions:
* v1: for OCS/ECS/AGA - bigger and slower (and no longer developed)
* v2: for AGA only - smaller and faster

https://www.retream.com/ALS

GENERAL FEATURES

· Layers usable as screens and vice versa
· Overlaying of multiple layers
· Overlaying order freely arrangeable
· Per-layer planes height
· Per-layer planes number
· Per-layer double-buffering
· Per-layer vertical positioning
· Per-layer colors
· Per-layer 257-degree opaqueness

· Per-color 257-degree opaqueness
· 24-bit internal colors
· LORES horizontal positioning of layers
· LORES and HIRES display resolutions
· Programmable display window size
· Automatic centering of display window
· Automatic adjustment to chipset (OCS/ECS/AGA)
· Automatic creation of layers from ILBM files
· Display descriptors
· Layer descriptors and snapshots
· Global snapshots
· Palettes management
· Banks management
· Basic file management

ECS/AGA FEATURES

· Selectable video standard (NTSC/PAL) <ECS Agnus / AGA>
· Display border blanking <ECS Denise / AGA>

AGA FEATURES

· Non-EHB 6-plane displays
· 24-bit display colors
· 24-bit palette colors
· SHRES display resolution
· HIRES and SHRES horizontal positioning of layers
· 4x planes fetch mode

RESTRICTIONS DUE TO HARDWARE

· Maximum number of visible planes / 1-plane layers: OCS/ECS, HIRES: 4; OCS/ECS, LORES: 6; AGA: 8
· On OCS/ECS, EHB mandatory for 6-plane displays
· On OCS/ECS, 12-bit display colors
· On OCS/ECS, 12-bit palette colors
· On OCS Agnus, video standard (NTSC/PAL) dictated by the hardware
· Limited horizontal positioning of display window
· Same width for all layers
· Same horizontal positioning for all layers

RESTRICTIONS DUE TO AMOS

· Maximum number of in-use/ready-to-use layers: 8
· Maximum number of planes per layer: 6

RESTRICTIONS DUE TO DESIGN

· Most AMOS display/screen commands not allowed/possible
· Floppy drives not usable when the display is on.

HOW ALS WAS BORN

In 2003 I wrote a Copper-based screen flipping effect for a developer who was making a game with AMOS. Eventually, the effect was not used (and the game was not made at all), but writing it gave birth to a whole bunch of ideas, which little by little transformed into a collection of procedures that constituted a small graphics system called XPF (Cross PlayField).

The development however, having started from an effect and having proceeded spontaneously, lacked the necessary rigour that a proper system requires, so I decided to rewrite everything from scratch and created CSS (Custom Screens System). That one turned out to be a clean, feature-rich system. It worked nicely and I even wrote a few tutorials for it.

But it did not support sprites. While pondering on how to add them, I realized that actually the core design was not good enough and that an alternative one would have allowed sprites and have been more efficient, too. Therefore, I wrote another system: AVS (Advanced Video System). When I was at about 80-90% of the development... I lost the sources. I cannot remember how that happened, but for sure I could not recover them, so I remained only with the sources dating back to some days before, when a lot of important code had not been written yet. The anger, which made me hate the idea of reimplementing what had been lost, coupled with the fact that I was about to move country, killed the project.

The idea of rewriting an old game of mine using CSS - which was good enough for the purpose - kept on lingering in my head through the years. I kind of promised myself I would do that sometime, as a smaller project between bigger ones - provided I could swallow the idea of using a suboptimal system, that is. In fact, in a few occasions, I considered completing AVS first... only to drop the idea immediately: I just could not bother getting acquainted with that old code, maybe discovering that, after all, I would do things differently once again.

Although, in the meanwhile, I dedicated myself to many other projects, the ghosts of those systems kept on haunting me. In 2019 I presented CSS to the world with a video preview: it was an attempt at doing justice to the system (and thus hopefully making peace with it) and at forcing myself to complete the work by exposing publicly the waste it represented. Since then, I made a new game (Blastaway), I continued the work on and released a preview of a game (QUOD INIT EXIT IIo), I released two little updates for another game (MeMO), I almost finished an update for yet another game (SkillGrid), I created two other graphics systems not related to ALS and I made a demo (THE CURE) with one of them. But the ghosts were still there.

Well - as they say - enough is enough and better later than never: the time to get rid of them came and in July 2020 I designed and implemented a new and proper system from the scratch - and so ALS was born.

PERFORMANCE

Whenever a color, an alpha value, or the stack of the layers gets changed, it is necessary to recalculate many or even all the colors starting from those relative to the first bitplane of the first layer affected. That is a CPU-intensive operation that AMOS struggles with, so it must be avoided as much as possible by not using dynamic changes or by precalculating palettes (ALS provides specific procedures for that).

Other than that, the system is quite lightweight.

WHY AMOS SOURCE CODE?

ALS could have been written in assembly and made available as high-performance source code, as a linkable object or as an AmigaOS library, but it has been written as pure AMOS code for AMOS programs, instead. Even within the AMOS world, it could have been written as an extension or at least as embedded machine language procedures, thus resulting more efficient (in particular, the ALS_CALCULATE_PALETTE*[] procedures would have been dramatically faster). Why AMOS source code, then?

First of all, because of how ALS was born; secondly, because returning to AMOS and writing everything in such language - the language I started programming on the Amiga with - after many years was fun; then, because I enjoy the idea of showing that AMOS, which way too often gets exaggeratedly bashed, is actually more capable than it is commonly thought of; finally, because I find it just too amusing to see the surprised reactions from people who can barely believe that what ALS achieves is done by means of the original AMOS alone.