The project is about an application which is a combination of emulation, dissassembler and debugger for the VCS 2600.
I have written cpu- and emu-core completely from scratch. Therefore there are still a lot of bugs and problems to fix at the moment, but things are looking promissing even at this early stage.

Restrictions in the current version:

- Support for 2 and 4k roms only (bank-switching will be supported in future versions).
- No sound support yet.
- While some roms are already playable (frogger, dodge em) others dont even show up propper video output (eg asteroids).
- Requires Windows 95,98 or NT with DirectX 3.0 or better with Video-Resoltion of 1024 by 768 or higher.

The emulation-core generates fast video-output in windowed mode by directly manipulating memory of the video-hardware. This feature requests DirectX-support (V3.0 or better).

On an 800MHz machine the emulation may run between 26 and 34 FPS (unoptimized emu-core yet, so still a lot of optimization potential)

The roms disassambled binary-data apears here as assambly-code.

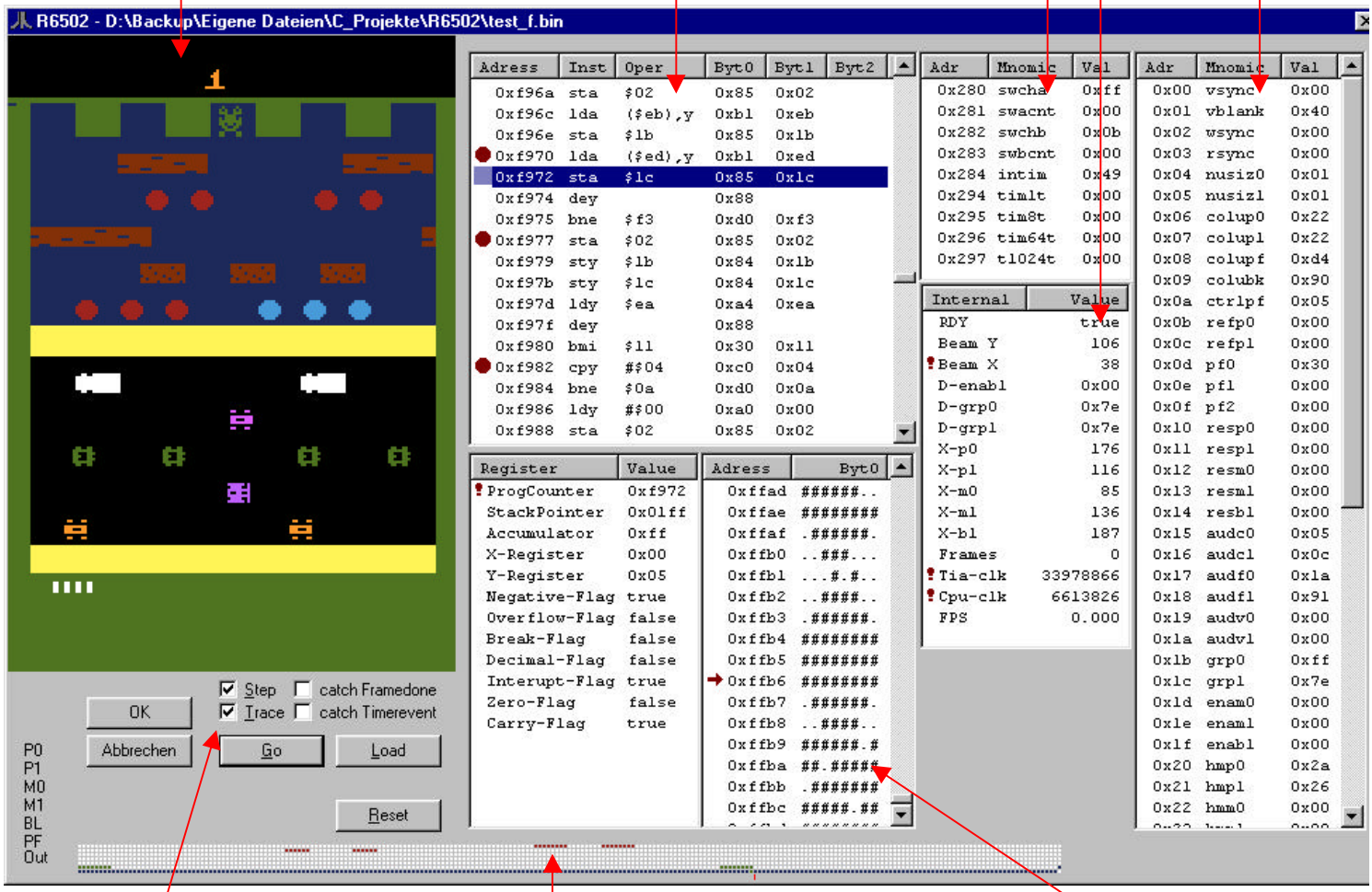
The instruction where the programm-counter points to and which will be executed next, ist highlighted.

Breakpoints can simply be toggled on and off, by mouseclick.(number of breakpoints limmited by number of instructions only).

While in trace-mode, the emu-debugger dynamicly changes the focused/highlighted line.

All importand registers, such as PIA, TIA, 6502 and some emu-internals are shown while emulation is freezed (space-bar) or in break-mode (pause/break-key).

While in single-step-mode, a red exclamationmark indicates register-changes since the last step.



Step-Flag: single-step mode, emu breaks automaticly after each emulated cpu-instruction.

Trace-Flag: animates the assembly-code list-control while emulation ist running (of course slows down emulation dramaticaly).

Catch Framedone-Flag: Breaks after completion of an individual frame

Catch Timerevent: Breaks if timer is activated and INTIM reaches 0

Visual output of the internal „rasterobjects-buffer“

For each of the visible objects (p0, p1, m0, m1, bl, pf) the emulation internaly computes an independent raster buffer, depending on such registers as COLUPx, GRPx, NUSIZx etc.

The order of the symbolic-names on the left side indicates the order of the buffer-lines.

The complete memory-space is shown in a special form of binary-mode (dot indicates 0, # indicates 1).

This is for easier recignition of graphical information.

A red arrow always indicates the last adress of cpu's memory-access.

Constructive comments are welcome under **stefan.burger@uumail.de**