MS-DOS demoscene 2021 - software rendering, sizecoding and modern approach to retro-demomaking

Artem Vasilev :: wbcbz7 @ Demodulation 2021

why? what?

while most of other oldskool platforms live through spike of retrocomputer interest, PC is no an exception
did you ever wanted to make something for oldskool PC but were afraid of?

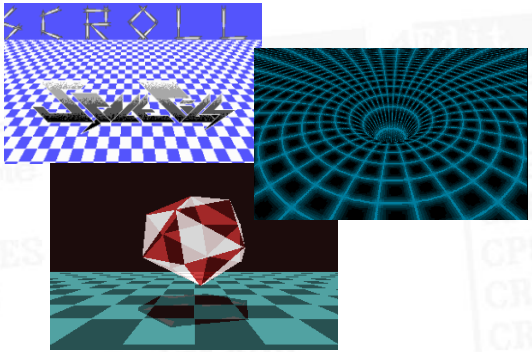
we'll be focused on these topics:

- PC/DOS scene origins and historic periods
- hardware overview – from original IBM machines to late-90 midskool stuff
- in-depth breakdown – x86, VGA and DOS/BIOS
- "ms-dos scener 101" guide – development tools and environment

why dos?

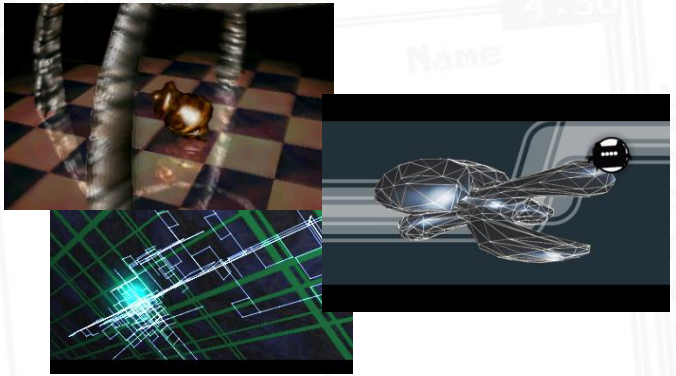
- MS-DOS is, scene-wise, quite a curious platform…
- being initially associated with “boring office typewriters”, in the 90s quickly made it’s own way on throwing competitors out of market with raw CPU power and open architecture
- at the turn of millennium, while almost every PC software making it’s way on Windows, Linux or other OSes, DOS still stayed as appealing platform for multimedia audiovisual enthusiasts
  - eventually moved onto Windows and DirectX/OpenGL, of course
- close-to-metal experience, software rendering being the most natural way
- very friendly environment for extremely tiny sized stuff
- … it’s fun :)

ms-dos demoscene :: distinctive periods



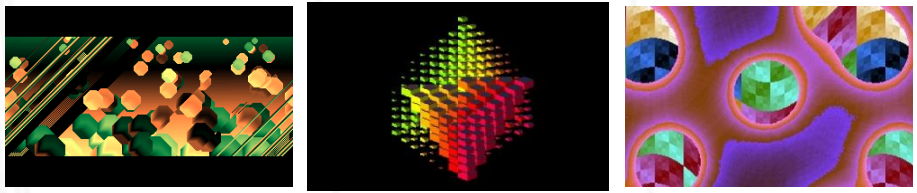
the birth
(1990 – 1992)
286-386, EGA/VGA, Covox/SB

the golden age
(1993 – 1996)
486, VGA, GUS



the transfiguration
(1997 – 1999)
Pentium, SVGA, GUS/SB16

Original IBM PC machines

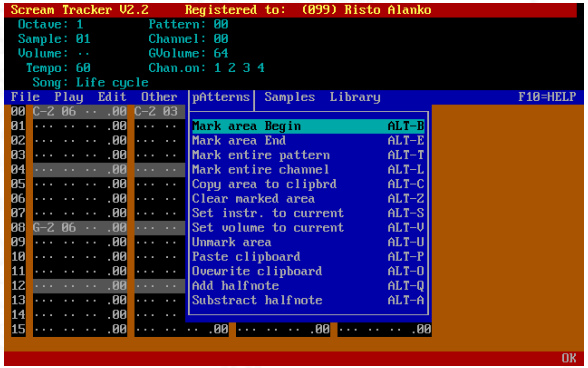


sizecoding

ms-dos demoscene :: the birth

inspired by both cracktros and Amiga demoscene, started from simple EGA/VGA experiments with ripped samples and 4 channel mods.

Slideshow by Future Crew (1990) – first ever production with 4-channel MOD music

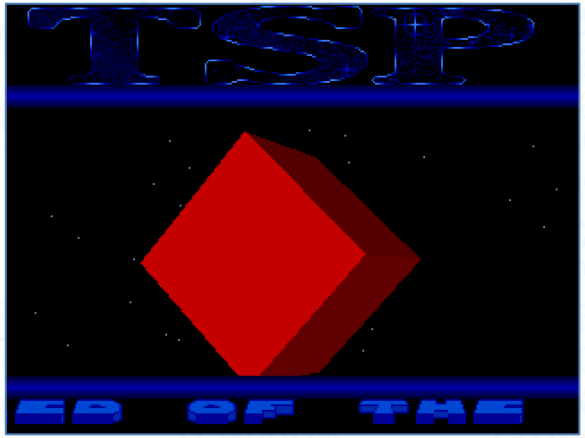


Scream Tracker 2 by Future Crew (1990-1992) – marking the start of PC trackers era

Megademo by The Space Pigs (1990) – the first big scene production, and it runs on EGA!

ms-dos demoscene :: the birth:: 1991

…slowly drifting towards vector worlds ….



Vicky by The Space Pigs
(1991)

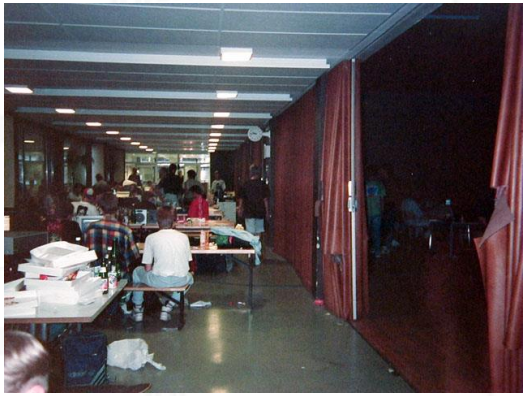
VectorDemo by Ultraforce
(1991)

Cronologia by Cascada
(1991)

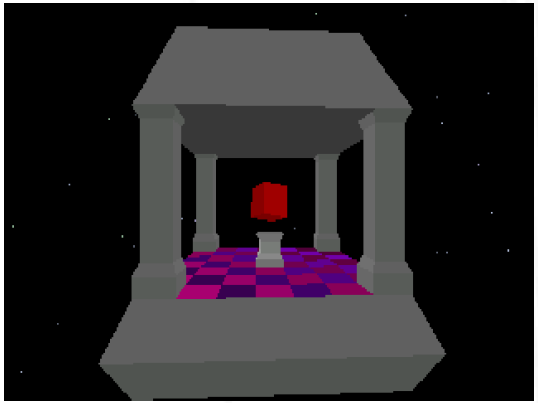
ms-dos demoscene :: the birth :: 1992

eventually reaching Amiga level of complexity with rapid performance growth

Unreal by Future Crew (1992)



Assembly 1992 – the first major demoparty with PC compos



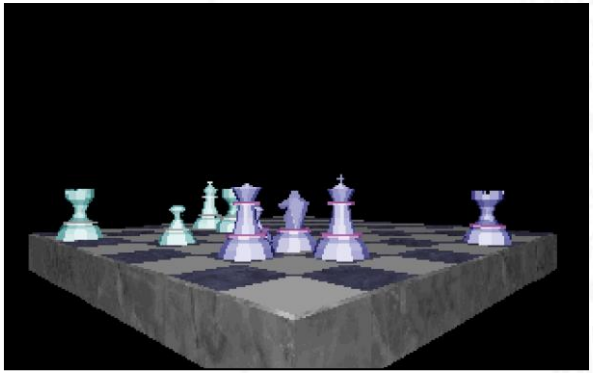
Amnesia by Renaissance (1992)

ms-dos demoscene :: the birth :: 1993

... driving beyond four channel MODs with new trackers and hardware like GUS …

Second Reality by Future Crew (1993)

Untitled by Dust (1993)

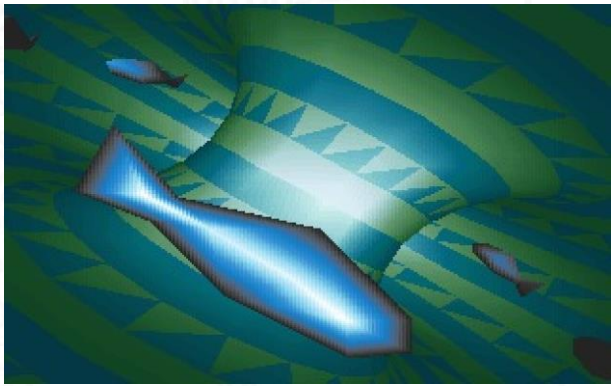


Crystal Dream II by Triton (1993)

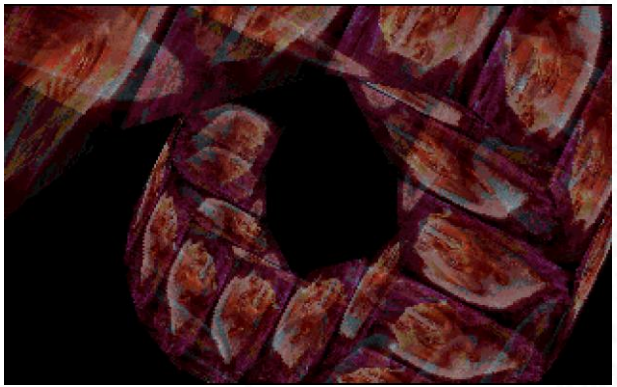
ms-dos demoscene :: the golden age:: 1994

... quickly catching up with and beyond the new Amiga AGA capabilities …

Verses by Electromotive Force (1994)



Cyboman 2 by Complex (1994)

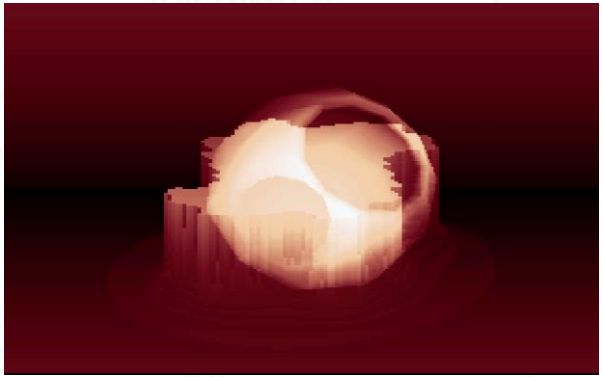


No! by NoooN (1994)

ms-dos demoscene :: the golden age:: 1995

... expanding to high resolution and phong shaded duck.3ds land …

Caero by Electromotive Force (1995)



Drift by Wild Light (1995)

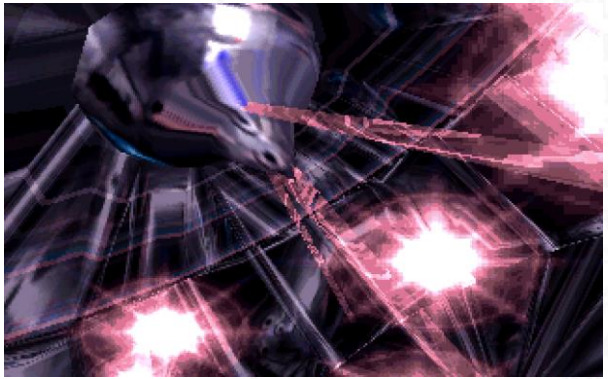
Stars: Wonders of the World by NoooN (1995)

ms-dos demoscene :: the golden age:: 1996

... sqeezing last bits from freedirs and lens flares …

Toasted by Cubic Team & $een (1996)

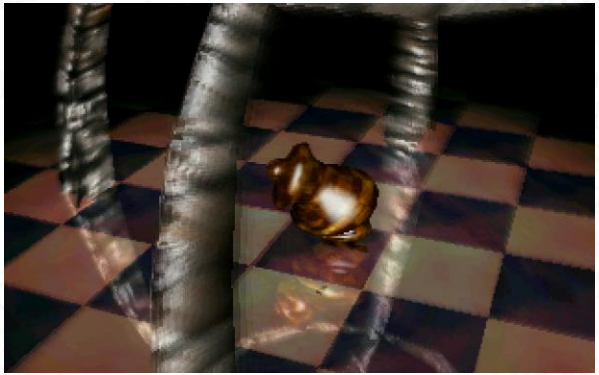
Contrast by Oxygene (1996)



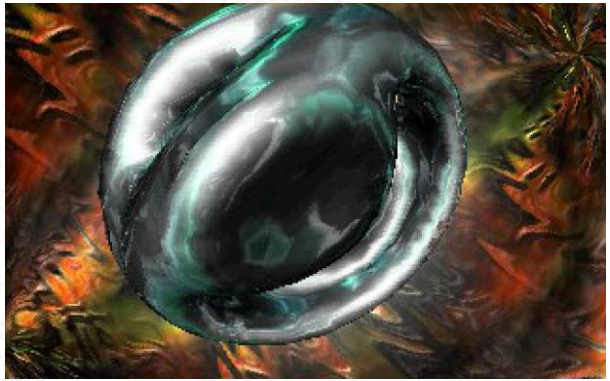
Inside by CNCD (1996)

ms-dos demoscene :: the transfiguration :: 1997

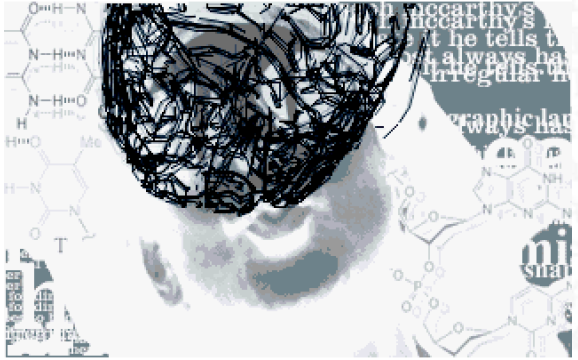
... getting backed by Pentium and High Color …



Boost by Doomsday (1997)



Stash by The Black Lotus (1997)



Square by Pulse (1997)

ms-dos demoscene :: the transfiguration :: 1998

... moving towards paying more attention design in lieu of code …

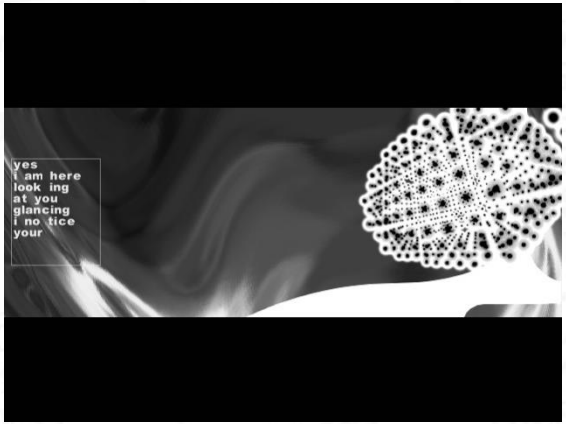
te-2rb by TPOLM (1998)

State of Mind by Bomb (1998)

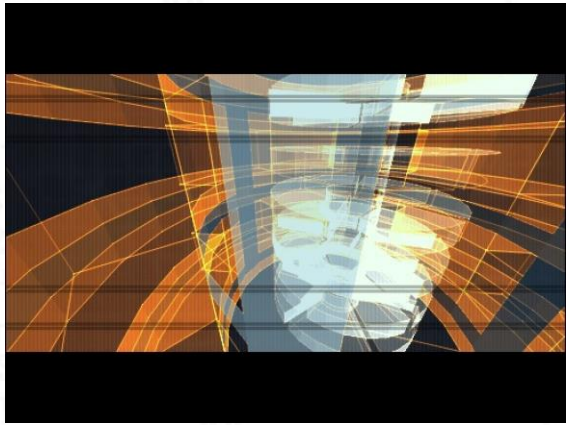
Kkowboy by Blasphemy & Purple (1998)

ms-dos demoscene :: the transfiguration :: 1999

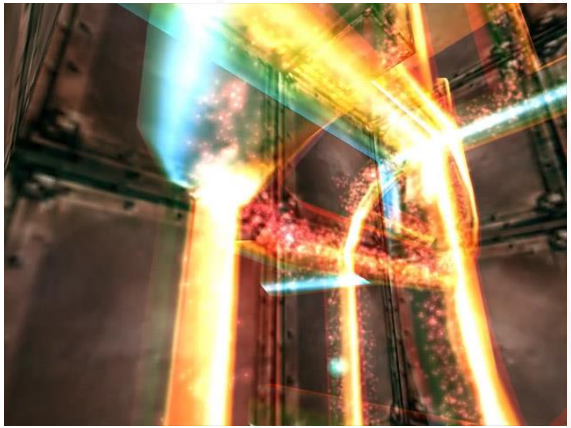
... and finally heading towards world of Windows …



bakkslide 7 by Hellcore & Omnicolour (1999)



moralhardcandy by Blasphemy (1999)



Discloned by Haujobb (1999)

ms-dos demoscene :: current state

during 1999-2000, the PC scene almost entirely moved on Windows, exploring new 3D accelerated horizons, running towards more polygons and shaders….

but in recent 10 years with a strong retroscene movement pushing from above, the DOS scene starts to resurrect (or kinda?...)

ms-dos demoscene :: current state

reserching platfoms known for so many years…

Rave On by Satori & da Jormas (2018)

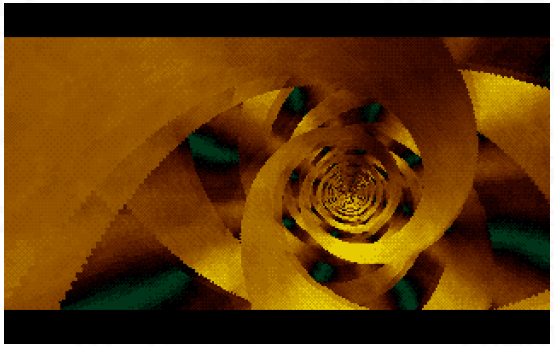
cöncept by Ümlaüt Design (2021)

8088 MPH by Hornet & CRTC & Desire (2015)

...and squeezing last bits from ancient PCs

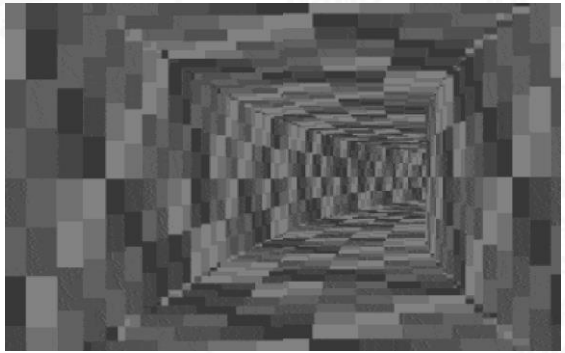
ms-dos demoscene :: current state

… as well as packing visuals to as less bytes as you can

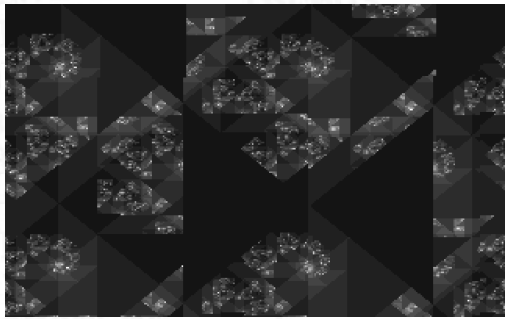


tube by 3SC (256b, 2001)

Gyroid by Řrřola (128b, 2021)



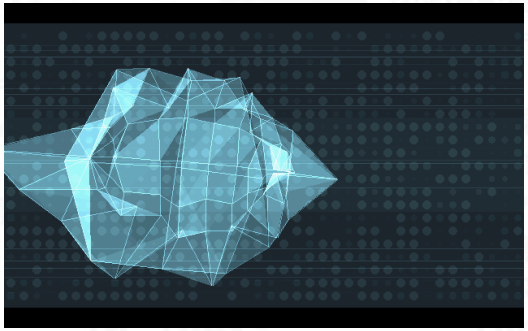
Memories by Desire (256b, 2020)



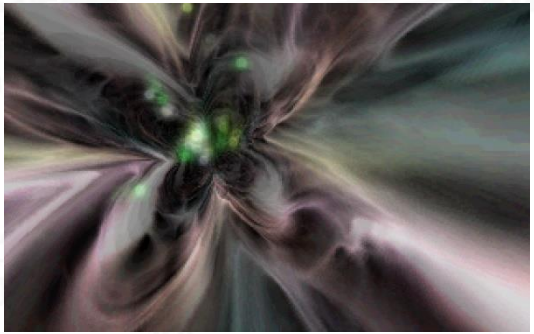
Blake 32 by Marquee Design (32b, 2021)

ms-dos demoscene :: my own experience

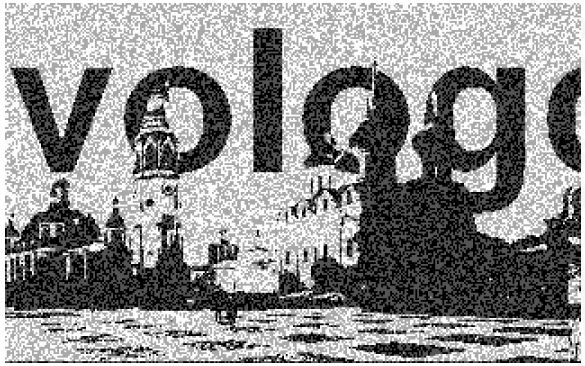
I also had quite a large scene career period associated with oldskool PC stuff (and still have :) so I've made a couple of demos back then…



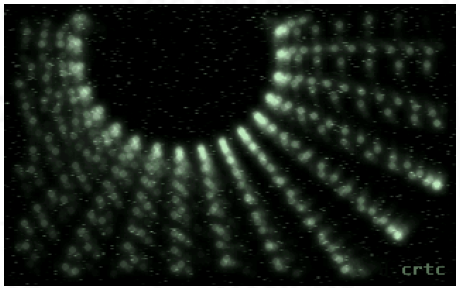
uribin (2016) – Pentium/VESA
2nd @ multimatograf 2016



volatile (2016) – Pentium/HiColor
1st @ Chaos Constructions 2016



#vologda8df (2015) – 286/EGA (partyhack :)
2nd @ multimatograf 2015

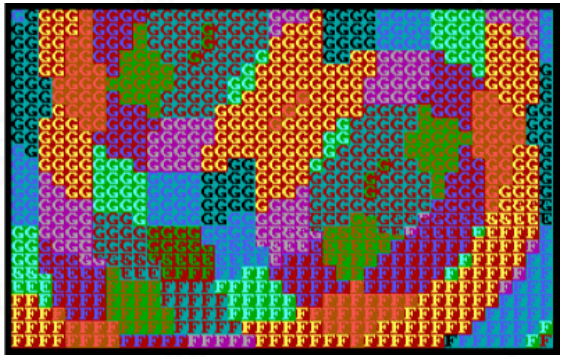


blash (2015) – 486/Pentium/VGA
1st @ Demosplash 2015

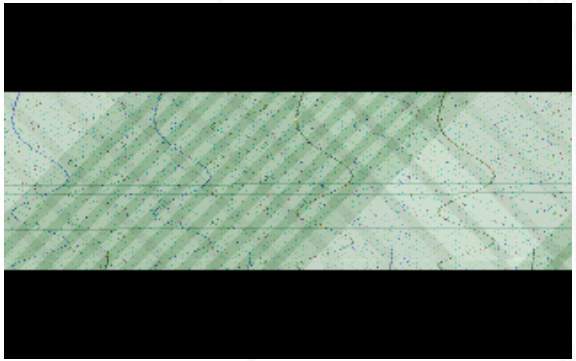
ms-dos demoscene :: my own experience

.. as well as tiny intros …

spirwd (2016) – 256b/FPU/VGA
2nd @ DiHalt 2016



pxor (2017) – 256b/8088/CGA!
7th @ Multimatograf 2017



mikrowave (2017) – HiColor
7th @ CC 2017

ms-dos demoscene :: my own experience

oh. by the way, all of my Watcom stuff had been open sourced :)
https://github.com/wbcbz7/watcom-stuff

computers :: ibm pc and xt

originally IBM didn't even intended to step on personal computer market

- initial proposals were purchasing design from 3rd party like Atari, later IBM internal "Project Chess" working group developed a prototype
- several CPUs were considered, notably TMS9900, MC68000 and i8086
- in the end, i8088 (striped-down 8 bit bus 8086 variant) was chosen

in August 12, 1981 - IBM PC was introduced, with i8088 CPU, optional 8087 FPU, and 64 KB of RAM (expandable up to 256 KB), 5.25 inch single/double side FDDs and MDA or CGA display adapter

- became a large success, sales exceeded IBM's expectations by as much as 800%, shipping 40,000 PCs a month at one point.
- shipped with PC-DOS (customized IBM version of MS-DOS), eventually made Microsoft the #1 software developer
- gained instant and widespread software and hardware support
  - open architecture and simple design extremely encouraged new developers

Later, in 1982, PC/XT entered the market

- Added internal hard disk, extended expansion capability, pumped memory capacity up to 640 KB

computers :: ibm pc/jr and tandy 1000

With the success of original PC, IBM tried to enter home and entertainment market with PCjr in 1984

- enhanced sound and graphics capabilities
- 64 or 128 KB of unified (CPU and video controller shared) RAM
- internal 5.25’ floppy drive and cartridge slot

ended up with numerous manufacturing and compatibility issues, becoming a failure

- nevertheless, some games take advantage of PCjr capabilities

Tandy 1000 was produced by Tandy Corporation for Radio Shack shops, introduced in 1984

- basically with same capabilities as PCjr, but fixed obvious flaws like allowing easier memory expansion and DOS compatibility
- lower priced than most PC clones, received moderate success
- later SX/LX models further fixed compatibility issues and added more features

computers :: ibm pc/at

Released in 1984, marks the start of new PC era

- new Intel 80286 CPU at 6 or 8 MHz
- extended XT system bus to 16 bit (later known as ISA)
- optionally bundled with new EGA graphics adapter
- enhanced IRQ and DMA support (16 IRQs vs 8 in XT, added 3 16-bit DMA channels)
- battery-backed real-time clock using MC146818 IC, with additional storage for date, time and system settings
- expanded memory capacity up to 16 MB
- 1.2 MB 5.25 FDDs and 16-bit hard disk controller
- still being functionally open design, actively cloned by many vendors
  - modern PCs also retain “AT compatibility” :)

computers :: ibm ps/2

introduced in 1987

- wide range of models – from 8086-based Model 20/25 to high-performance 486 and Pentium machines
- software compatible with PC and AT, much like other PC clones
- introduced 3.5 inch floppy drives, PS/2 keyboard and mouse interfaces, VGA display adapter and 72-pin SIMM memory modules
- moved from 8/16-bit ISA bus to proprietary MCA bus, increasing performance but also locking out independent vendors
- overall being more an attempt to control the entire PC market, ultimately failed
  - IBM then turned into niche PC vendor

computers :: meet the clones

After IBM PC success, other companies started to produce functional or full clones of original machine, often with expanded capabilities

- IBM published schematics and detailed operation info in Technical Reference manuals, only BIOS remained copyrighted
  - several vendors made clean room implementations, notably American Megatrends, Phoenix and Award Software
- 1982's Columbia Data Products MPC 1600 is the first 100% IBM PC compatible clone
- first i386 PC was produced by Compaq in 1986, year before IBM PS/2
- by the end of 80s, PC clones became dominant in market, and IBM eventually lost control under PC ecosystem
  - Intel and Microsoft became the leading force of PC evolution

MPC 1600

Compaq Deskpro 386

computers :: meet the clones

..and since then, everyone could built their own PCs…

graphics cards :: mda/hercules

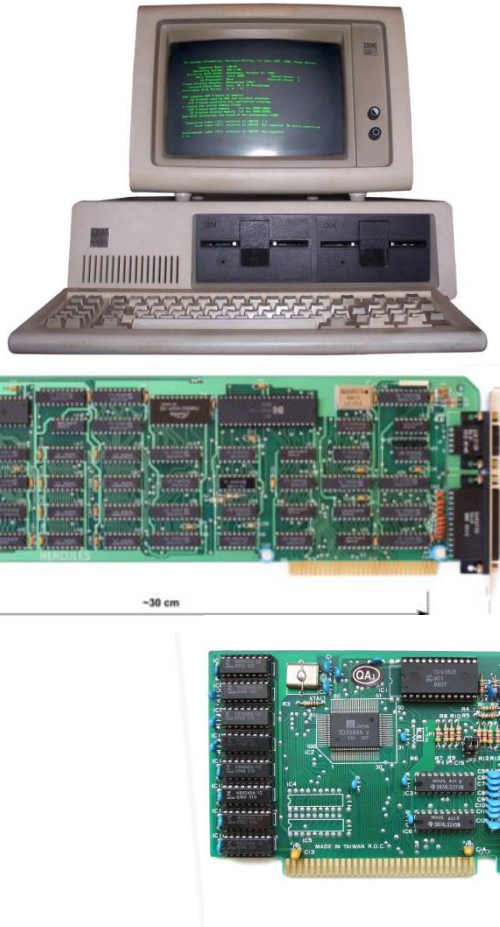
MDA introduced with the first IBM PC in 1981

- based on Motorola MC6845 CRTC
- supports only monochrome 80x25 textmode with 9x14 charset -> 720x350 resolution at 50 Hz
- included parallel port at 0x3BC
- 4kB of VRAM at 0xB0000

Later, Hercules released HCG with monochrome graphics mode, backwards compatible with MDA hardware/software

- 720x348 pixels, stored as 4-line interleaved bitmap due to 6845 limitations
- 64 kB of VRAM, with 2nd page at 0xB8000

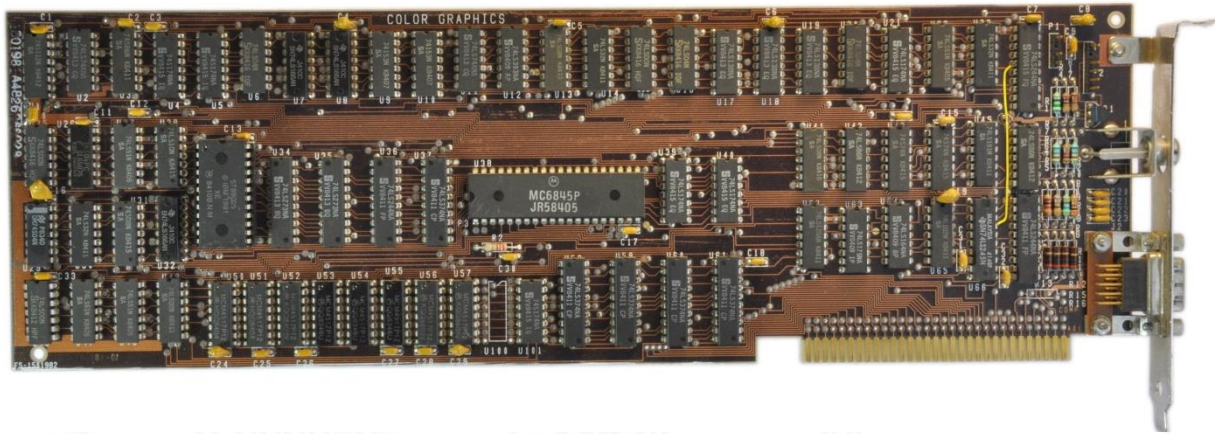
cheap HGC cards were popular as both low-cost business and dual-head soultion for software debugging

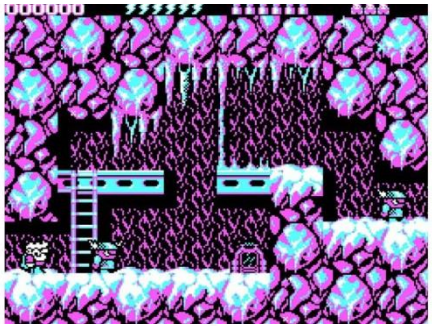


graphics cards :: cga

along with MDA, Color Graphics Adapter was also released

- also based on Motorola MC6845 CRTC, but more entertainment and home use oriented
- supports 40x25 and 80x25 text modes with 8x8 matrix -> 320/640x200 resolution @ 60 Hz
  - 80x25 mode uses entire DRAM bandwidth, “snow” artifacts apparduring screen updates while active area
- supports 320x200 graphics mode with 4 colors and 640x200 mode with 2 colors
  - + undocumented 160x100 16 colors mode (basically tweaked 80x25 textmode)
- 16kB of VRAM at 0xB8000, graphics mode layout is 2-line interlaced
- drives both RGBI TTL (15 kHz) monitors and composite NTSC TVs/displays
  - Due to simplified composite circuitry, hires modes create artifact colors, extending available colors to 1024 and more



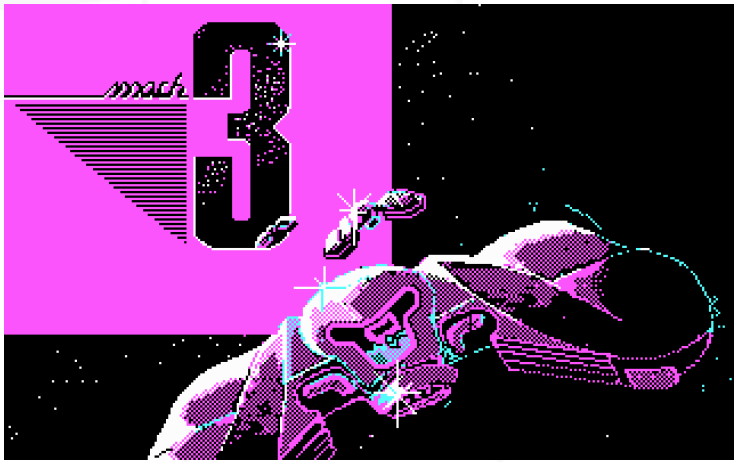


graphics cards :: cga :: magic trickery

by default, CGA support

- two palettes in 320x200 mode + selectable 0th color + two intensity levels
  - “monochrome” mode 5 actually activates 3rd palette on TTL monitor
- selectable foreground color in 640x200 mode

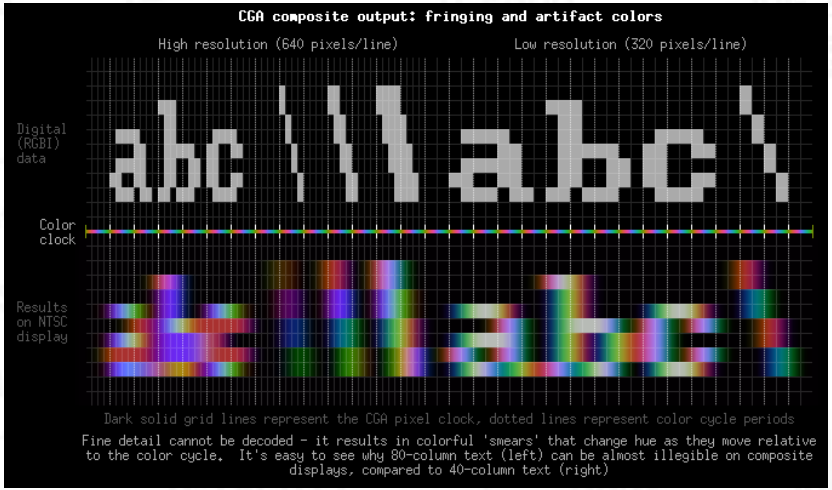
while it sounds pale enough, complex visuals can be delivered with clever artist work



graphics cards :: cga :: magic trickery

but as composite output lacks proper filtering, artifact colors are possible

- setting high-resoluition mode and using alternating patters allows to create artifact colors
- already often used in commercial games, like King's Quest
- abused to the limit and beyond in 8088MPH, giving 1024 simultaneous colors on screen!
- check great article by VileR (if you still didn't) for more details!



(screenshots from aforementioned write-up)

graphics cards :: pcjr/tandy 1000

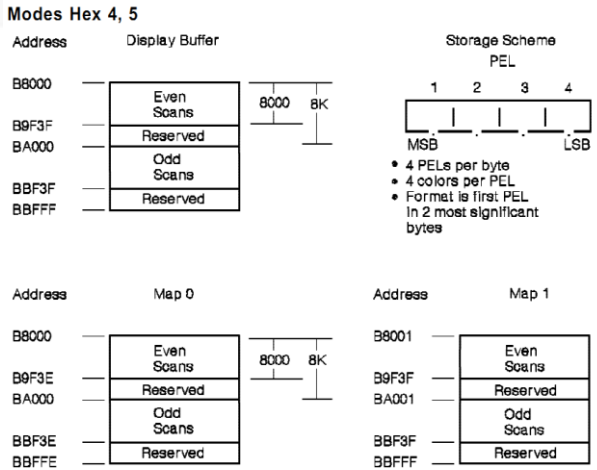
PCjr and Tandy 1000 both use similar graphics adapter, known as “Video Gate Array”

- like MDA/CGA, also based on Motorola MC6845 CRTC
- unified memory architecture – VRAM is shared with system RAM, MC6845 is responsible for DRAM refresh
- all 16 colors can be redefined through the palette to any of 16 RBGI colors
- three additional modes:
  - 160x200 16 colors, using 16 kB and 2-line interlaced, like CGA modes
  - 320x200 16 colors and 640x200 4 colors, 32 kB, 4-line interlaced (unavailable in 64kb models)
- composite and TTL RGBI output

graphics cards :: mda/cga/pcjr :: mc6845

Motorola MC6845 is one of most popular CRT controllers for personal and home computers

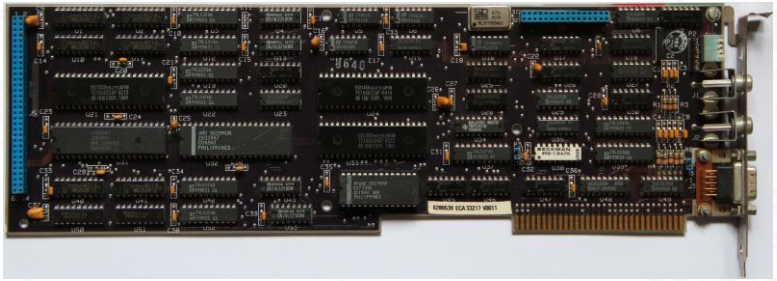
- originally intended for character based displays
- with additional circuity, graphics can be displayed
- used in several popular 8-bit home computers
  - BBC Micro and Amstrad CPC are notable examples
- had only 14 bit address support – constrained to 16 kB of VRAM for graphics mode without remapping
- maximum 7 bit vertical counter – limited to 128 rows without special tricks
  - remapping character row address to high bits of memory address increases vertical graphics resolution
  - that also explains “interlaced” framebuffer structure
- manufactured by several vendors (Hitachi, UMC, Amstrad), implementations differ in edge cases
  - known as CRTC types in CPC scene
  - i.e. MC8645 is used in old CGA revisions and HD6845 in new



graphics cards :: ega

introduced in 1984 along with IBM PC/AT line

- based on custom chip design
- four bitplane memory, 64 kB total in first model, later expanded up to 256 kB
- dual-sync design, 16 colors out of 16/64 color palette
  - 8x8 character textmode and 320/640x200 16 colors from 16 on older CGA/RGBI monitors
  - 8x14 char textmode and 640x350 16 from 64 colors for EGA displays, plus 80x43 enhanced textmode with 8x8 character size
  - monochrome 640x350 graphics/text for MDA/Hercules dispalys
- complex read/write logic and data latches offloading CPU and permitting fast VRAM copies
- programmable character set
- VBLANK interrupt (rarely used, deprecated in VGA)
- smooth scrolling, page flipping and split screen support
- feature connector for interfacing with additional h/w
- BIOS and memory layout-compatible with CGA



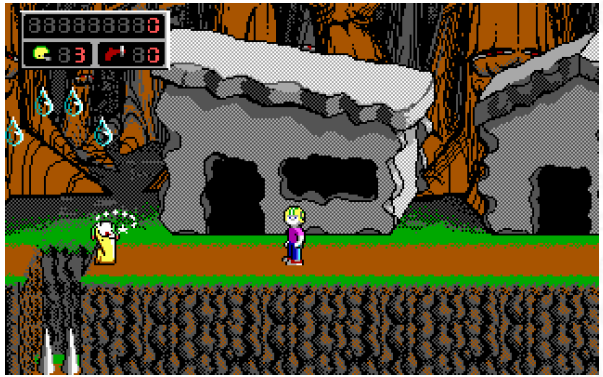
Original IBM EGA

ATi EGA Wonder

graphics cards :: ega :: art

due to required compatibility with older CGA monitors, 200-line graphics modes are limited to 16 colors

- that's why most EGA games often look comparable with PCjr/Tandy in terms of colors
- but thankfully to powerful EGA write logic and scroll capabilities, EGA games tend to feel more smoothly compared to PCjr titles
  - Commander Keen being notable example, showcasing novel Carmack's Adaptive Tile Refresh technique
- bitplane graphics also favored fast flat-shaded 3D



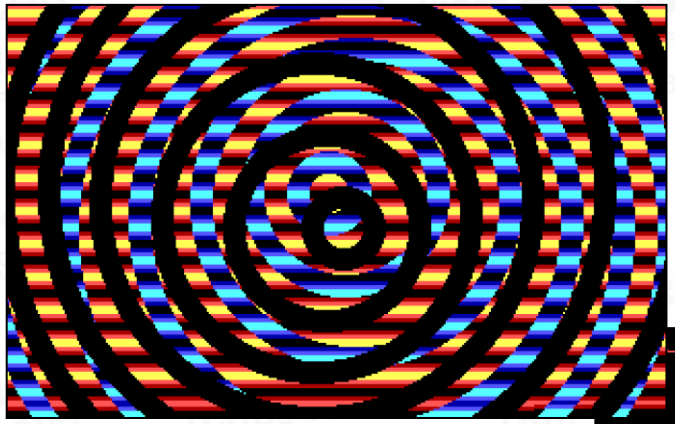
graphics cards :: ega :: art

as EGA monitors became widespreading, high resolution (640x350) games start to appear, making full advantage of enhanced RGB222 palette

- most ex-USSR EGA software are HiRes, as imported machines were equpped with EGA monitors
- strategic games also prefer EGA HiRes, like SimCity

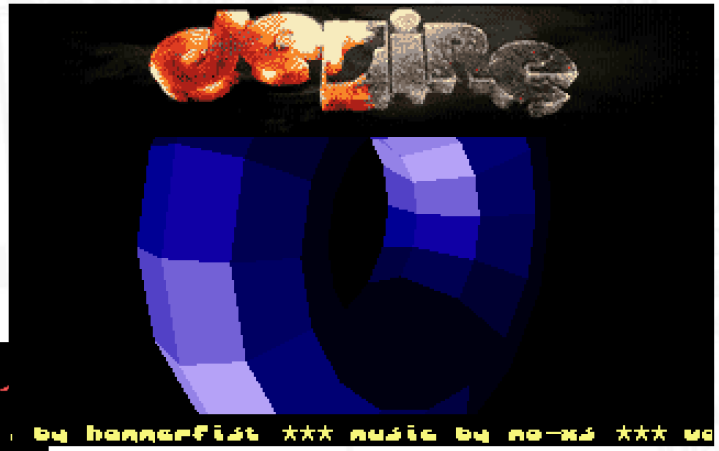
graphics cards :: ega :: demoscene

and moreover, there are plenty of demoscene releases for the EGA!



The MEGA-EGA Demo by S-Cubed (1994)

Gr8 by Future Crew (1990)

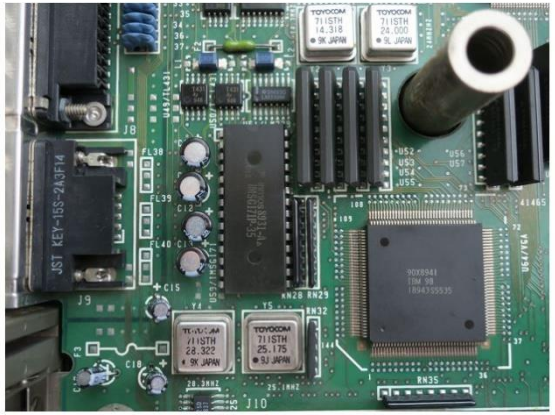


1991 Donut by Desire (2013)

graphics cards :: vga

introduced in 1987 in IBM PS/2 machines, later released as standalone (rare) ISA card

- single-chip ASIC design, versatile and widely cloned :)
- moved from digital to analog RGB, doubled scan rate to 31 kHz and vertical refresh to 70 Hz
- RAMDAC allowing up to 256 color from 262144 3x6-bit palette
- new graphics modes:
  - 8x16 character 40x25 and 80x25 text modes
  - double-scanned CGA/EGA compatible 320x200/640x200modes with enhanced palette capabilities (16 sets of 16 colors from 256K palette)
  - 640x480 2/16 colors 60 Hz high resolution and 1:1 pixel aspect mode
  - 320x200 256 color linear (chain-4) graphics mode + various Mode-X unchained 256 modes
- 256 kB of VRAM, 4 bitplane arranged
- register and memory layout compatible with EGA
- became an instant hit, copied and enhanced countless tiles, supported by almost every OS since 90s, still supported by modern graphics cards
- more in-depth VGA breakdown in “in depth” section



VGA integrated in IBM PS/2

Standalone IBM VGA card

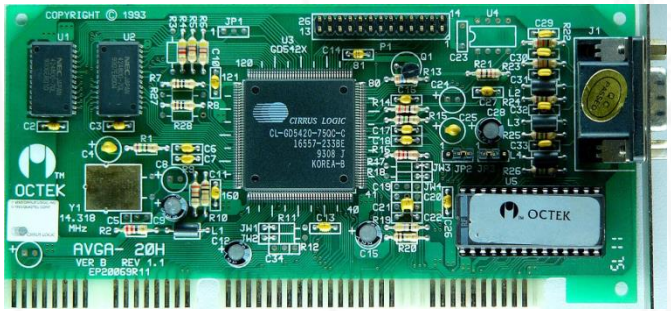
graphics cards :: vga :: super vga

original IBM VGA is pretty slow due to 8-bit CPU bus, and limited to low-res 256-color modes
several manufacturers copied VGA design, enhanced it with 16-bit host bus, FIFO buffers for optimizing VRAM access, faster and larger memory, higher resolution and color modes….

- commonly known as “Super VGA” cards
  - released from firms like Cirrus Logic, ATi, Tseng, Trident, Realtek…. varying in VGA compatibility but being totally incompatible between vendor SVGA modes
- VESA standartized SVGA features with the release of BIOS Extensions (1989 and onward)

Tseng ET4000AX (1989) – best performance at the time, SVGA landmark for several years

Trident TVGA9000C (1992) – low-cost range adapter with abysmal planar/Mode-X performance



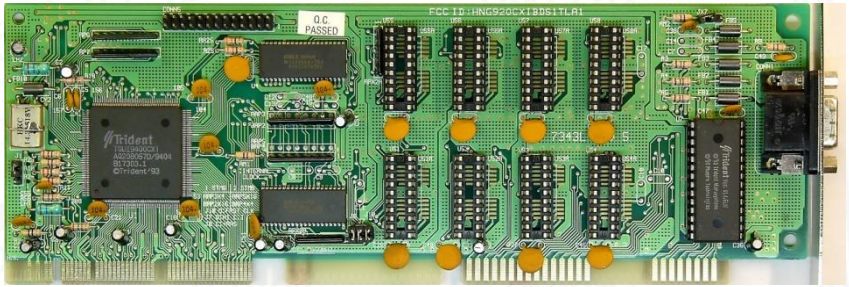
Cirrus Logic GD5420 (1992-1994) – solid low-cost Super VGA adapter, integrating VGA, RAMDAC and PLL in one IC

graphics cards :: vga :: super vga

… later being extended by 2D acceleration features, moving from 16-bit ISA to 32-bit VESA Local Bus….

S3 805 (1992) – mid-range SVGA 2D accelerator

Cirrus Logic GD5429 (1992-1994) – common mid-range 2D accelerator and solid VGA chipset



Trident TGUI9400CXi (1993) – low-cost 2D accelerator (and still DOOM decelerator :)

graphics cards :: vga :: super vga

… transitionsing by mid-90s to PCI …

Matrox Millennium (1995) – best 2D acceleration and image quality, moderate VGA compatibility

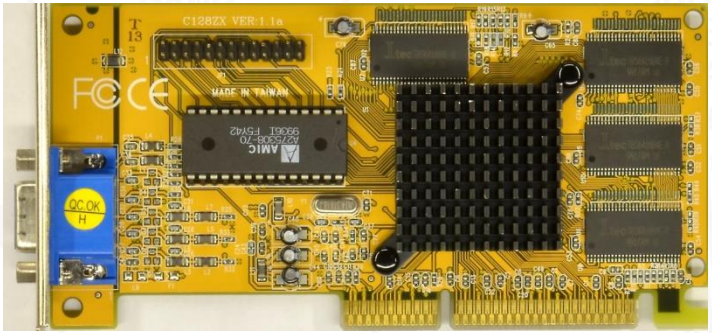
Tseng ET6000 (1996) – fast VGA performance, not so good VGA compatibility

S3 Trio64 (1994) – all-in-one chip, low-cost bestseller and perfect VGA compatibility

graphics cards :: vga :: super vga

… approaching 3D acceleration and moving even further...

3dfx Voodoo (1996) – standalone 3D-only accelerator, marking the era of fast and smooth 3D on the PC



NVIDIA RIVA 128 (1997-1998) – pioneering 3D, great and compatible VGA core

S3 ViRGE (1995) – miserable 3D, nice 2D and perfect VGA compatibility

sound devices :: pc speaker

the most distinctive and ubiquitous PC sound device :) supported all the way from first IBM PC to recent machines, and literally by every PC OS

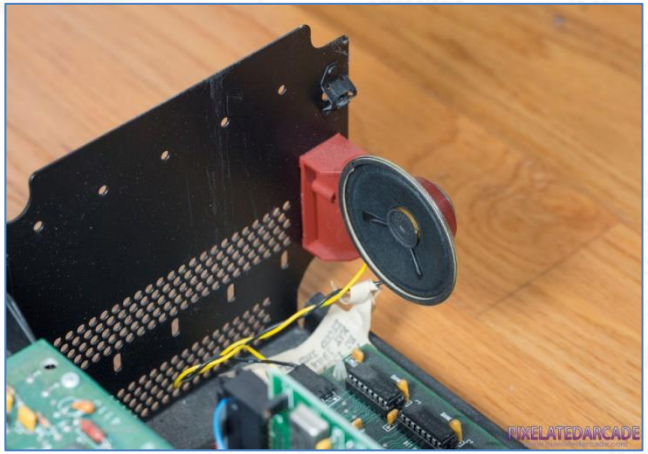
originally used for system diagnostics and simple beeps, later being “enhanced” to digitized PWM samples and multichannel music

from hardware side, it’s basically:

- 1-bit output from PIT Channel 2…
- ANDed with port 0x61 bit 0 (+ PIT gate @ bit 1)

but let alone PWM, the square wavy nature of sounds and one channel is more than enough!

(check System Beeps musicdisk and 8088MPH, also Monotone tracker



PC Speaker on original IBM PC 5150 (image by PixelatedArcade)

sound devices :: pcjr/tandy 1000

PCjr and Tandy 1000 machine were equipped with SN76489 programmable sound generator IC

- three square wave channels
- one noise channel, with 3 predefined frequencies + chained from secon channel
- 16 volume levels
- used also in BBC Micro, TI-99/4A and SEGA Master System

PCjr and Tandy 1000 games take advantage of enhanced sound capabilities, but since demoscene on those machines is still a rare thing, not much

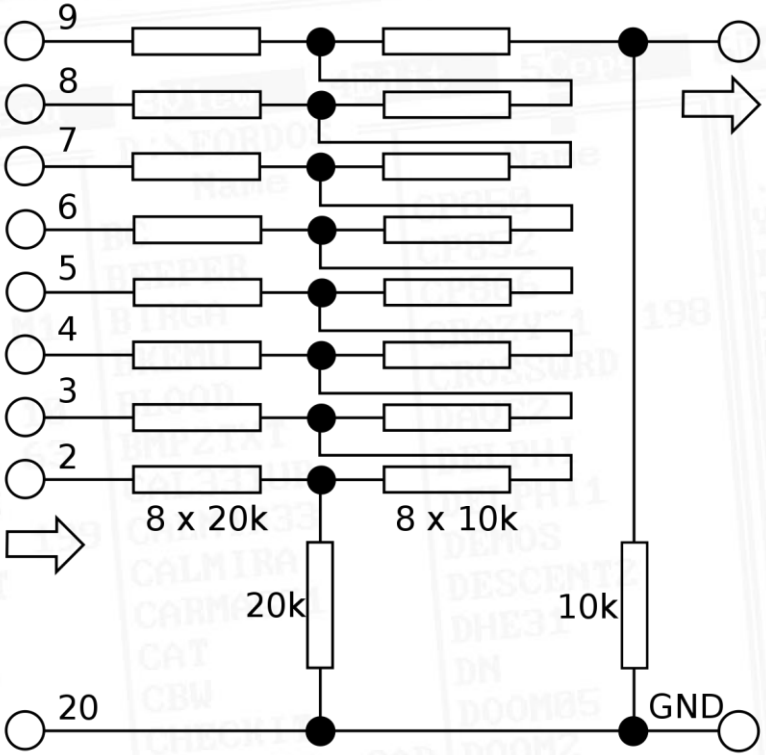
sound devices :: covox

a totally simple and cheap DIY 8-bit DAC for parallel port

originally introduced in 1987 as Covox Speech Thing, become widespread as DIY device

- provides full 8-bit sample resolution, (depends on DAC accuracy but still generally better than PC Speaker)
- enhanced versions (stereo on one/two LPT ports, DAC ICs, filters and amplifiers, ADC capability, ISA cards, etc)
- adapted and further enhanced for almost every 8/16-bit platform (Amstrad CPC, ZX Spectrum, MSX, etc)
- enhanced variant known as Disney Sound Source added 16-byte FIFO and built-in loudspeaker, partially compatible with original Covox but has own diffirences

sound devices :: covox :: build it yourself!



sound devices :: opl2/3

Yamaha YM3812 OPL2 FM synthesizer, introduced to PC with Adlib Music Synthesizer card in 1987

- 9 two-operator FM channels + percussion mode
- 4 waveforms, mono out
- no PCM output capability, although it's possible to get logarithmic 6bit DAC via FM operator tricks
- easy to program, requires long delays between register writes (doesn't matter much for XT, but becomes cumbersome on 286 and faster)
- supported by most ISA soundcards (popularized by original Sound Blaster, two OPL2 in SB Pro)

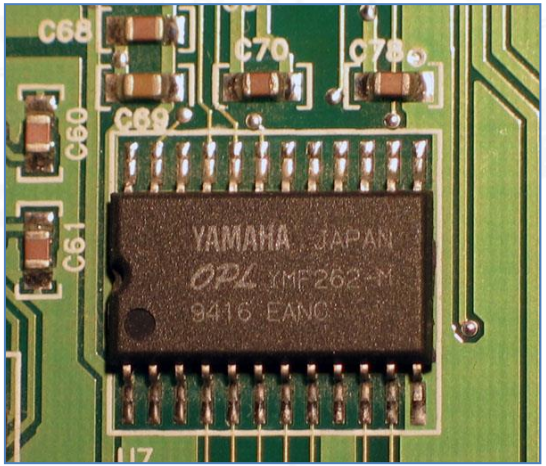
supported by majority of DOS games, frequently used in BBS intros and cracktros

sound devices :: opl2/3

later YMF282 (used in SB Pro 2/16 and Adlib Gold) is backwards software compatible with OPL2

- 18 2-OP or 6 2-OP + 6 4-OP FM channels, stereo capability, 4 new waveforms
- reduced access delays
- many later ISA sound chips implement custom OPL3 compatible cores (Creative CQM, ESFM, etc.), with varying sound faithfulness to the original OPL3

most OPL trackers supports OPL2 feature level (HSC, RAD, AMusic, Scream Tracker 3), others like Adlib Tracker support OPL3 features



Adlib Tracker II

sound devices :: sound blaster

released in 1989, introduced DMA PCM to PC market

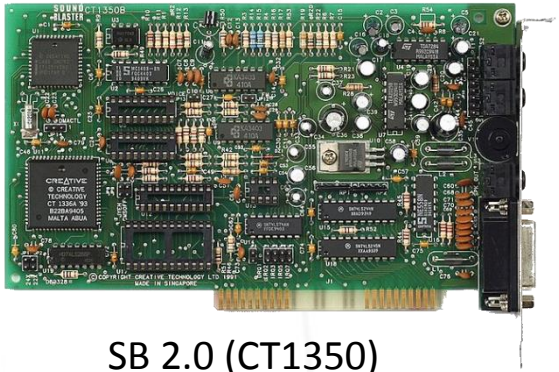
bundles OPL2 for Adlib Compatibility + two SAA1099, “DSP” chip (relabeled Intel 8051 MCU) handles PCM playback/recording + ADPCM playback + MIDI

originally supported max. 23 kHz 8 bit mono, later SB 2.0 added auto-init DMA capability for buffered playback + hi-speed mode for 44 kHz audio

- relatively easy to program, supports DMA/IRQ for CPU offloading

quickly become a market success, supported by many DOS games, demos are quite different story though

Original SB 1.0 (CT1320)



SB 2.0 (CT1350)

sound devices :: sound blaster pro/16

SB Pro finally added stereo PCM + OPL playback

- max 22 kHz 8 bit stereo or 44 kHz 8 bit mono
- de-facto standard across ISA soundcards, often enhanced with own extensions (ESS cards)

later SB Pro 2.0 uses single OPL3 instead of two OPL2

SB 16 introduced 16-bit CD-quality PCM + MPU-401 compatible MIDI interface

- 44 kHz 16 bit stereo, backwards compatible with SB 2.0 but not SB Pro's stereo mode

games rarely take advantage of 16-bit capabilities, but many trackers/demos/players support SB16 for better sound quality (still can't beat the GUS :)

SB Pro 2.0 CT1600

SB 16 CT2940

sound devices :: gravis ultrasound

the DOS demoscene golden standard :)

released in 1992, used GF1 chip + onboard RAM for wavetable synthesis and sample playback

- 32 wavetable channels with linear interpolation, volume, pitch and panning control, up to 44 kHz 16 bit hardware mixing (full 32ch is 19 kHz though)
- up to 1 MB of onboard RAM, 8/16bit samples
- No Adlib/SB compatibility, PCM streaming has to be done by free wavetable channel + DMA
- very easy to program – ideal for tracked music

almost all post-1993 demos support GUS (often as sole sound device), as well as trackers and players

Classic GUS with 1 MB RAM

GUS Ace

sound devices :: gravis ultrasound :: max/pnp

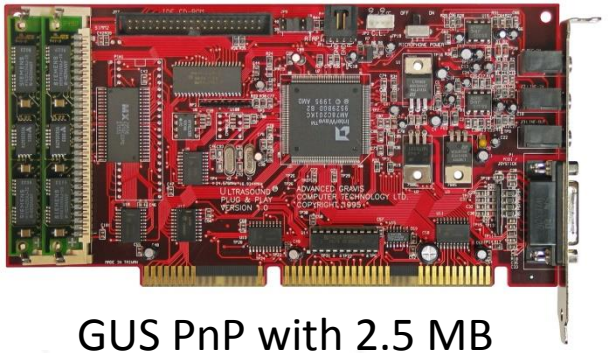
GUS MAX bundled CS4231A codec for streamed PCM capability and better SB emulation

in 1995, AMD develops Interwave IC, used in GUS PnP and several 3rd party cards

- full 32 channels at 44 kHz + MPU-401 MIDI
- up to 16 MB of sample RAM (upgradeable with 30-pin SIMMs) + 1 MB ROM
- backwards compatible with GF1

some demos/players support Interwave natively, other will work fine in classic GUS mode

GUS MAX



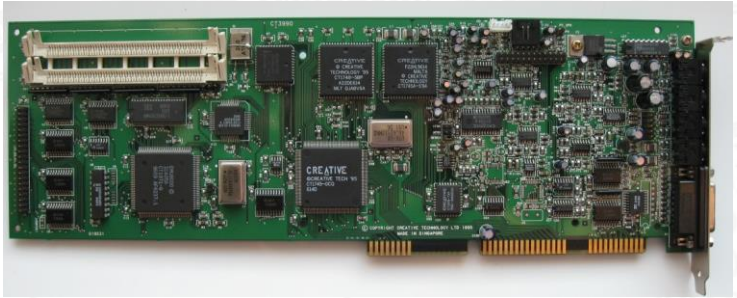
GUS PnP with 2.5 MB

sound devices :: sound blaster awe32/64

basically SB16 + EMU8000 wavetable synthesizer

- 32 wavetable channels with 4-point interpolation, volume, pitch and panning control, up to 44 kHz 16 bit mixing + resonant filters + effect processing
- 16-bit only always-looped samples, up to 28 MB RAM + 1 MB ROM
- Creative originally didn't disclosed register details, several people reverse engineered EMU8000 internals then finally official hardware reference has been released
- needs complex init compared with GUS, wavetable limitations (max 176 kHz pitch, 16 bit samples, etc.)

not as common as GUS, supported by some trackers (e.g. IT2) and demos



SB AWE32 (CT3990)

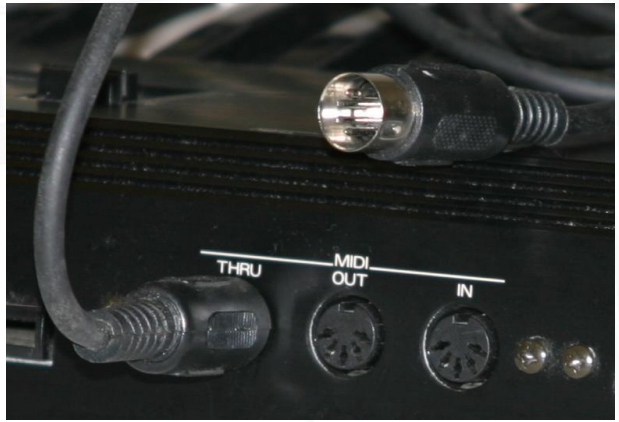
SB AWE64 Value (CT4380)

sound devices :: midi

originally used for music equipment control

- 1984 – Roland MPU-401 – universal MIDI interface, supports many platforms, incl. PC, support UART and intelligent accelerated mode
- 1987 – Roland MT32/LAPC-1 – reference music synth for early DOS games
- 1991 – General MIDI – standardizes instrument definitions and controls, Roland General Sound expands instrument set further
- incredibly easy to program, popular in sizecoding

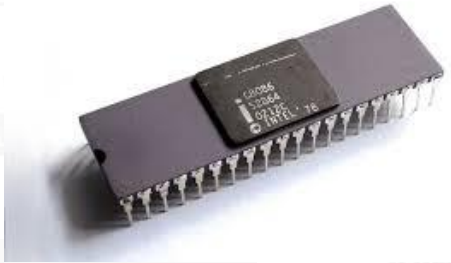
late-90s ISA soundcards usually implements MPU-401 in UART mode, some (GUS/AWE32) able to emulate through wavetable engine

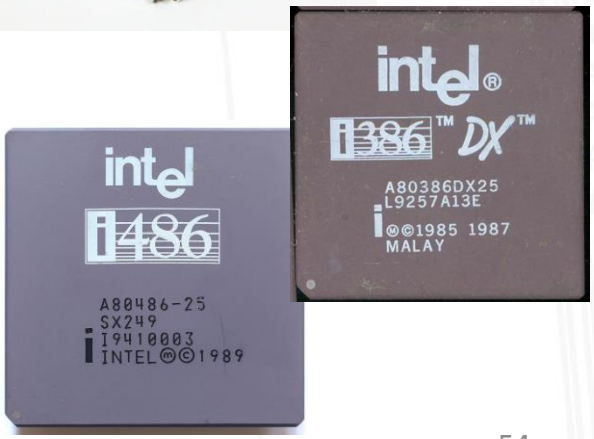


Roland LAPC-1

in depth :: x86 :: cpu evoluition

- Intel 8086 (1978):
  - greatly enhanced i8080 design, used as basis for IBM PC family
  - i8088 is 8-bit data bus cost-effective variant of i8086
- Intel 80286 (1982):
  - introduced protected mode, increased performance relative to i8086
  - increased address space to 24 bits (up to 16 MB)
- Intel 80386 (1985):
  - extended x86 architecture to 32 bits, added virtual memory paging
  - available in two versions, SX with 16 bit data bus and full-blown DX
  - 32-bit address space, addressing up to 4 GB of memory
- Intel 80486 (1978):
  - pipelined design, instruction timings improved up to 1 clock per simple instructions
  - Integrated level-1 cache (8 KB, 16 KB
  - integrated FPU on die (except SX versions), clock multiplier in DX2/DX4

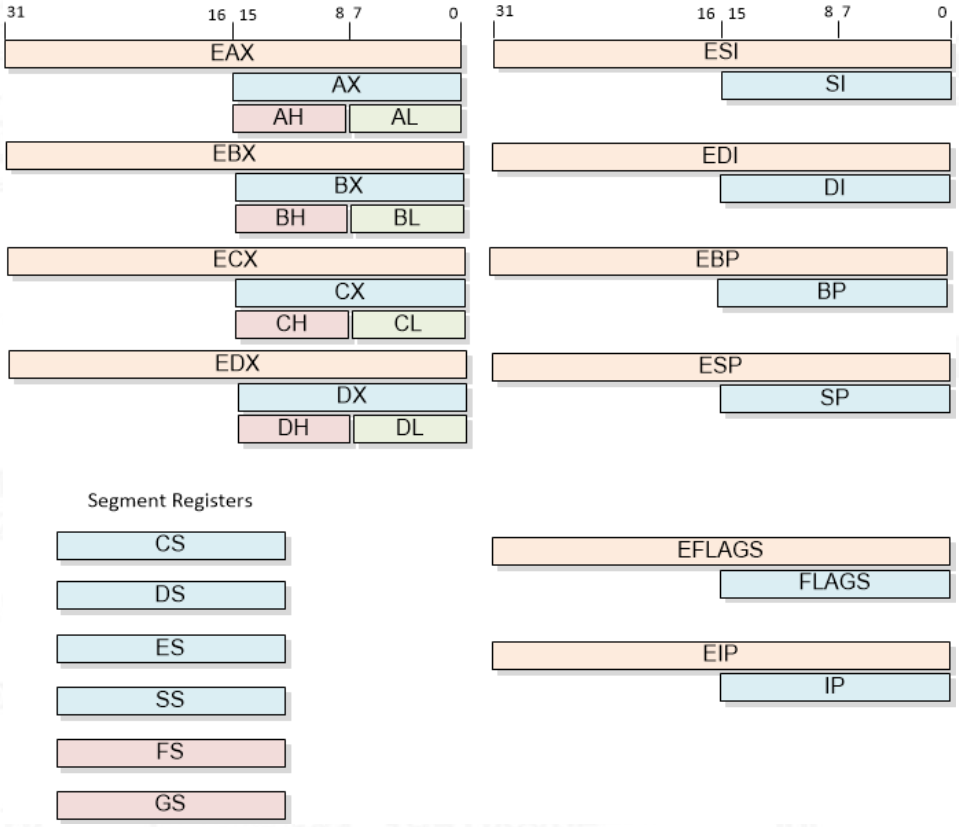




in depth :: x86 :: cpu evoluition

- Pentium (1993):
  - superscalar in-order architecture, able to execute up to two instructions per clock
  - completely redesigned pipelined FPU
  - 64-bit data bus
  - Pentium MMX (1997) added integer SIMD instructions for accelerating multimedia and data processing tasks
- Pentium Pro/II (1995/1997):
  - out of order superscalar architecture, highly optimised for 32-bit software
  - integrated level-2 cache (up to 512)
  - Pentium II moved cache on CPU cartrige and added MMX instuctions
- AMD K6 (1997):
  - out of order superscalar design, outperforming Pentium in integer performance but performing significally slower in FPU intensive tasks
  - K6-2 (1998) introduced 3dNow! floating point SIMD instructions

in depth :: x86 :: registers



7 general purpose registers,
1 stack pointer

i386 extended all general registers to 32 bit and added FS/GS registers

- used for system control structures

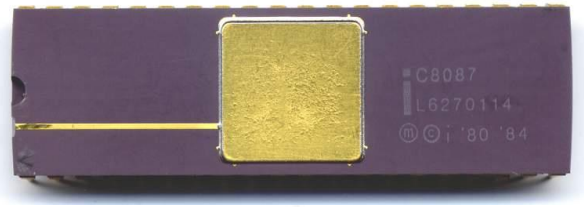
relatively low count compared with other platforms, like MC68000/ARM

in depth :: x86 :: fpu

i8087 numeric processing unit provides floating point mathematical operations accelerating

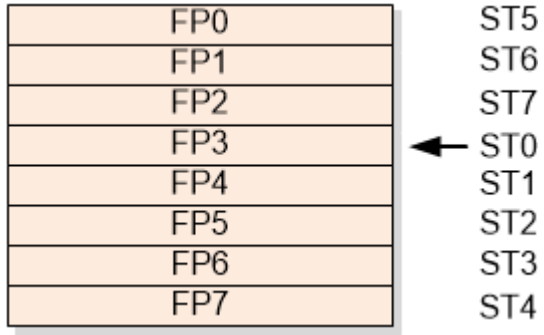
operates on single precision (32 bit), double precision (64 bit) and extended precision (80 bit) numbers, also supports conversion between integer and floating point variables

- provided basis for IEEE-754 floating point standard



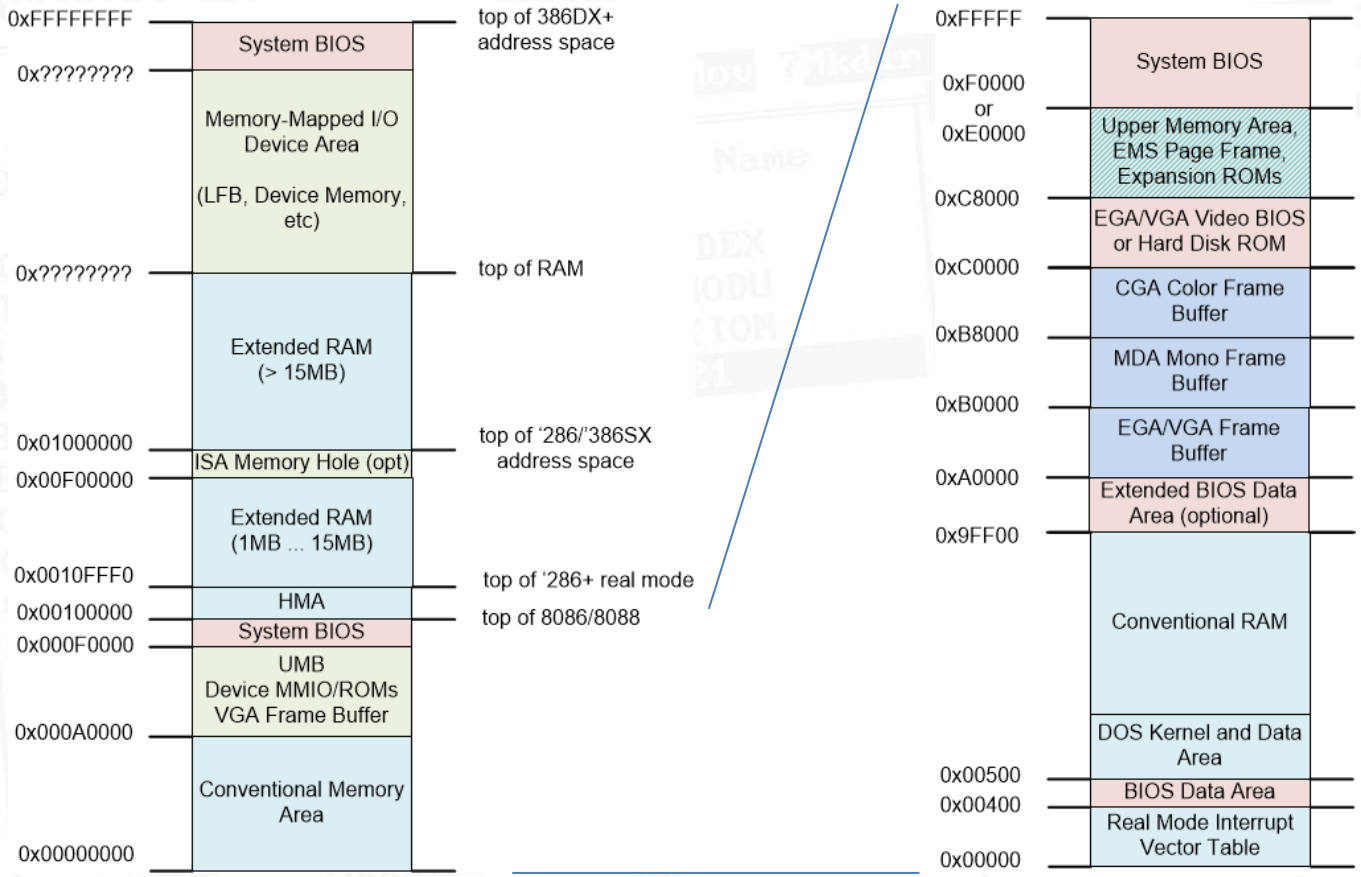
registers are organized as stack of 8 80-bit registers (ST0-ST7), ST0 used as destination operand, and also can be exchanged with ST1-ST7 with FXCH instruction

FPU operations pop values off the stack and push results onto it, also providing integer/floating point variables load and store



- i387 added direct sin/cos operations, making FPU fully IEEE-754 compliant
- starting from i486, FPU is integrated on CPU core (except 486SX)
- Pentium updated FPU with new pipelined architecture and register renaming, paring FXCH with FPU operations and performing slow instructions (divide, square root) in parallel with integer operations

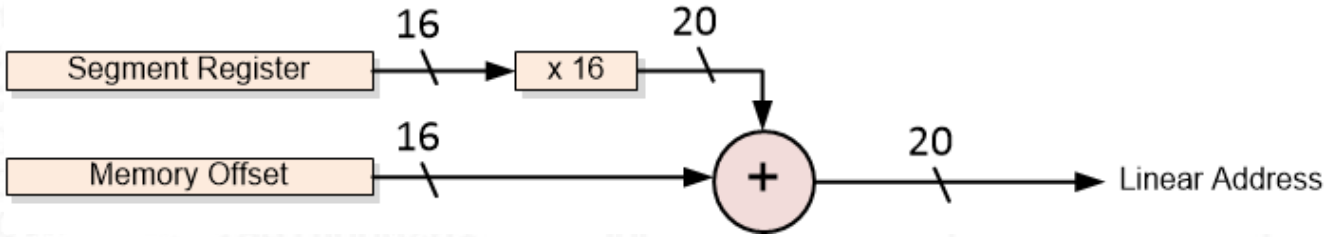
x86 :: memory layout



x86 :: memory model :: real mode

In real mode, memory pointer is constructed from two components: segment and memory offset

- segment value defines memory start address in units of 16 byte (paragraphs)
- then, 16-bit or 32-bit offset is added to resulted segment value to form memory linear adress
- segments can overlap: i.e. 0x0000:0482 and 0x0040:0082 point to same memory address 0x000482
- i286 and higher calculate addresses in full 24/32bit: address overflow is possible with segments > 0xF000, allowing access to 64 Kb (minus 16 bytes) above 1 MB
  - called High Memory Arena (HMA) in DOS world
  - on 8086/8088, addresses beyond 1 MB are wrapped to start of memory: some older software depend on this behavior, and IBM implemented A20 line gate to emulate such wraparound



x86 :: memory model :: real mode

as following, memory pointers in x86 are of two types:

- near pointer (16 or 32 bit), referencing memory within current selected segment
- far pointer (16 bit segment + and 16\32 bit offset) referencing memory across different segments

thus, data and code can be either located in one segment (utilizing near pointers), or split in several segments (far pointers)

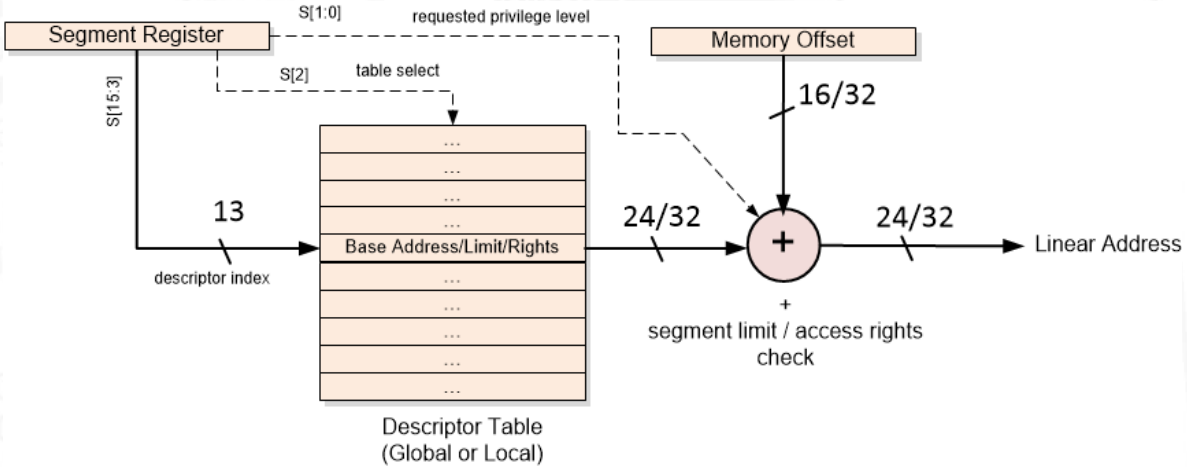
16 bit real mode applications can be compiled using 6 distinctive memory models

Model	Data	Code	Definition
Tiny	near	near	CS=DS=SS, used in .com files
Small	near	near	DS=SS
Medium	near	far	DS=SS, multiple code segments
Compact	far	near	single code segment, multiple data segments
Large	far	far	multiple code and data segments
Huge	huge	far	multiple code and data segments; single array may be >64 KB

x86 :: memory model :: protected mode

In protected mode, segment registers do not contain offset, instead, descriptor selector is used

- defined as 13-bit selector field, Global/Local Descriptor Table select bit and requested privilege field (2 least bits)
- each GDT/LDT entry contain of 8 bytes descriptor with defined stasrt offset, limit, descriptor type and access rights (whether is executable, read, write, etc., privilege level, etc)
- CPU checks access rights and descriptor validity while segment register usage, raising General Protection Fault in case of failure



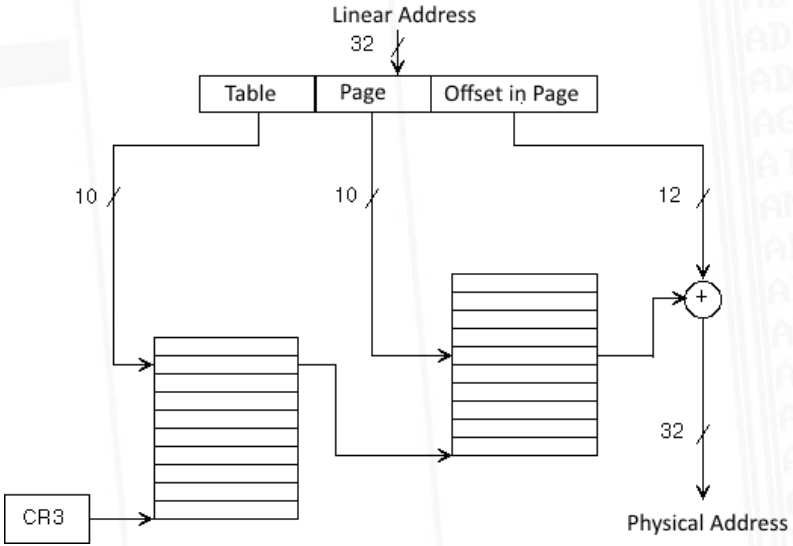
x86 :: memory model :: protected mode

while segmentation still was crucial for 286 because of 16 bit offsets, 386 and 32-bit modes do not generally need it

- “flat” mode involves setting CS to code segment and DS/ES/SS with data segment, both with base 0 and limit of 4 GB, effectively bypassing segmentation and treating every logical address as linear

i386 also introduced Memory Management Unit allowing to map virtual memory to physical in units of 4 KB pages

- EMM386, Windows 3.x in 386 Enhanced mode, Windows 9x/NT and Linux/FreeBSD use paging for isolating applications address space
- another notable i386 feature is Virtual 8086 mode, emulating real mode environment as dedicated process task, eliminating the need if expensive mode switches for DOS/BIOS calls



x86 :: memory model :: flat real mode

apparently, 80386 and newer CPUs use same segment descriptor pipeline for both real and protected mode, differing only in address calculation

- each segment register has associated descriptor cache
- it's possible to enter protected mode, set all data segments to 4 GB limit and switch back to real mode, allowing to access data over first MB while retaining compatibility with DOS/BIOS services and interrupts
- requires raw or XMS environment, most PM supervisors (like EMM386/Windows) don't allow such tricks and either reset segment limits to 64 KB upon return to RM or terminate the application
- used in several early (pre-1995) demos and even some commercial games (Ultima 7/8)
- expanding CS limit over 64 KB or switching to 32 bit code segment is practically useless (RM IRQs corrupt high 16 bits of EIP, and DOS/BIOS code won't work in 32-bit real mode)
  - nevertheless, there is at least one demo (Project Angel by Impact Studios, 1994) using 32-bit flat real mode
- With the availability of DOS extenders, 32-bit protected mode is more attractive and easier to use than flat real mode

in depth :: x86 :: i/o ports

Like i8080, 8088 has separate I/O and memory address

- up to 65536 byte, 32768 word and 16384 dword input/output ports (16 bit address space)
- originally used 10-bit address decoding, then being enhanced to 12 bit
- in original architecture, ports 0x00-0xFF are reserved for motherboard devices, ports 0x100-0x3FF are designated for expansion devices
- relatively slow compared with memory-mapped IO, most of todays PC devices abandoned I/O space in favor of MMIO

in depth :: interrupts

hardware interrupts serviced by Intel 8259A interrupt controller

- up to 8 hardware interrupts, hardware interrupt mask, selectable interrupt priority
- XT uses one interrupt controller (IRQ 0-7), AT added another (IRQ 8-15)

x86 CPU supports up to 256 interrupt vectors

- first 32 vectors are reserved for CPU exceptions
- IRQ 0-7 are mapped to INT 08-0F, IRQ 8-15 mapped to INT 70-77
  - conflicts with exception vectors – doesn't matter much in real mode, needs to deal with or remap in protected mode
- several software INTs used for BIOS and DOS services

protected mode enhanced interrupt capabilities

- able to switch between privilege levels, providing system call facilities

in depth :: interrupts

IRQ#	Interrupt	Function
IRQ0	8	PIT timer
IRQ1	9	keyboard service required
IRQ2	A	EGA/VGA vertical retrace
IRQ8	70	real time clock
IRQ9	71	software redirected to IRQ2
IRQ10	72	free
IRQ11	73	reserved (AT,XT286,PS50+)
IRQ12	74	PS/2 mouse interrupt
IRQ13	75	numeric coprocessor error
IRQ14	76	Secondary IDE controller
IRQ15	77	Primary IDE controller
IRQ3	B	COM2 or COM4
IRQ4	C	COM1 or COM3
IRQ5	D	sound card
IRQ6	E	floppy disk drive
IRQ7	F	LPT1 or sound card

INT #	Function
10	**video services**
11	equipment resuest
12	memory size
13	**disk I/O service**
14	serial communications)
15	system services, cassette
16	keyboard services
17	parallel printer services
18	ROM BASIC loader
19	bootstrap loader
1A	Time/Date inter
1F	graphics character table
20	DOS general program termination
21	**DOS function API**
22	DOS multiplexer
31	DOS Protected Mode Interface
33	mouse support
67	EMS specification

in depth :: isa dma

based on Intel 8237 DMA controller with external 74LS612 memory mapper

- PC and XT have 4 8-bit DMA channels, AT added 4 16-bit channels
- original i8237 is unable to work with more than 64 KB of RAM, so mapper supplies high order address bits
- because of this, DMA transfers cannot cross 64 KB boundary, and work only in first MB (8-bit) or 16 MB or RAM (16-bit)
- used in PC and XT for DRAM Refresh and floppy/hard disk transfers (channels 0 and 2 respectively), AT
- due to slow speed and compatibility issues, used mostly by low rate devices such as sound cards, network interfaces and floppy disk controllers

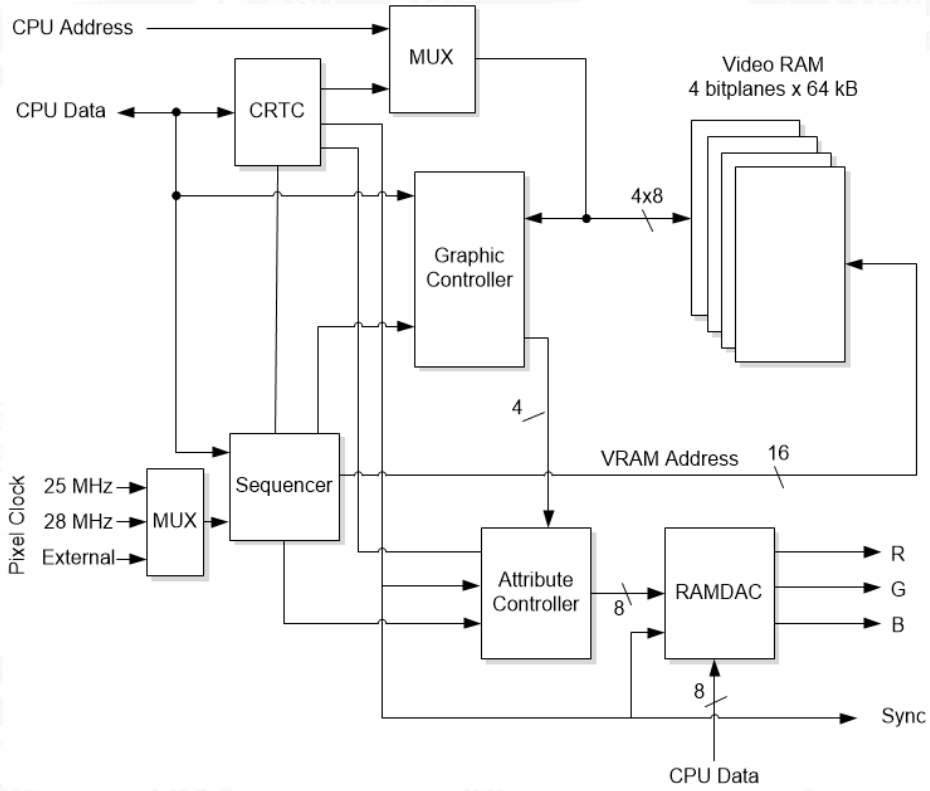
in depth :: timing

PCs are usually employ several timing methods:

- Programmable Interval Timer Channel 0 at IRQ0
  - running at 18.2 Hz by default, used by BIOS for disk timeouts and time counting
  - the primary and precise enough universal timing source
  - often being used by TSRs and sound systems, needs precautions for timer sharing
- RTC timer at IRQ8 (AT and higher)
  - counts at rates from 8 up to 8192 Hz, with power-of-two frequency step
  - unused by BIOS routines by default, easy to interface (example in rtc_test folder)
- Time Stamp Counter (Pentium and higher)
  - Incremented each CPU clock, available by RDTSC instruction in EDX:EAX
  - needs calibrating to other timing source (i.e. IRQ0) before use, provides extremely precise timing, very useful in profiling
- VGA Vertical Retrace bit in Input Status Register (port 0x3DA bit 3)
  - unfortunately VBLANK IRQ is deprecated on VGA and unavailable, and port polling can result in retraces being mussed, causing slowdowns

in depth :: vga

your working horse for DOS journey :)



in depth :: vga :: access

Memory Address	Description
0xA0000 – 0xAFFFF	EGA/VGAs framebuffer
0xB0000 – 0xB7FFF	Monochrome framebuffer
0xB8000 – 0xBFFFF	Color text and CGA famebuffer

I/O Port Address	Description
0x3C2/0x3CC	Miscellanous Output Register
0x3D4-3D5/0x3B4-3B5	CRT Controller Index/Data
0x3C4-3C5	Sequencer Index/Data
0x3DA	Input Status Register
0x3CE-3CF	Graphics Controller Index/Data
0x3C0/0x3C1	Attribute Controller Index/Data (combined)
0x3C6-3C9	RAMDAC Registers

in depth :: vga

CRT Controller

- generates all the necessary display timings, provides memory address for display refresh

Sequencer

- generates basic display timings (dot and character clock), arbiters VRAM accesses between CPU and CRTC, protects bitplanes for being changed

Graphics Controller

- interfaces between display memory and both attribute controller and CPU, converts and translates VRAM data, performs complex CPU write/read logic for speeding up access

Attribute Controller

- formats input graphics and text data to pixel stream, handles blinking, underline and cursor logic

RAMDAC

- converts digital pixel stream throung color lookup table to RGB triplets, then converting RGB digital data to analog

Many SVGA cards add additional functions, e.g. 2D/3D acceleration, fast FIFO interface to VRAM, etc.

in depth :: vga :: graphics controller

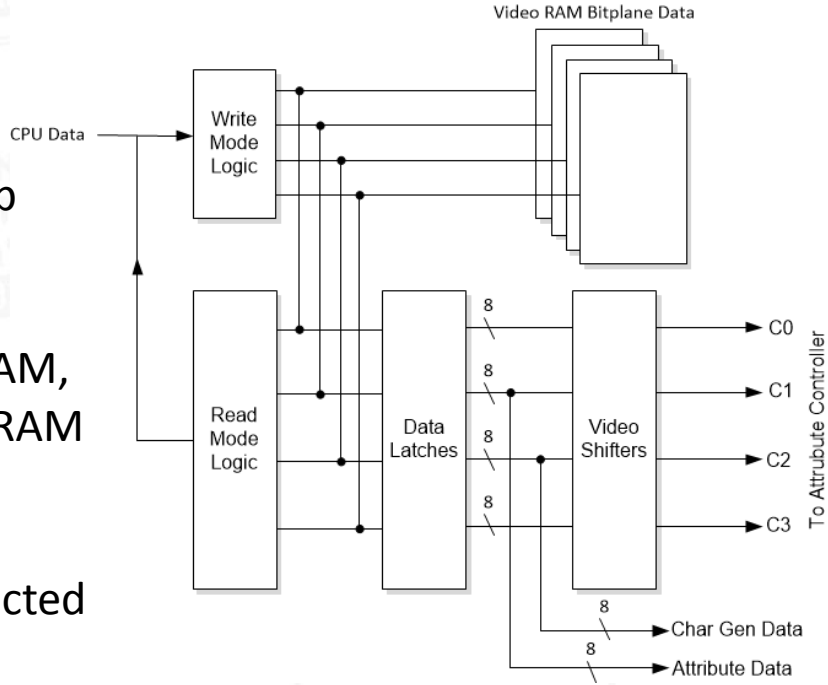
Setting individual pixels with bitplane structure is slow

- Read from all 4 bitplanes, modify, write back

VGA includes special circuity for speeding up common graphics tasks

Latches work as 4 8-bit buffers between VRAM, and graphics controller logic, loaded with VRAM data at CPU read address

Read modes include either reading one selected bitplane, or comparing latches data against reference color, allowing pattern searching and fast flood fills

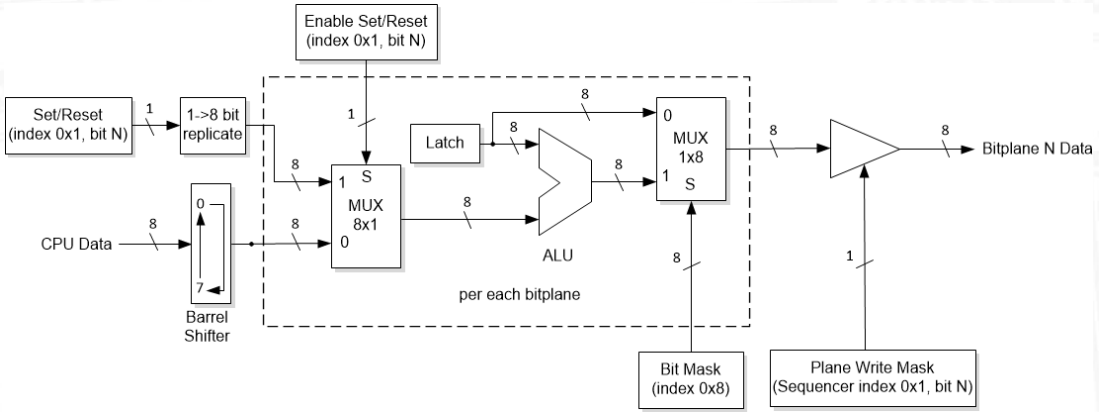


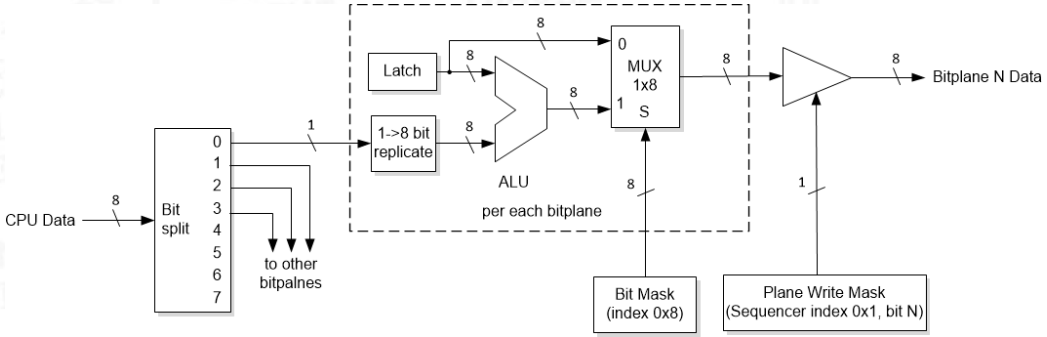
in depth :: vga :: write modes

Write Mode 0:

- CPU data rotation
- Individual plane Set/Reset control for plane masking
- ALU operations: move\AND\OR\XOR
- Masking individual pixels by bit mask

ideal for pattern fill





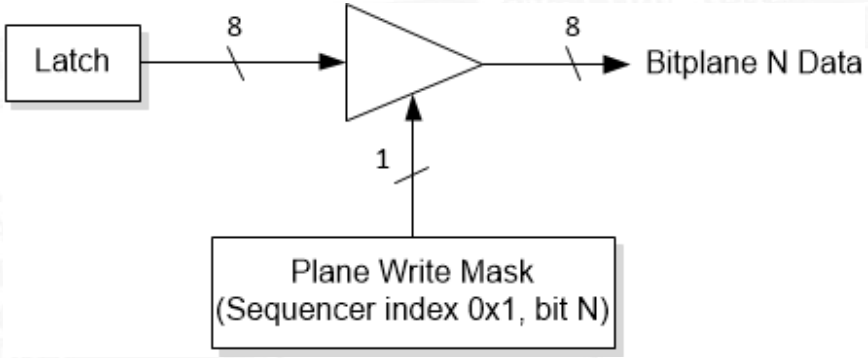
Write Mode 2:

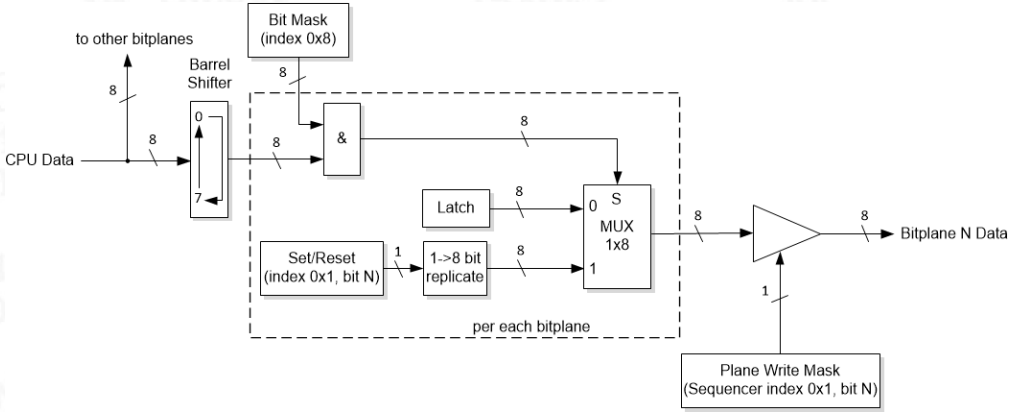
- CPU data replicating “color” bitwise for all pixels in 8x1 block
- ALU operations: move\AND\OR\XOR
- Masking individual pixels by bit mask

suitable for plotting individual 16c pixels or Mode-X line drawing

in depth :: vga :: write modes

Write Mode 1:
direct latch->VRAM copy
as latches are loaded for each byte read from memory, this can be used for fast VRAM->VRAM blits





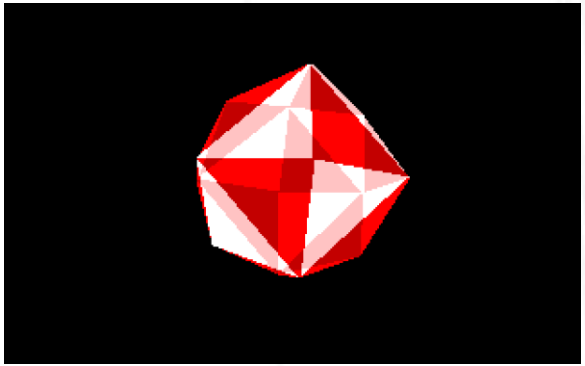
Write Mode 3:
CPU data rotation and masking with bit mask, used as bitwise “selection” between latch data and constant color suitable for pattern or line drawing in 16 color modes

in depth :: vga :: 16 color modes

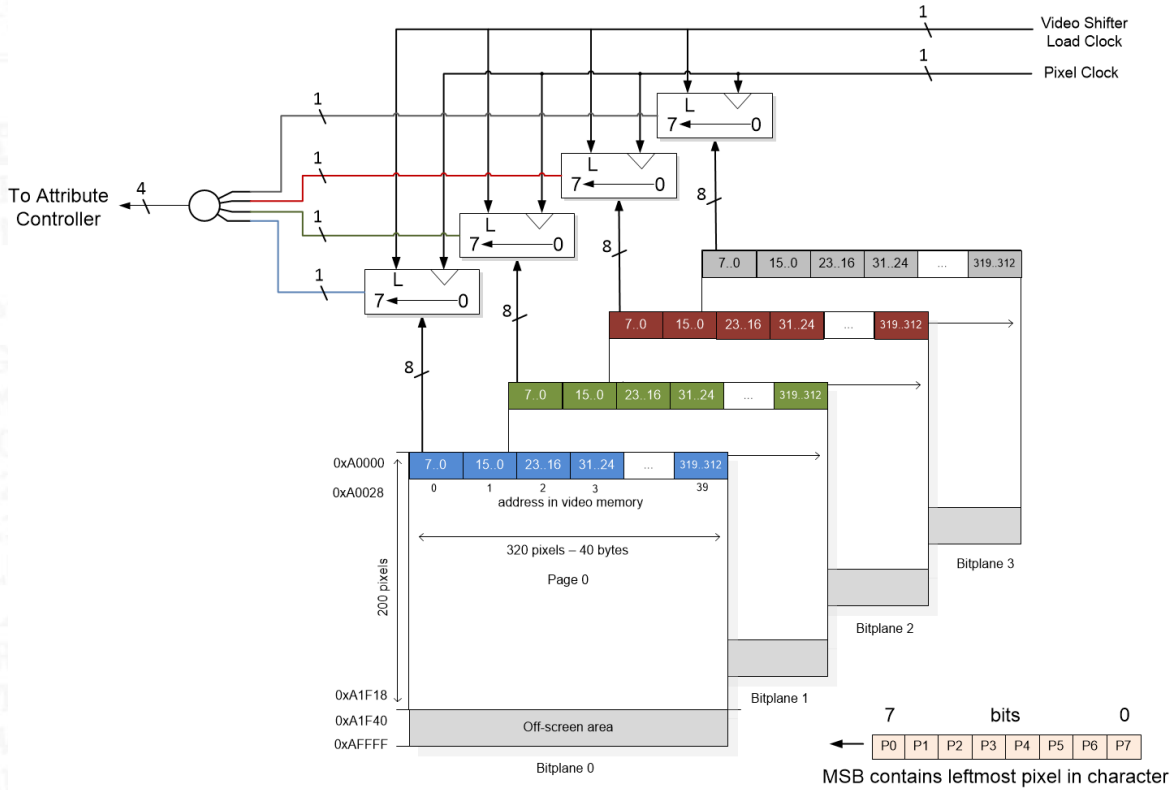
known from EGA days and 640x480 HiRes modes, being greatly enhanced in VGA:

- 16 sets of 16 colors, out of total 262144 colors
- split screen and smooth scrolling
- up to 8 video pages
- 200-lines mode double scanning improving graphics quality
- bitplane structure, ideally suited for flat-shaded 3D and translucency effects

lots of VGA demos and games take advantages of 16-color modes for fast and smooth action



in depth :: vga :: 16 color modes



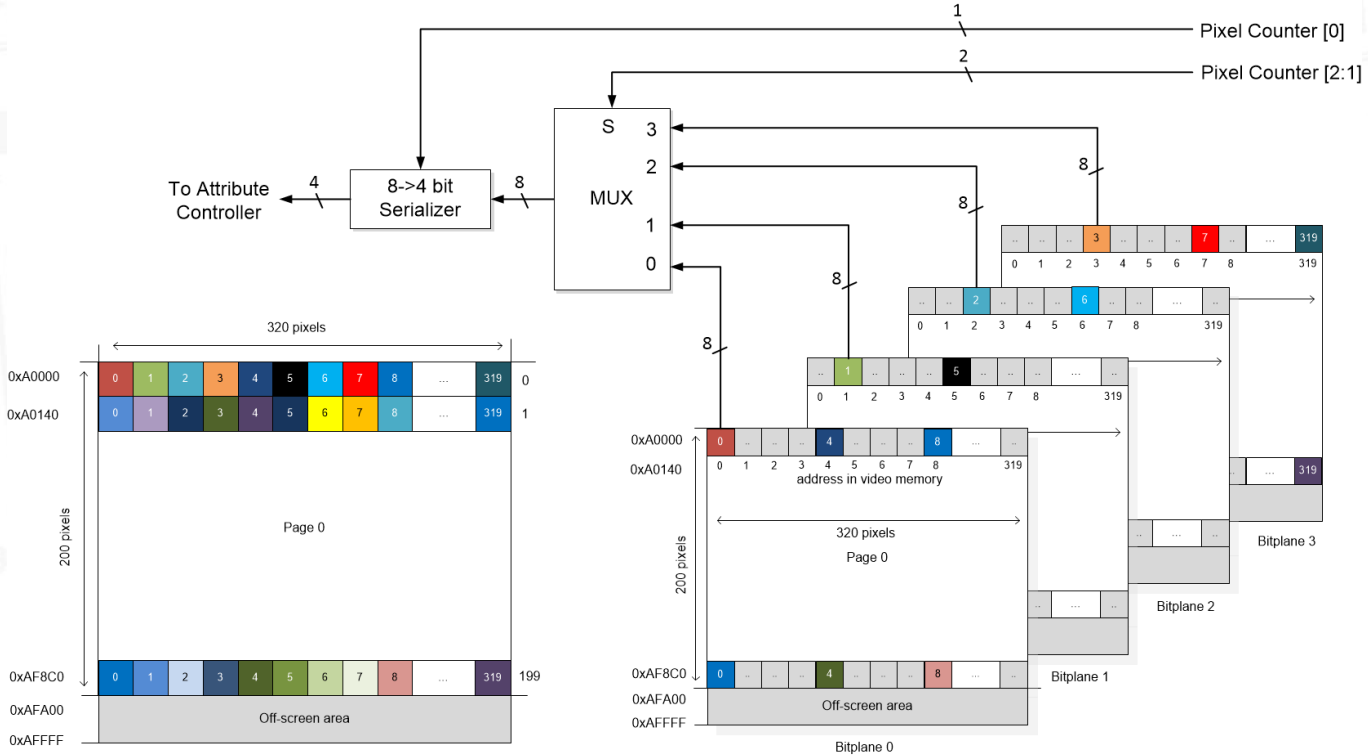
CRTC memory address (MA) is incremented by 1 every character (CRTC byte mode), Video Shifters loaded every Load Clock
For CPU access, bitplane is selected by Plane Write Mask/Read Plane Select registers

in depth :: vga :: 256 color mode

the most innovative addition to VGA, giving vivid colors and outperforming competing platforms

- linear memory layout (one byte – one pixel), ideal for graphics applications, faster than planar approach and giving green light to fast paced 3D action FPS
- wide palette range – 64 RGB levels, total 262144 colors palette
- cons: chained on top of EGA planar architecture, wasting ¾ of VRAM and permitting one page only :(

in depth :: vga :: 256 color mode

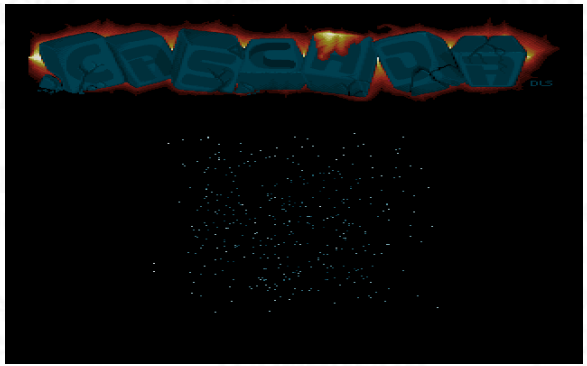


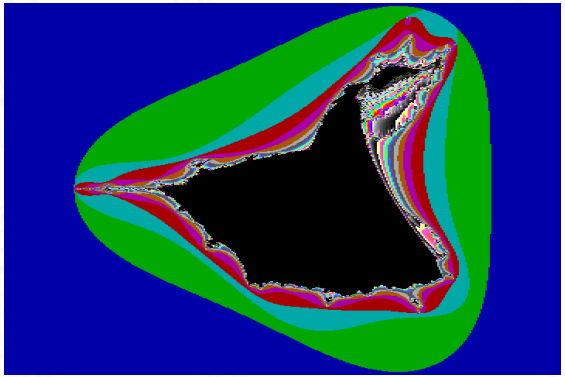
MA is incremented by 4 each character (CRTC double-word mode), Pixel Counter [2:1] select bitplane to read pixel data by CRTC
For CPU access, CPUA[1:0] selects bitplane for both read/write
Pixel Counter [0] selects nibble to be sent to attribute controller (0 – P[7:4], 1 – P[3:0]), actual 256-color pixel clock is thus divided by 2

vga :: 256 color mode :: mode-x

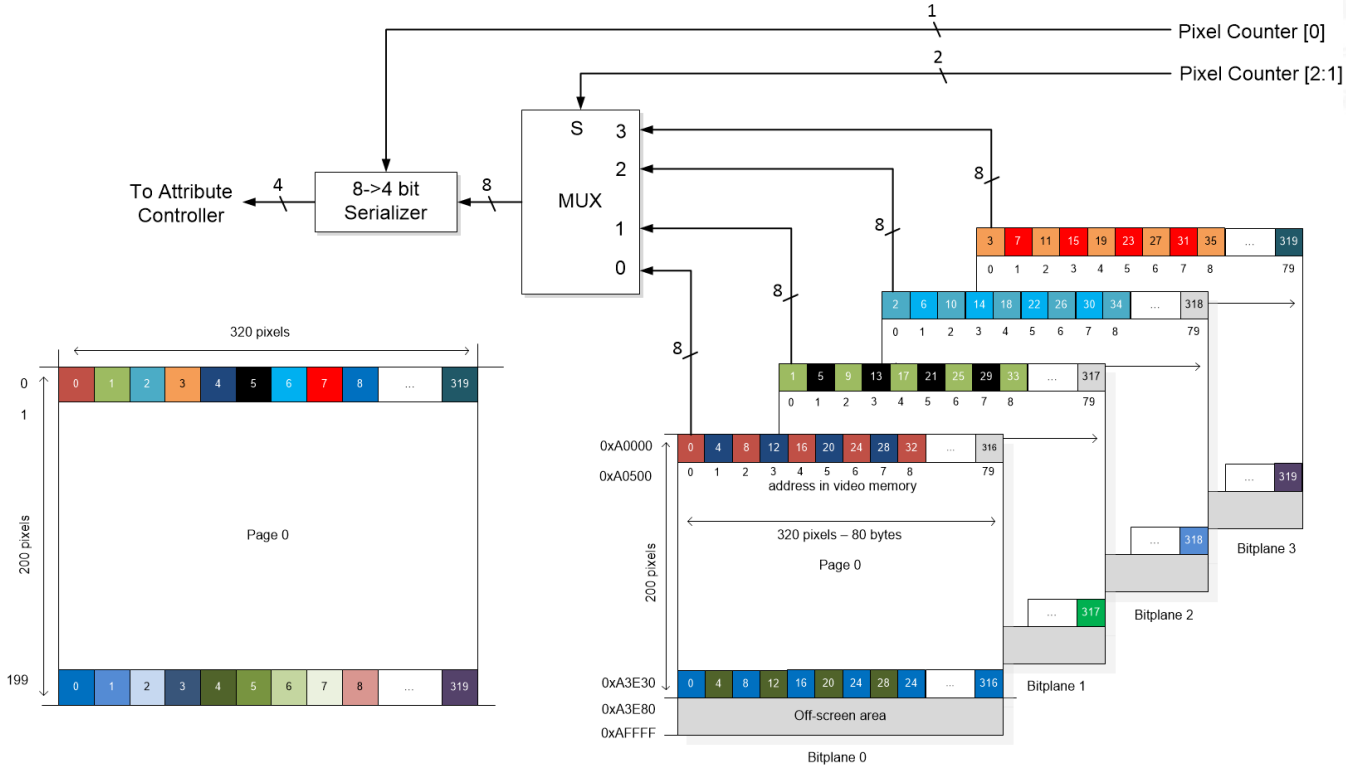
discovered by several people, popularized by Michael Abrash, the way of disabling plane chaining results in faster

- entire VRAM is useable, allowing 4 pages in 320x200 mode, smooth scrolling/panning, double\triple buffering and high resolutions up to 360x480
- Write Mode 1 (VRAM copy) can be used for fast sprite and image objects blitting
- somewhat ankward memory layout, still pretty fast for column filling
  - early 3D shooters like Wolfenstein 3D (1992) and DOOM (1993) render walls as textured columns of constant depth, which is equally fast in both Chain-4 Mode 0x13 and Mode-X, while the latter having advantage of multiple page buffering





vga :: 256 color mode :: mode-x



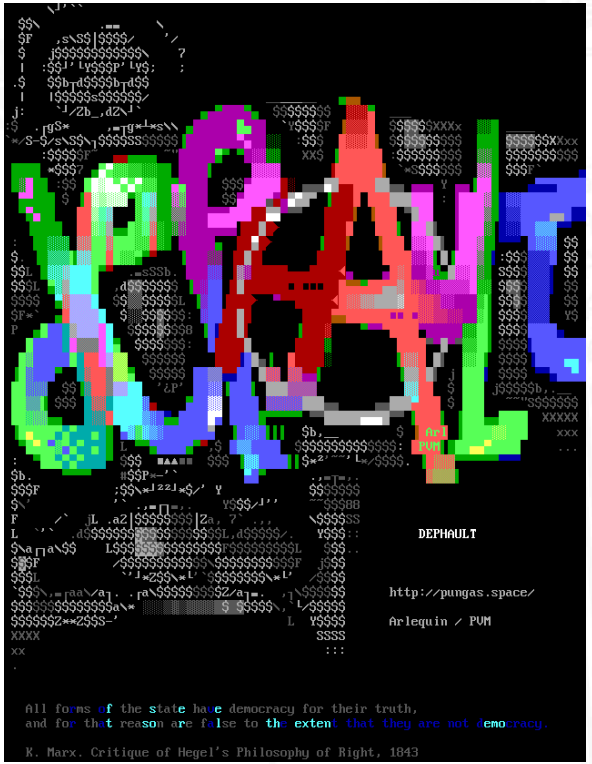
CRTC memory address (MA) is incremented by 1 every character (CRTC byte mode), Pixel Counter [2:1] select bitplane to read pixel data by CRTC
For CPU access, bitplane is selected by Plane Write Mask/Read Plane Select registers
Pixel Counter [0] selects nibble to be sent to attribute controller (0 – P[7:4], 1 – P[3:0]), actual 256-color pixel clock is thus divided by 2

in depth :: vga :: text mode

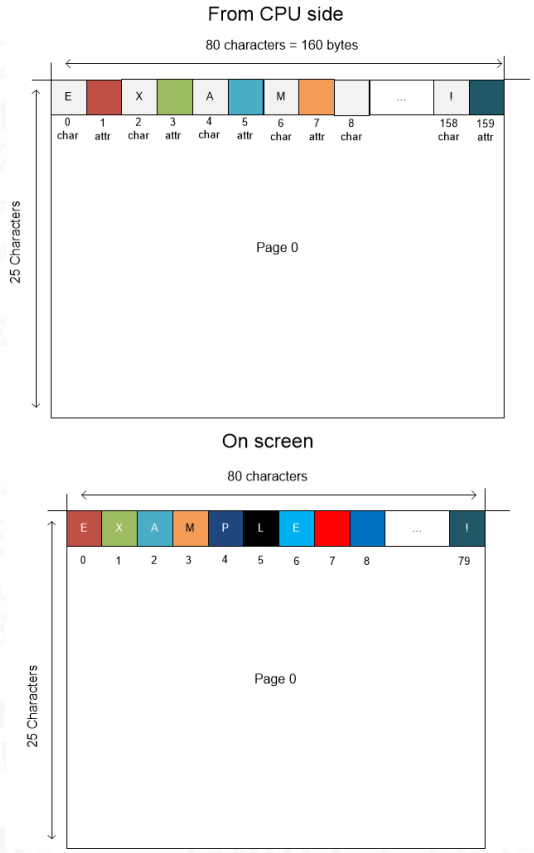
the good ol’ textmode got also heavily upgraded in EGA and VGA:

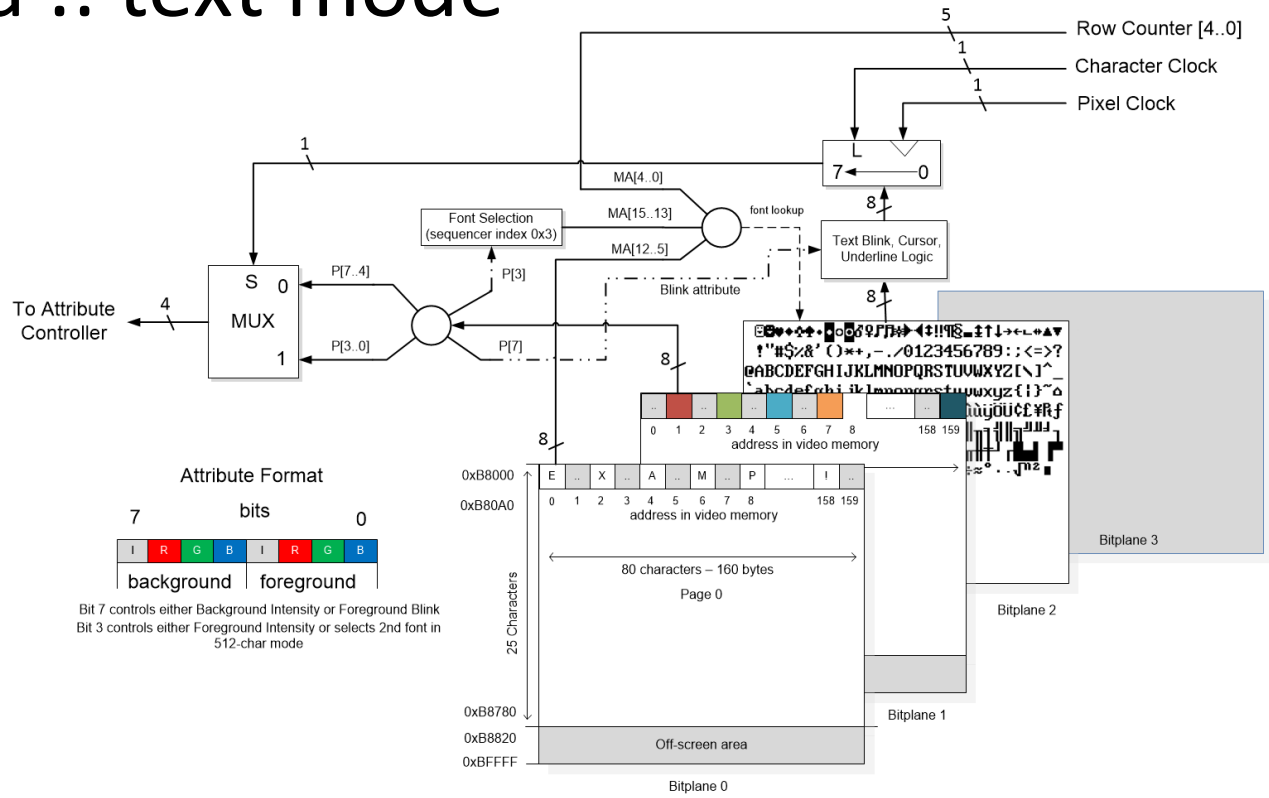
- up to 8 user-defined character fonts, 1 or 2 on screen
- blink, underline and hardware cursor effects
- smooth text scrolling and panning
- high character resolution – 9x16 font, 1-32 lines

enhanced capabilities were hugely praised by ASCII/ANSI scene



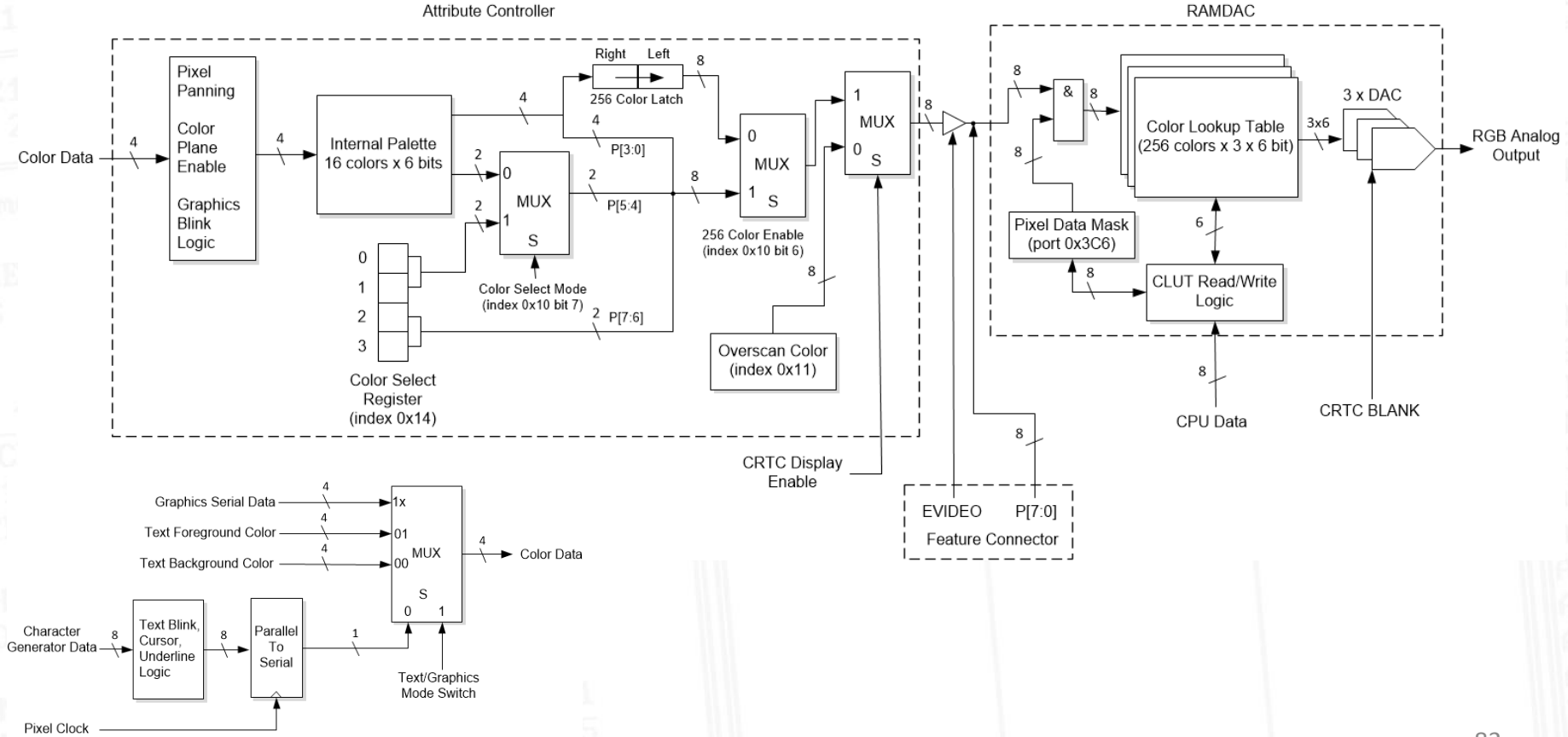
in depth :: vga :: text mode





MA is incremented by 2 each character (CRTC word mode).
2 bytes read from bitplanes 0 and 1, plane 0 used for font lookup from plane 2, plane 1 selects attribute
For CPU access, CPUA[0] selects bitplane 0/1 for both read/write

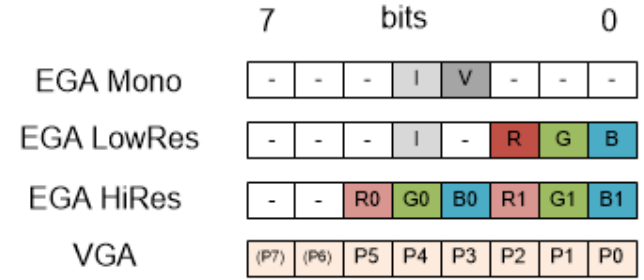
in depth :: vga :: color pipeline



in depth :: vga :: color pipeline :: colors

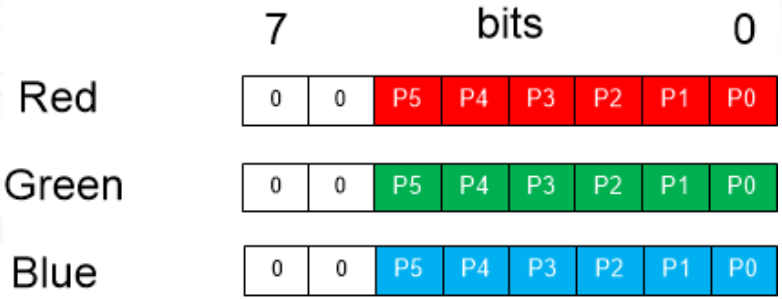
attribute controller palette colors

with Color Page registers, either 4 sets out of 64 colors or 16 sets out of 16 can be selected with one register write



note: for VGA, P[7:6] are available for Overscan color only, 0 for palette colors depending on ATC mode, either P[7:6] or P[7:4] are substituted by Color Page register content

RAMDAC palette colors

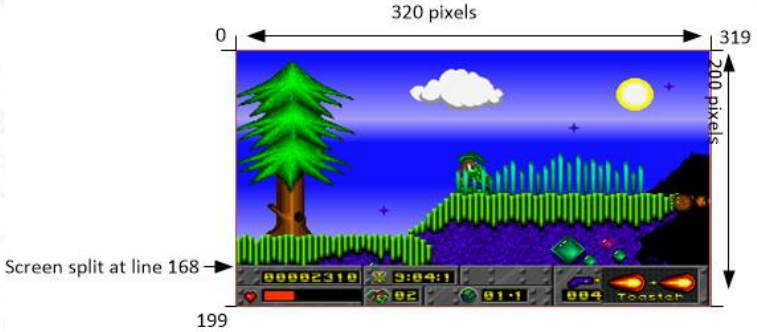


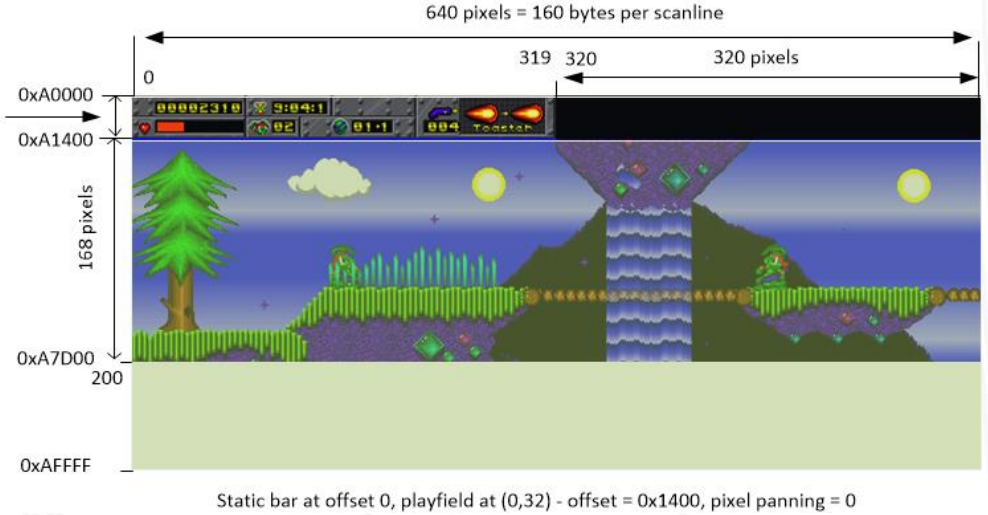
total 3x256 colors = 768 bytes, Red first then Green and Blue; palette index is autoincremented for batch color updates

in depth :: vga :: split screen

ability to scroll playfield independently of bottom bar

- static graphics is placed at offset 0
- CRTC Line Compare register sets to number of split line
- when split line is reached, display address counter resets to 0

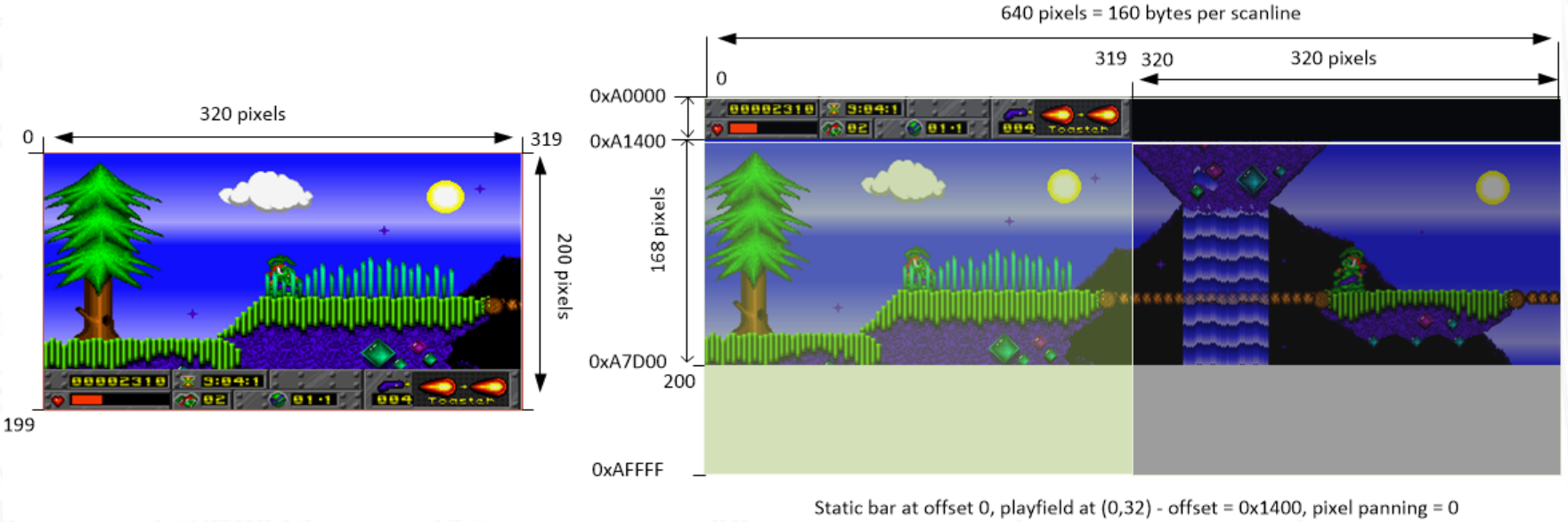




Static bar at offset 0, playfield at (0,32) - offset = 0x1400, pixel panning = 0

in depth :: vga :: scrolling

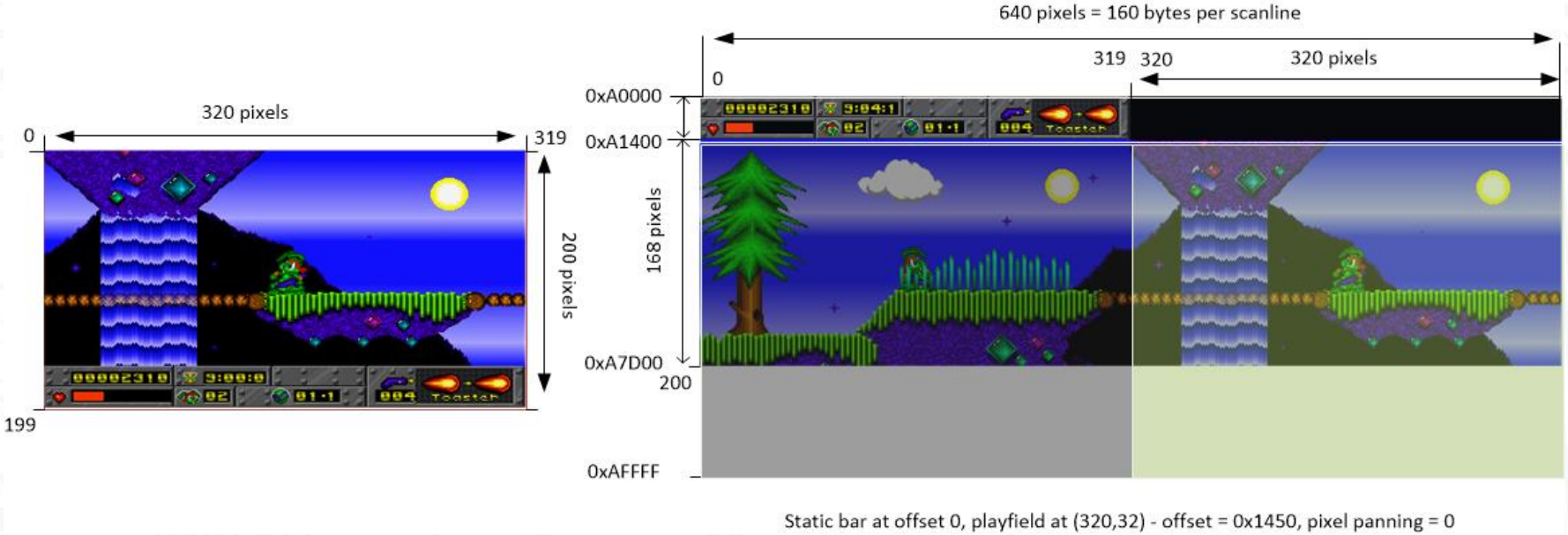
smooth scrolling\panning as combination of CRTC and attribute controller functions



Static bar at offset 0, playfield at (0,32) - offset = 0x1400, pixel panning = 0

in depth :: vga :: scrolling

smooth scrolling\panning as combination of CRTC and attribute controller functions



Static bar at offset 0, playfield at (320,32) - offset = 0x1450, pixel panning = 0

in depth :: vga :: scrolling :: precautions

unfortunately, scroll settings are distributed among 4 registers:

- Horizontal Pixel Panning (attribute controller index 0x13, pixel-precision H scroll)
- Preset Row Scan (CRTC index 0x8, pixel-precision textmode V scroll)
- Start Address (CRTC index 0xC/0xD) – start position in pixels or characters

moreover, CRTC and ATC registers are latched once and cannot be changed mid-frame

- CRTC registers are latched at leading edge of VSYNC pulse (port 0x3DA bit 3 0->1 transition)
- Horizontal Pixel Panning latched at start of display enable (port 0x3DA bit 0 1->0 transition)

and to make everything complicated further, some SVGAs (ATi, Matrox) latch scroll registers at different positions than most of VGA cards (check marvellous Gona's compatibility chart)

solution: update CRTC registers during active display, update ATC registers on VSYNC (port 0x3DA bit 3 0->1 transition)

side advantage: pseudo-triple buffering

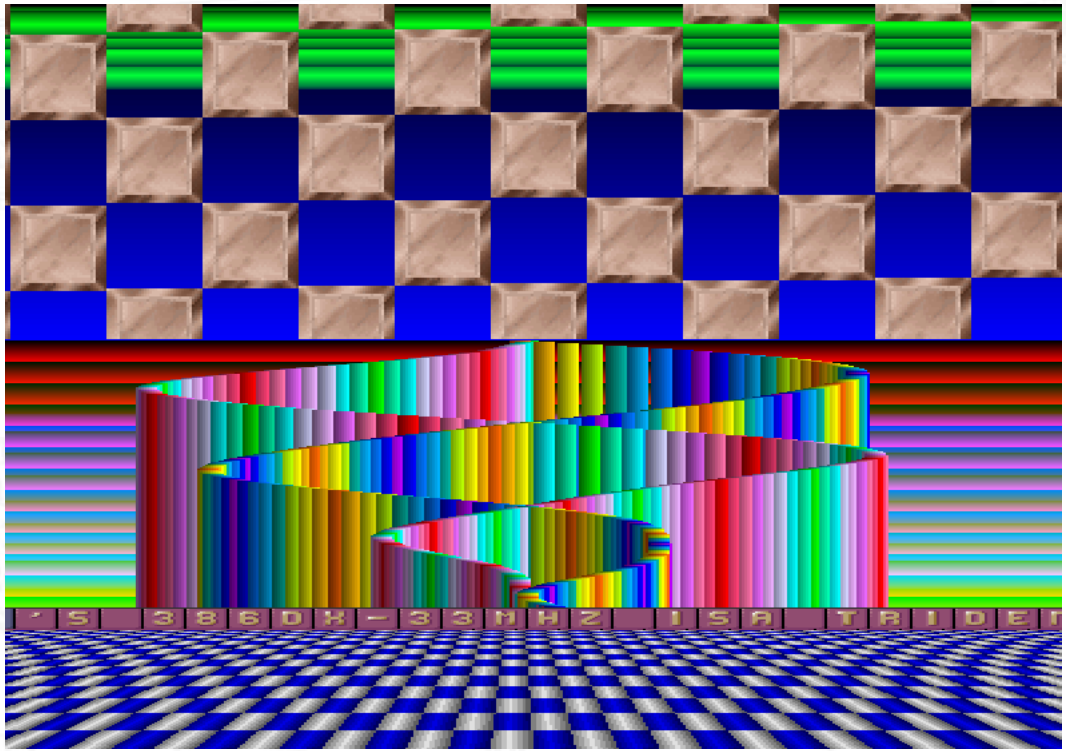
in depth :: vga :: black magic

even if changing start address midscreen is not possible, there are still lots of usable tricks

- polling for display enable (port 0x3DA bit 0) for tracking HSYNC and changing palette mid-line
  - COPPER BARS! AMIIIIGAAAAAAA!
- Setting Offset register (CRTC index 0x13) to 0 – displays same line on every scanline
  - kefrens bars, Y-zooms and even crude X-wobbles
- resetting split screen position multiple times per frame, changing modes mid-frame
- oddball graphics modes: 16 color linear, 256 color semi-planar, CGA chain-4, etc.
- simulating true color modes on standard VGA with color interlacing
- …and much much more :)

SVGATRIX by Type One/TFL-TDV serves well as reference

in depth :: vga :: black magic



Kukoo 2 by TFL-TDV (1994) – breaking every rule in the book

in depth :: vga :: aspect ratio and refresh rate

common VGA modes (except for 640x480) are 16:10 and runs at 70hz vertical while being displayed on 4:3 monitor, graphics is being distorted

70 Hz refresh considered as advantage during CRT era, but nowadays causes troubles with modern frame grabbers and displays

- with CRTC tweaking, 320x200 letterboxed 60hz modes can be achieved

in depth :: vga :: bios

VGA adapters typically contain ROM with BIOS firmware, providing video services via Int 0x10

- mode set with AH= 0
- pixel set/get (slow, but often used in sizecoding)
- palette loading functions (AH = 0x10)
- text output (notable functions are TTY output (AH=0E) and string write (AH=0x13))
- font manipulations (AH = 0x11): charset uploading, font pointer retireving
- graphics page, text scrolling and cursor control
- …and some more

some aspects of VGA programming are easier to interact via BIOS than direct programming, and vice versa

  - i.e. palette updates are done faster with direct IO port access

in depth :: vesa bios extensions

by the early 90s, VGA limitations became obvious, and Super VGA adapters were not quite compatible from software side

- i.e. 640x480 256 colors uses 300 kB of VRAM, requiring bank switching or linear frame buffer
  - every vendor used its own hardware method to select banks, and bank size/granularity differ from each other
  - moreover, even SVGA mode numbers were different between vendors!
- High/True Color modes (16/24/32 bits per pixel) are starting to gain attention, first appearing on professional accelerators, then making its way on mass market

while it wasn't such an issue for Windows applications or popular software packages like AutoCAD, because they implemented own driver model, independent from VGA BIOS API, DOS applications have to reimplement support for each VGA chip

as result, in 1989 Video Electronics Standard Association (VESA) developed a unified API for accessing SVGA features from bios and DOS applications, called VESA BIOS Extenstions (VBE)

in depth :: vesa bios extensions

key features:

- extending VGA BIOS Int10 calls with new API (AH = 0x4F)
- standard interface for graphics adapter information retrieving and display mode enumeration
- mode numbers are extended to 15 bit (+ VRAM clear request bit), with common modes being pre-assigned
- unified interface for bank switching, either via Int10 or direct procedure far call
- support for smooth scrolling, virtual screens and page flipping
- starting from VBE 2.0 (1994), Linear Frame Buffer support added, eliminating the need of bank switching for 32-bit protected mode applications
  - also, protected mode VBE interface was implemented, although not much used in software
- VBE 3.0 (1998) added refresh rate control and stereoscopic display support

unfortunately, not every graphics chip vendor did implemented VBE correctly, causing incompatibility issues (missing LFB support, no lowres modes, scrolling issues, etc.)

- SciTech Display Doctor (aka UniVBE) is alternative VBE driver for numerous SVGA cards, implementing full VBE specifications with additional features

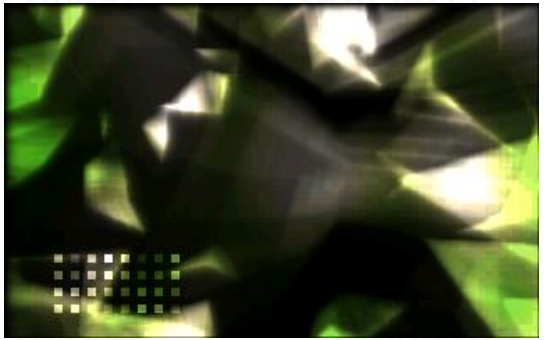
in depth :: vesa bios extensions :: in demos

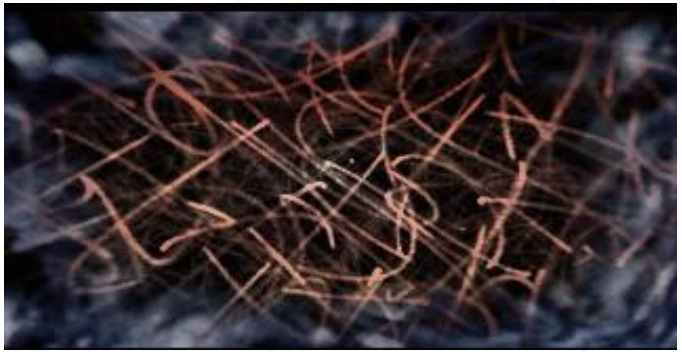
lots of late 90s demos use either one of these mode

- 640x480 256 colors, sometimes 16/32 bpp also
- low resolution Hi/True Color modes, like 320x240 16/32 bpp
- exotic 15 and 24 bpp modes are not uncommon
- same with oddball resolutions like 400x300 and 512x384 (half of 800x600 and 1024x768, respectively)

some productions don't respect screen pitch(aka logical width)

- resulting in squashed or skewed picture on some SVGAs



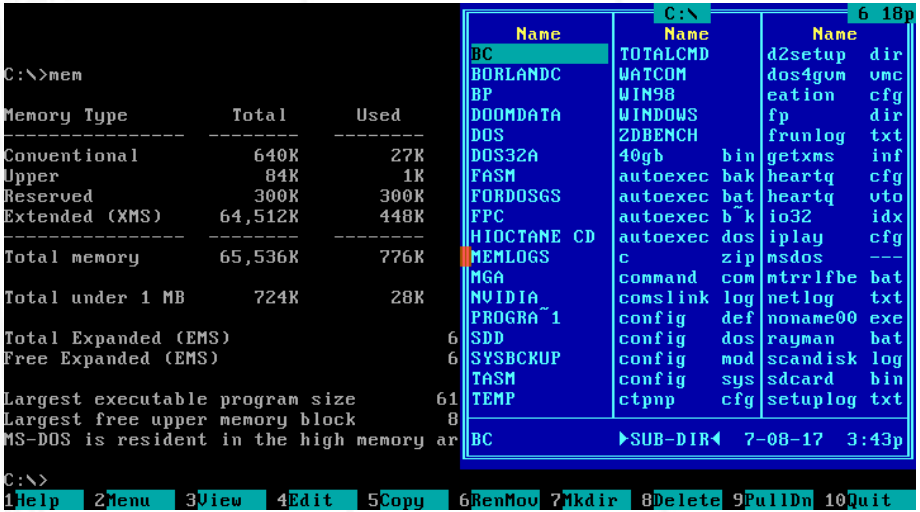


in depth :: dos

- single-tasking operating system
  - applications can hook interrupt vectors, then terminate with staying resident in memory, providing basic multitasking and system enhancing capabilities
- 16 bit real mode only, applications have to us special software APIs for extending capablilities

evolved during years, notable versions:

- MS-DOS 3.30 (1987) – well suitable for XT
- MS-DOS 5.0 – 6.22 (1991-1994) – used during 386/486 era



in depth :: dos :: memory interfaces

- eXtended Memory Specification (XMS):
  - allows to allocate, free and transfer data between extended and
  - applications can hook interrupt vectors, then terminate with staying resident in memory, providing basic multitasking and system enhancing capabilities
- Expnaded Memory Interface (EMS)
  - access to extended memory via page aperture and bank switching
  - available even for 8088-calss machines, practically emulated using memory management unit by memory managers such as EMM386
- DOS Protected Mode Interface (DPMI)
  - provides services for protected mode DOS applications , including mode swithcing, segment descriptors and memory management, calling real-mode procedures and interrupt, physical address mapping for memory mapped devices (e.g. LFB), etc.

development :: getting started

How to draw an owl

1.

1. Draw some circles

2.

2. Draw the rest of the [redacted]owl

development :: old toolchains

- QuickBASIC?
  - slow, and doesn't have any decent sound capabilities without calling external code libraries
  - besides, there are plenty of decent QBASIC demos (hey, DC5!)
- Borland Pascal?
  - you're probably loved (or hated) it back in the days
  - plenty of code examples and sound systems, all in one IDE
  - poor code optimization, no 386 target support -> DB $66 HELL
  - ending up making 90% of demo in assembly *shrug*
- Turbo Assembler?
  - while it is actually pretty decent choice…
  - however coding in 32-bit protected mode is tricky (fortunately PMODE can save you from cracking your head)
  - there are enough new and easier to use assemblers
- … as time flies, we can choose better tools…

development :: open watcom c\c++ ::

- the flagship DOS development tool, popularized by DOOM (1993) and used by almost every major DOS title since
- friendly 32-bit DOS environment – zero-based memory model (with 1st MB mapped 1:1 – no need to fiddle with segments), extended Int21 interfaces, comprehensive DPMI support, up to full 4GB
- cross-platform – targets 16-bit and 32-bit DOS, Win16, Win32, Linux, OS/2, Netware, upcoming x86-64 support;
  can cross-build DOS apps under Windows/Linux
- supported by lots of video/sound libraries
- decent code optimization – not the best by today's standards but still doesn't struggle like SDCC on Z80 :)
- language features support still near C++98/C99 levels (no C++11 sweetness, sorry)
- active development ceased after 1997, has been open-sourced in 2002 then being maintained by enthusiasts

development :: djgpp

djgpp

- a port of GCC toolchain for 32-bit DOS environment
- updates frequently (last build in 2020 featuring GCC 10.2)
- used by less software, Quake (1996) being the notable example
- Easy cross-compiling on Windows/Linux host, good 3rd party IDE/code editor support
- built-in GO32 DOS Extender- depends on DPMI host only
- less memory-friendly than Open Watcom – non-zero based memory model – more stable but requires to change segments to access DOS memory/VGA framebuffer
- great code optimization, latest C++17 features supported
- scarcer library support (Allegro is pretty redundant for demos)
- anyway a better (probably) than Watcom choice if you only want to blit frames and don't go deep into hardware trickery (i.e. targeting Pentium/VGA/VESA), unsuitable for XT/286

development :: assemblers

Flat Assembler (FASM)

- relaxed syntax compared to TASM, easy to use
- targets 16/32/64 bit code, supports all instructions from 8086 to AVX2
- outputs .COM/binaries, MZ/PE EXE, ELF and COFF object files
- open source, active community
- cross platform, runs under DOS, Windows and Linux
- well suited for sizecoding

Netwide Assembler (NASM)

- feature-wise comparable to FASM, has more complex and versatile macro facility
- in addition to FASM output formats, supports a.out, OMF, Mach-O and more
- also open source, runs on Windows, Linux, macOS, DOS and even KolibriOS
- overall a slightly better choice than FASM

development :: ide

depends on what you want to do

- for simple projects (like tiny intros), text editor like Notepad++ or Sublime Text + console is enough
- complex stuff often requires some kind of building system + comprehensive IDE
- Visual Studio Code is popular and versatile IDE choice
  - doesn't like Watcom C++ stuff much, needs some tweaks
    - Since Watcom's C++ dialect closely resembles Microsoft compilers, "msvc-x86" works pretty well
    - https://github.com/intbeam/watcom-vscode suits well as template
  - DJGPP and pure x86 asm is almost out of box
- makefiles are fine, CMake is also a decent choice for crossplatform projects
  - supports targeting both Watcom C++ and DJGPP

development :: debugging

for 16-bit apps, Turbo Debugger suits pretty well, if you really love to debug on real thing

Watcom C++ package comes with own versatile debugger (wd.exe)

- comes with both DOS and Windows (console and GUI wdw.exe) versions
- supports remote debug via serial link, TCP/IP, named pipes or Win 9x DOS session
  - serial null modem is most stable and useable option
- debugs both 16-bit and 32-bit DOS executables

for DJGPP, gdb is the number one choice

- also supports serial link debugging

of course, you have to compile your application with debug information to see anything other than pure asm :)

DOSBox and Bochs comes with their own built-in debuggers

- best suitable for debugging system-related code

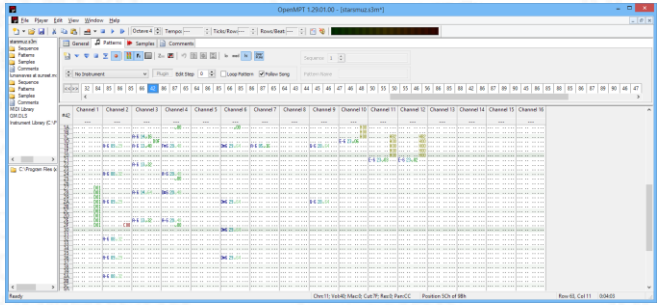
development :: graphics/music

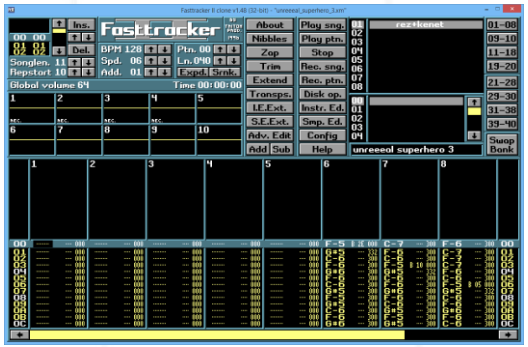
the choice of graphics tool depending on what type of graphics do you prefer

- pixelart: aseprtie, usenti, GrafX2, Multipaint, etc.
- illustrative gfx: better you know :)
- batch processing/conversion: ImageMagick, Image Alchemy

regarding the music, it mostly depends on your preferred sound device:

- PC Speaker: MONOTONE, PCSPE VSTi plugin, Vortex Tracker II
- OPL2/3: Adlib Tracker II, several native DOS OPL trackers OpenMPT, Scream Tracker 3
- modules: OpenMPT, MilkyTracker, FT2/PT2.3D Clone, native DOS trackers (ST3/FT2/IT2)
- streaming ADPCM/MP2/MP3: any DAW you like :)





development :: sound system

as already stated, this heavily depends on what you want to play

- PC Speaker: write your own routine, it's not that hard
  - tip: many PC games supporting PC speaker play raw periods dump
- OPL2/3: depends on target platform
  - most DOS OPL trackers come with player routine, although it's almost always 16-bit real mode
  - for 32-bit, you can abuse game MIDI libraries like AIL or HMI, but that would be totally overkill
  - as alternative, convert module to OPL register dump and play with custom routine, much like PC Speaker case
- streaming music (ADPCM/MP2/MP3):
  - unfortunately, didn't was a thing back in the days, and comparing to Amiga/Atari Falcon scene, no suitable streaming music players are available
  - theoretically MIDAS 1.1.2 can play ADPCM samples, albeit I didn't tested that yet
  - alternative 1: grab amp/mpg123/State of Mind (1998) demo sources, add SB output routine, done
  - alternative 2: cut your music into long samples, put into XM/IT, play with module player (see below)
  - extra tip for DJGPP users: use Allegro for playing in background (branch approves)

development :: sound system :: mods

tracked modules: the sweet spot of DOS scene :)

16 bit – forget about XM/IT

- MIDAS 0.40a is the best you can have, plays MOD/S3M, links with ASM/C/Pascal code, supports GUS and SB (unfortunately needs EMS), open source, doesn't work fine under Win9x
- DemoVT – resident player (can be used with BASIC demos!), MOD/S3M, GUS/SB
- MODOBJ and STMIK are too old, GUSPlay is simple to use but GUS-only, BWSB nags upon exit

32 bit – "are you still complain against 386? we're going after you!"

- Indoor Music System – Cubic Player' internal engine, plays MOD/S3M/XM/IT, supports GUS/SB/AWE32 native, Windows-compatible, a bit slippy to use, Watcom only
  - the MXMPlay fork is compact and fits great for 64k intros, XM/GUS only, PMODE/Watcom, Windows-friendly
- MIDAS 1.1.2 – the definite music system, plays MOD/S3M/XM/IT, supports GUS/SB, cross platform (Watcom/DJGPP/Win32/Linux)
- JUDAS Sound System – optimized mixer, plays MOD/S3M/XM/IT, supports GUS/SB, Watcom only, doesn't work under Windows
- Useless Module Player - plays MOD/S3M/XM/IT, supports GUS/SB/PC Speaker (yes!), PMODE/EOS/Watcom, also hates Windows
- RXMPlay – XM only, compact, GUS/SB/AWE32, PMODE/Watcom, Windows-friendly

emulators :: virtual machines

don’t use them. Not even VMware. Not even Oracle VM VirtualBox.

since VMs are focused to run newer host systems, they don’t bother to emulate common features being used by DOS high-performance stuff (cycle-accurate VGA emulation, VBE, etc)

- Bochs/QEMU is relatively decent, but please refrain from using them as first-class testing environment

emulators :: dosbox

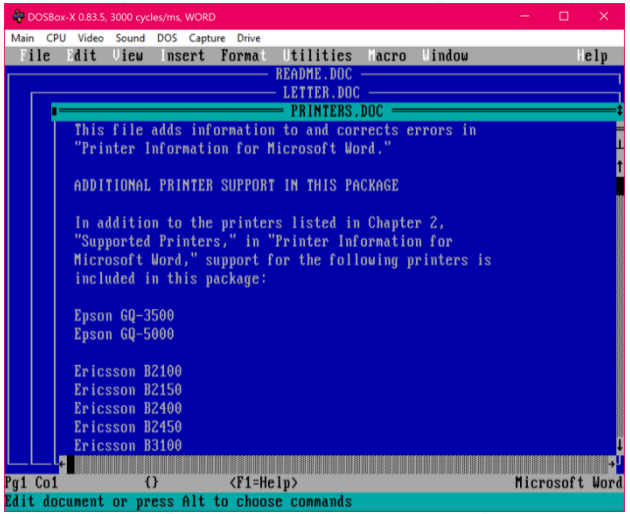
the easiest and most balanced way to test your DOS stuff :)

emulates DOS environment, can mount host folders as disks, floppy and CD images

vanilla DOSBox builds are games-oriented and feature-limited, doesn't cycle-accurately emulate VGA well, had trouble with tricky stuff, seldom updates

…fortunately, as it's open-source, better forks exist, i.e. DOSBox-X:

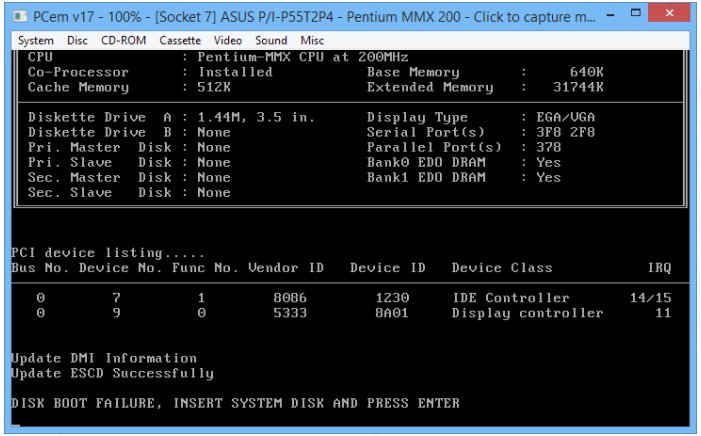
- emulates everything from PC/XT to Pentium Pro/MMX, and even PC-98!
- near perfect VGA emulation (still improving but runs 99% of demos great), great SB/GUS support, emulates SVGA (as S3 Trio64) and 3dfx Voodoo well
- can boot and run Windows 3.x/95/98 (if you really want it)
- works on Windows, Linux and macOS, and even able to run under real MS-DOS :)



emulators :: pcem

contrary to DOSBox and like other platforms emulators, emulates full PC environment

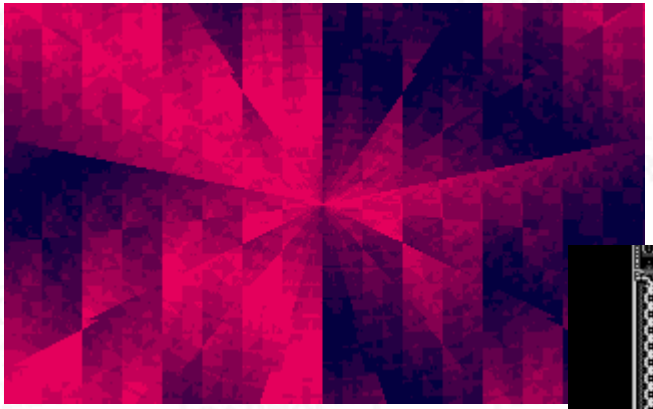
- cycle-exact device emulation (the 8088+GCA configuration runs 8088 MPH almost perfectly)
- emulates lots of devices (floppy and hard drives, CGA/EGA/VGA + several SVGA models, SB16/AWE32/GUS/Interwave, 3dfx Voodoo and much more)
- easy to adjust configuration – from 8088/4.77MHz to Pentium MMX/II
- uses real hardware ROMs (not HLE like DOSBox), you have to find them on the Internet and place in emulator ROM folder
- stopped active development in 2021 :(latest version released in Dec ‘20

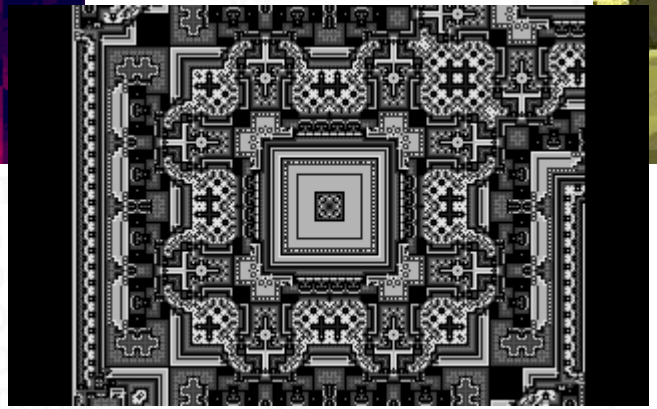


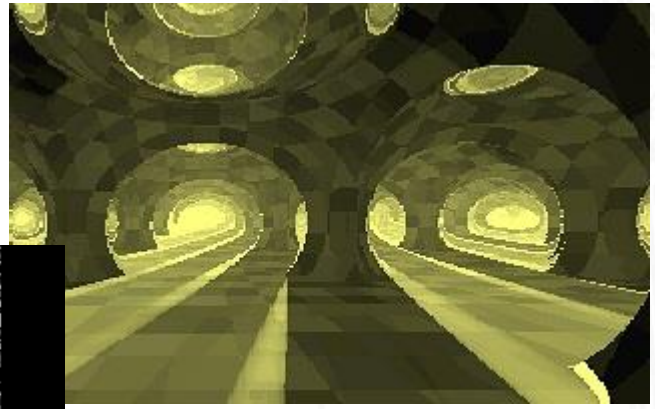
sizecoding

art of making audiovisual stuff in extremely tiny executables

(512 bytes or less)







sizecoding :: why dos?

usually, larger size intros (1k and more) are coded for Windows or Linux, taking advantage of advanced graphics capabilities

- in case of 512b or less, the EXE header alone eats about 50-100 bytes and initialization code takes as much as same, if not more!
- in contrast, DOS .COM files are just pure header-less executables, and graphics/sound initilaization takes only several dozens bytes
- cons: no fancy hardware 3D, 16-bit code by default

on the other hand, accessing i.e. VRAM or playing sound is easier and smaller code-wise than under Windows

- some intros even take advantage of high VBE resolutions and modern instruction sets like SSE4/AVX, and everything under DOS! :)
- sound is more troublesome, though
  - MIDI is easiest to use, following by Covox and PC Speaker

sizecoding :: tricks

- sizecoding.org provides lots of incredibly valuable info about tiny intro tips and tricks
- linear modes like 320x200 256 colors are the easiest and fastest to work with
- .COM files suits best for sizecoding, and you can even write your intro in a boot sectpr!
- during .COM startup, several registers are set to known state, so you can use default values to save some bytes
- enable listing output in your assembler and mark the hot spots with long opcodes your code, then prioritize them for size optimizing
- use full 16-bit register for simple instructions (dec, inc, add and so on), take a look on 1-byte instructions (movs/stos/xlat/etc.)
- use code as floating point constants – can really save a byte or two, if not more :)
- even default VGA palette can be useful, esp. if you're running out of free space

closing words :: tips & advices

- learn from others, don’t hesitate to look into code sources
- but at the same time, be creative! restrain from blatantly riping and copying effects, add something from yourself :)
- test your production on as much machines/platforms as you can
- do not lock out on specific hardware without a reason (like making a Pentium/VESA demo GUS-only)
  - yea, Gargaj, I’m talking about you :)
- provide a decent video capture, but try to forget about real HW enjoyers :)
- don’t be afraid of your first steps, create, release, learn and improve
  - everything is evolving iteratively

Q & A time :)

….and don’t forget to come to Combined 256b Intro Compo!...

…i made a little surprise for you, IBM PC lovers :)

…and other compos @ Demodulation as well :)

thanks for reading and watching!

seminar materials (with examples)

https://github.com/wbcbz7/yadm21_seminar

contacts:
wbcbz7.at(at)gmail.com
https://t.me/wbcbz7
discord: wbcbz7#3519