0.1 Keen Dreams in CGA

The first Commander Keen serie, *Commander Keen in Invasion of the Vorticons*, was only released for the EGA video card. Keen Dreams and later versions included a CGA version as well. The game play was exactly the same, sounds were the same, it was just that the graphics were CGA. Before diving into the source code, let's first get a better understanding of the CGA video hardware.

Trivia : It's an ironic twist that Softdisk did not use the original Keen's engine, as the code violated the company policy by depending on 16-color EGA hardware without supporting older 4-color CGA cards!

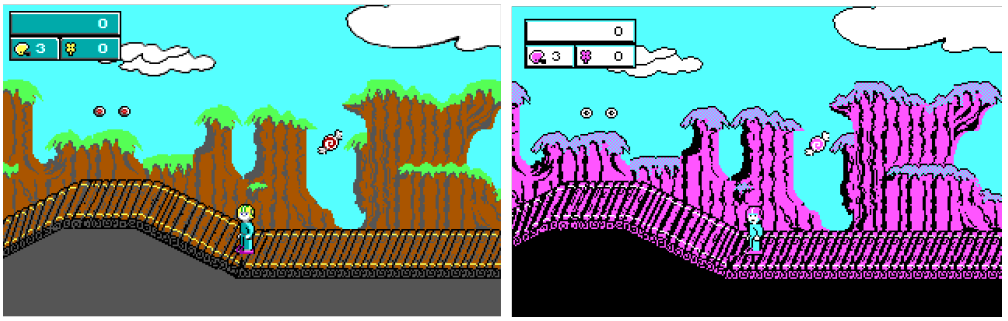

Figure 1: Keen Dreams EGA and CGA version.

0.1.1 CGA Video card

The Color Graphics Adapter (CGA), originally also called the Color/Graphics Adapter or IBM Color/Graphics Monitor Adapter, introduced in 1981, was IBM's first color graphics card for the IBM XT.

The CGA card can be summarized by the following hardware:

- It was built around the Motorola 6845 display controller.
- The framebuffer (the VRAM) contained two memory banks of 8KiB each, resulting in 16KiB total.
- Character generator ROM, containing a 9x14 font and two 8x8 fonts. This is the same ROM as used on the MDA video card.

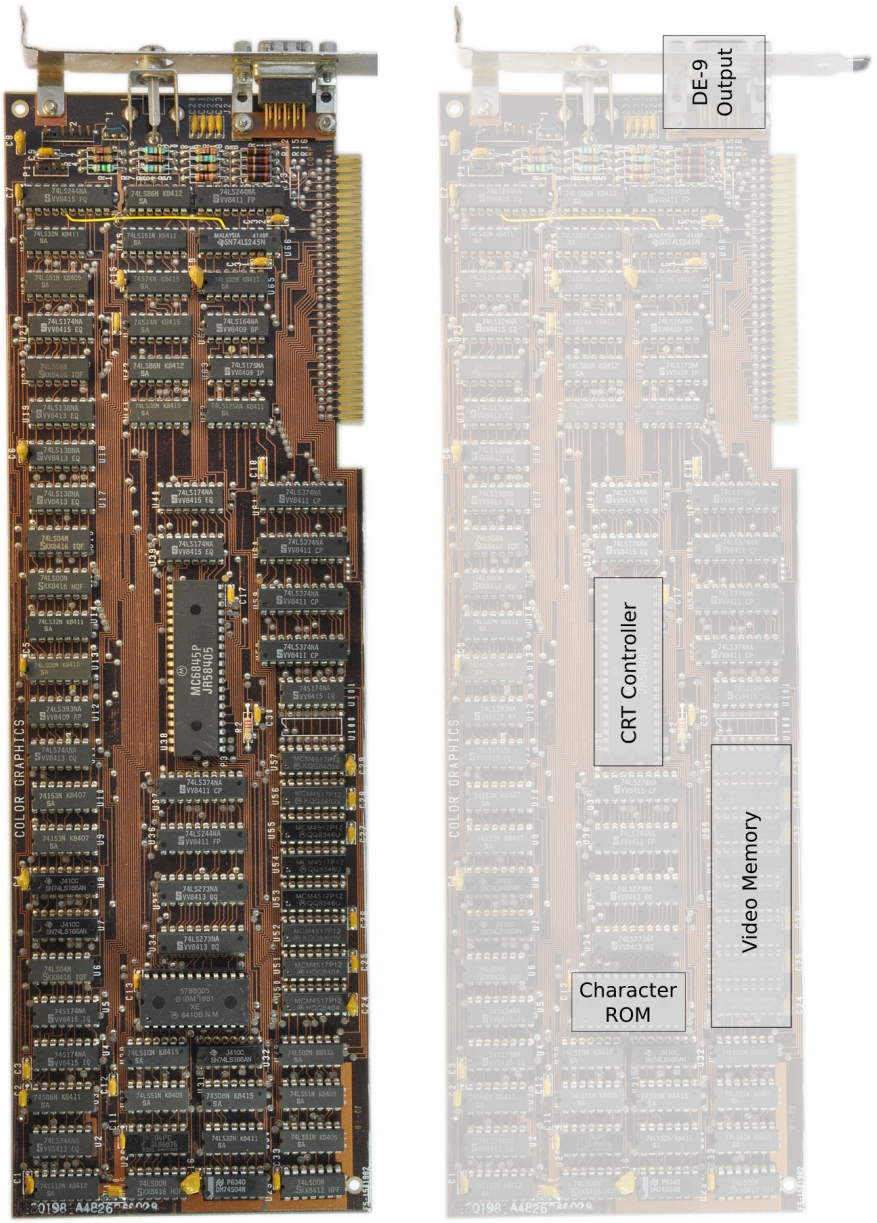

Figure 2: Original IBM CGA card

The CGA card has the following text and graphics modes:

Mode	Type	Format	Colors	RAM Mapping	Hz
0	text	40x25	16 (monochrome)	B8000h	60
1	text	40x25	16	B8000h	60
2	text	80x25	16 (monochrome)	B8000h	60
3	text	80x25	16	B8000h	60
4	CGA Graphics	320x200	4	B8000h	60
5	CGA Graphics	320x200	4 (monochrome)	B8000h	60
6	CGA Graphics	640x200	2	B8000h	60

Figure 3: CGA Modes available.

In graphics mode 4, which is used by Commander Keen, only four colors could be displayed at a time. These four colors could not be freely chosen from the 16 CGA colors, there were only two official palettes for this mode:

1. Magenta, cyan, white and background color (black by default).
2. Red, green, brown/yellow and background color (black by default).

The background color could be any of the 16 colors, but often it was kept black. For each mode there is a high- and low-intensity version of the palette.

Palette 1		Palette 2	
low intensity	**high intensity**	**low intensity**	**high intensity**
0 - Background	0 - Background	0 - Background	0 - Background
2 - Green	10 - Bright Green	3 - Cyan	11 - Bright Cyan
4 - Red	12 - Bright Red	5 - Magenta	13 - Bright Magenta
6 - Brown	14 - Yellow	7 - Bright Grey	15 - White

Figure 4: CGA color palettes.

The default palette when switching to Mode `04h` is palette 2 with high intensity, which is used by Commander Keen.

```
_AH = 0x0B    // Set color palette
_BH = 1       // 4-color palette mode
_BL = 1       // 0=palette 1, 1=palette 2
geninterrupt (0x10) // Generate Video BIOS interrupt

_AH = 0x0B    // Set color palette
_BH = 0       // brightness
_BL = 0x10    // 10h=high, 0h=low
geninterrupt (0x10) // Generate Video BIOS interrupt
```

0.1.2 Interlacing

In the early days of television and computer monitors, technology was far less advanced than it is today. Picture resolution was constrained by the limitations of the hardware, and engineers were constantly devising clever solutions to maximize performance. During the 1950s and 1960s, when CRT[1] monitors and TVs were standard, one major challenge engineers faced was how to produce clear images without overwhelming the hardware. The solution was a technique called interlacing.

Interlacing worked by splitting each video frame into two halves: the odd-numbered lines and the even-numbered lines. Instead of rendering the entire frame at once, the screen would first display all the odd lines and then, after a fraction of a second, display the even lines. This approach effectively halved the amount of data processed and displayed at any given time, which was a significant advantage given the limited processing power and bandwidth of the era.

The CGA memory layout in graphics mode utilized a similar interlaced architecture. VRAM was split into two banks: VRAM bank 0, which held even rows of pixels (0, 2, 4, etc.), and VRAM bank 1, which held odd rows of pixels (1, 3, 5, etc.). This layout required additional calculation steps for many CGA graphics operations if the programmer wanted to avoid visual artifacts during screen updates.

In CGA graphics, each pair of two bits represented a single pixel, allowing for a color value between 0 and 3, based on the CGA color palette. The two leftmost bits in a byte represented pixel 0, the next two bits represented pixel 1, and so on. Each byte in VRAM corresponded to four pixels on the screen.

[1] Cathode Ray Tube

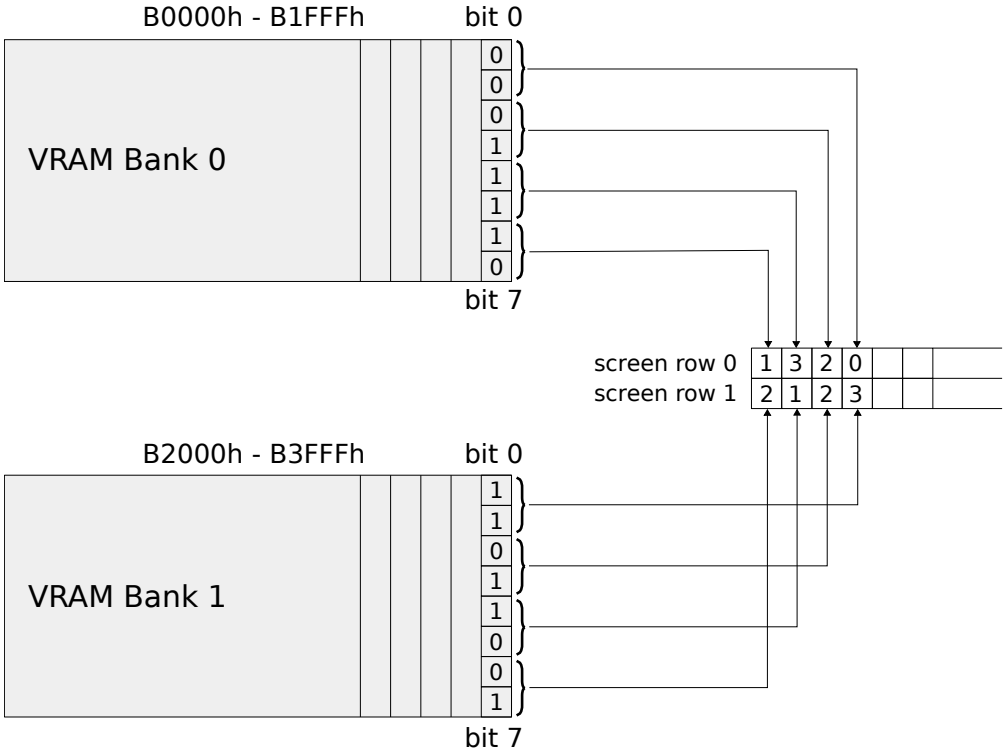

Figure 5: CGA interlaced memory.

Trivia : Interestingly, interlacing was never actually implemented in CGA monitors. When displaying VRAM to the screen, the CGA used a progressive (linear) scan, alternately reading from bank 0 and bank 1.

The CGA card employed memory mapping, similar to EGA. In Mode 4, VRAM bank 0 is mapped to memory addresses ranging from `B0000h` to `B1FFFh`, and VRAM bank 1 was mapped to `B2000h` to `B3FFFh`. Unlike EGA, the CGA memory model didn't require masking since the total 16 KiB of VRAM easily fit within a 64 KiB memory segment.

0.1.3 Double buffering

A full display screen in mode 4 requires 320 pixels x 2 bits per pixel x 200 lines, which equals 16,000 bytes of memory. Since the display screen consumes all 16 KiB of available VRAM, there is no capacity for additional screens. The only way to implement double buffering on a CGA card is by creating a 64 KiB buffer in RAM memory.

```
#if GRMODE == CGAGR
    grmode = CGAGR;

    // grab 64k for floating screen
    MM_GetPtr (&(memptr)screenseg,0x10000l);
#endif
```

This memory buffer contains both the video buffer page and a master page. The buffer page starts at offset 0000h, while the master page begins at 8000h. Both pages float within the 64 KiB memory segment, leveraging the same memory wrapping mechanism described in section "**??**" on page **??**.

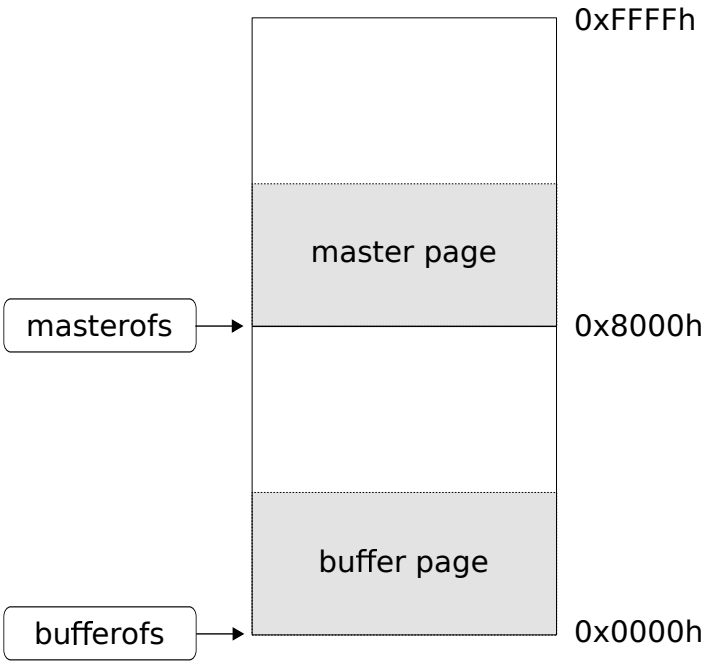

Figure 6: CGA double buffering memory layout.

0.1.4 Screen refresh

With double buffering in place, the same algorithm used for EGA can be applied. The final step of the algorithm involves updating the screen display by copying the buffer page to

VRAM and perform fine pixel adjustment. However, there are two complications with CGA.

The first complication is that the CGA card does not support pixel panning. As a result, the smoothest scrolling achievable is in increments of one byte. Since one byte represents four pixels, horizontal scrolling is limited to steps of four pixels, resulting in a more choppy scrolling experience compared to EGA.

```
void RFL_CalcOriginStuff (long x, long y)
{
    originxglobal = x;
    originyglobal = y;

    panx = (originxglobal>>G_P_SHIFT) & 15;
    pansx = panx & 12;  //pansx is 0, 4, 8 or 12 pixels
    pany = pansy = (originyglobal>>G_P_SHIFT) & 15;
    panadjust = pansx/4 + ylookup[pansy];
}
```

The second complication is copying the RAM buffer to the interlaced VRAM. This requires to split the linear memory buffer into copying all even rows to VRAM bank 0 and odd rows to VRAM bank 1.

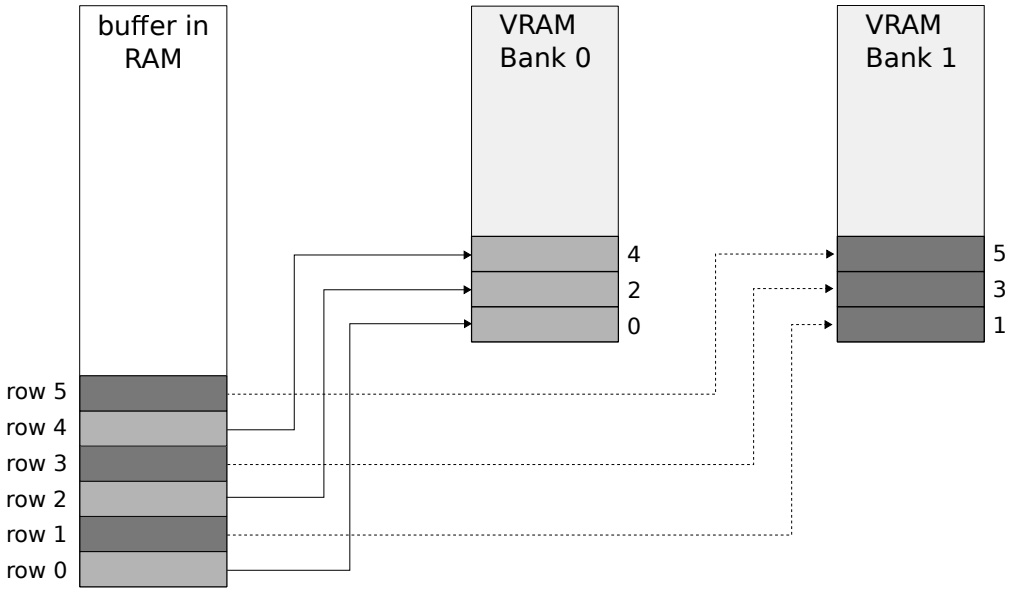

Figure 7: CGA memory to VRAM copy.

To avoid screen tearing, the system must wait for a vertical retrace, just as it does with EGA. However, the 286 CPU simply lacked the speed to copy all the necessary bytes from the RAM buffer to VRAM within the vertical retrace period. Consequently, in the CGA version of Commander Keen, screen tearing was unavoidable.

```
void VW_CGAFullUpdate (void)
{
    displayofs = bufferofs+panadjust;

    asm mov ax,0xb800
    asm mov es,ax

    asm mov si,[displayofs]
    asm xor di,di

    asm mov bx,100          // pairs of scan lines to copy
    asm mov dx,[linewidth]
    asm sub dx,80

    asm mov ds,[screenseg]  // buffer segment in memory
    asm test  si,1
    asm jz  evenblock

    [...]

    evenblock:
    asm mov ax,40           // words accross screen
    copytwolines:
    asm mov cx,ax
    asm rep movsw           // copy row to VRAM bank 0
    asm add si,dx
    asm add di,0x2000-80    // go to the interlaced bank 1
    asm mov cx,ax
    asm rep movsw           // copy row to VRAM bank 1
    asm add si,dx
    asm sub di,0x2000   // go to the non interlaced bank 0

    asm dec bx
    asm jnz copytwolines

    [...]
}
```