The original Commander Keen, Commander Keen in Invasion of the Vorticons, was only released for the EGA videocard. 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.

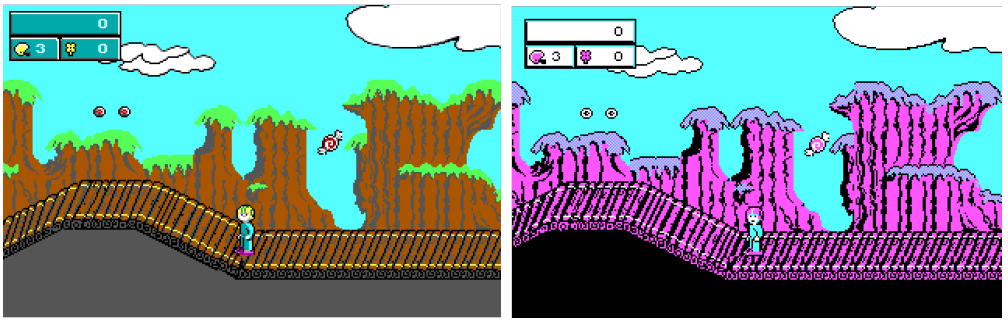


Figure 1: Keen Dreams EGA and CGA version.

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!

0.1 CGA Videocard

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 8 kilobytes each, resulting in 16 kilobytes total.
- Character generator ROM, containing a 14-row font and two 8x8 fonts. This is the same ROM as used on the MDA videocard.

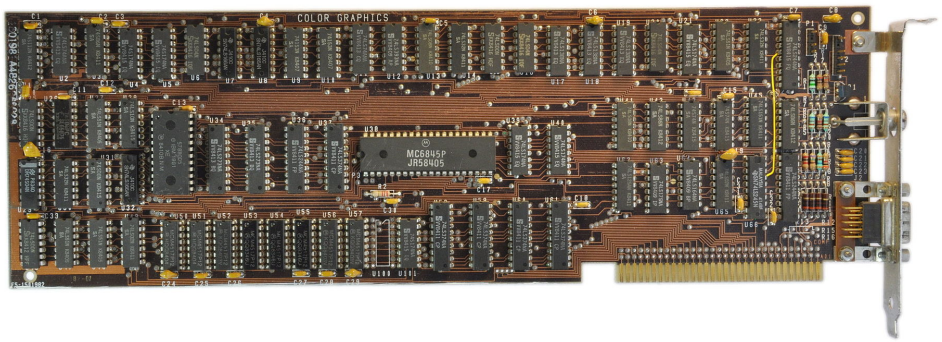


Figure 2: The CGA is a full-length 8-bit ISA card.

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: EGA Modes available.

In the graphics mode 4, which is used by Commander Keen, each pixel is using 2 bits for color, resulting in only four colors being 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 colour (black by default).
2. Red, green, brown/yellow and background colour (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.

0.2 Memory architecture and Interlacing

The CGA memory layout in graphics mode is different compared to EGA, as it is based on interlaced architecture. Normally, video memory is strictly linear: the next row of display data corresponds to the next row of pixels. But with CGA, the next row of display data corresponded to the row of pixels two rows down. This continued until the end of the screen and only with the second half of display data were the in-between rows addressed. So VRAM bank 0 was for even rows 0, 2, 4, etc., until the end of the screen and VRAM bank 1 was for odd rows 1, 3, 5, etc. This added calculation steps to most CGA graphics operations if the programmer wanted to avoid visual artifacts when updating the screen.

Each pair of 2 bits is one pixel with a color value of 0-3, referring to the CGA color palette. The 2 most left bits represent pixel 0, the next 2 bits pixel 1, etc. So each byte in VRAM represents 4 pixels on screen.

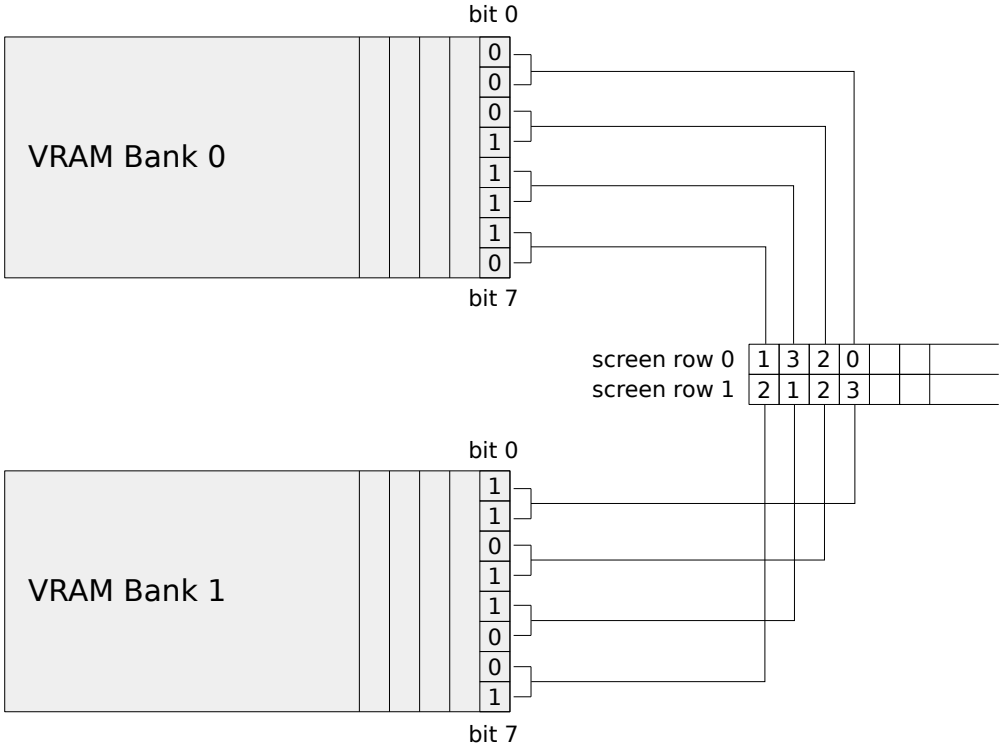


Figure 5: CGA interlaced memory.

The CGA card is making use of memory mapping, just like EGA. In mode 4, the VRAM bank 0 is mapped from 0xB0000 to 0xB1FFF and VRAM bank 1 is mapped from 0xB2000 to 0xB3FFF. Unlike EGA, the CGA memory model doesn't require masking as the total 16KiB VRAM fits easily in a 64KiB memory segment.

Trivia : Interesting enough, interlacing is never really implemented in CGA. When displaying the VRAM to screen it does a progressive (lineair) scan, where it alternately reads from bank 0 and bank 1.

0.3 Double buffering

A full picture in mode 4 requires 320 pixels * 2 bits per pixel * 200 lines = 16,000 bytes of memory. This means the display screen requires all 16KiB memory and there is no capacity in VRAM for extra screens. The only way to introduce double buffering on CGA is by creating a 64 KiB buffer in conventional memory.

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

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

The memory buffer contains both the buffer page and the static master page. The buffer page starts at offset `0x0000h` and the master page start at `0x8000h`. Both pages float around in the 64KiB memory segment, making use of the same memory wrapping as explained in section **??**.

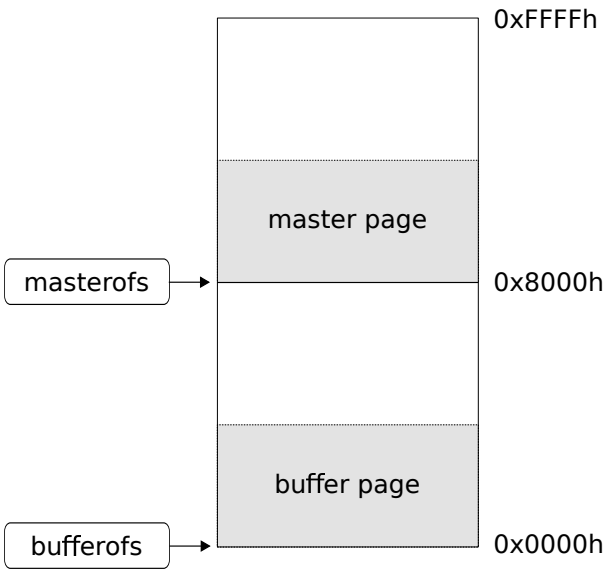


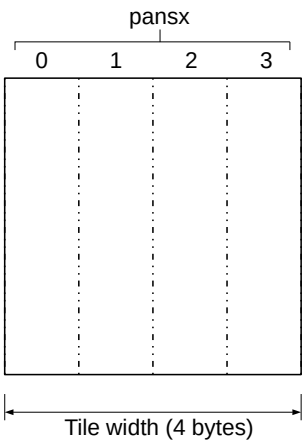
Figure 6: CGA double buffering memory layout.

0.4 Screen refresh

With the double buffering in place, the same algoritm as implemented for EGA can be used. The final step of the algoritm is updating the screen display by copying the buffer page to the VRAM. However, there are two complications with CGA.

The first complication is that the CGA card does not support pixel panning. So the smoothest pixel scroll is equal to scroll the screen with one byte. Since one byte represents 4 pixels,

it means scrolling to left or right is in steps of 4 pixels.




```
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 involves copying the RAM buffer to the interlaced VRAM. This requires to split the lineair memory buffer into copying all even rows to VRAM bank 0 and odd rows to VRAM bank 1.

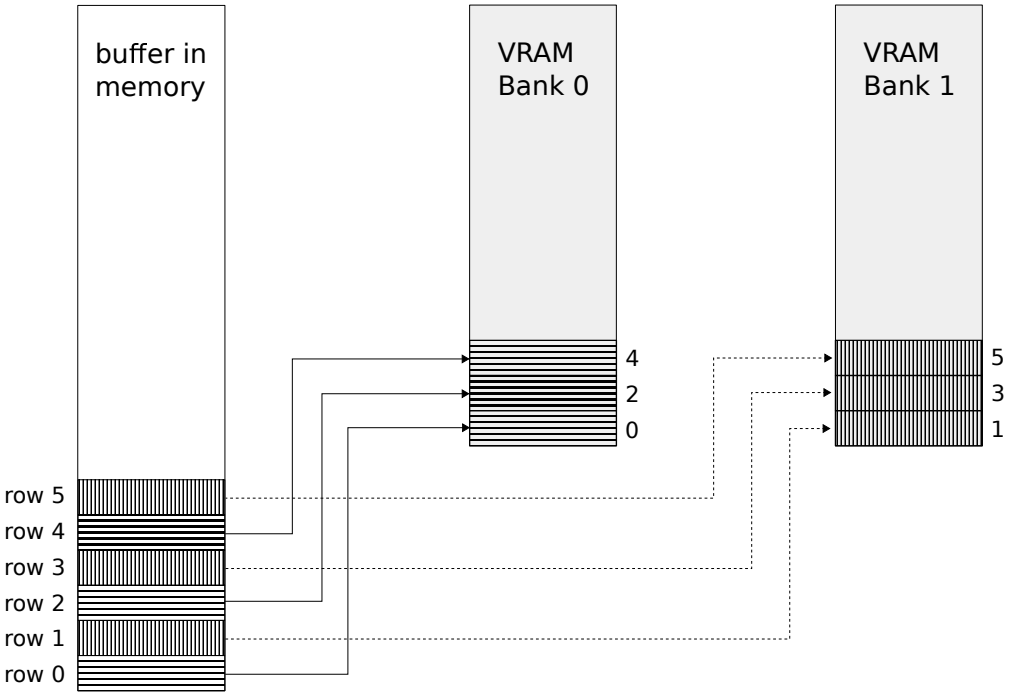


Figure 7: CGA memory to VRAM copy.

To avoid screen tearing the system should wait for a vertical retrace, like it was done with EGA. The problem is that computers weren't fast enough to copy all bytes from RAM buffer to VRAM during the vertical retrace period. So in the CGA version of Commander Keen, it was not possible to avoid screen tearing.

```
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

[...]

}
```

The original IBM CGA card could not handle writing and reading VRAM at the same time. Because video memory on the IBM CGA isn't dual-ported, when the CPU and the video card need access to the same byte of video RAM, the CPU wins; the card ends up reading a random value, causing "snow" on the screen. The only way to avoid the snow was,

also in this case, to wait for the vertical retrace. Most CGA clones resolved the issue by enabling read/write VRAM at the same time, but if one would play Commander Keen on the original IBM CGA card you experience both screen tearing and snow on the screen.