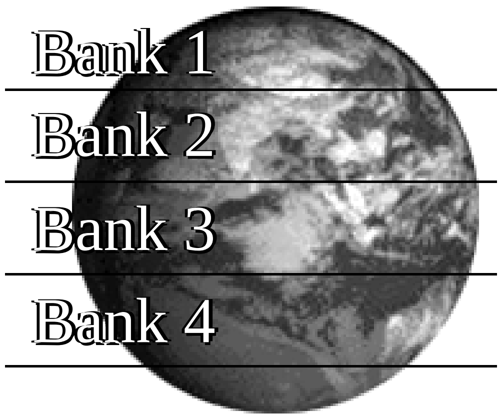
Using Video Modes With Banks

Intro

We've got a pretty good idea of how to write to VRAM using screen resolutions under 64k, but how do we do it with the higher screen resolutions? If we decided to just use the usual equation (x,y) = video_screen[screen_width*y+x]; it would totally crash our program using resolutions with screen sizes larger that 64k. The problem is that we only have a 64k window to access ALL of VRAM. Before you start crying in disbelief, let me tell you that there is still hope! Let's say our current video mode is 640x480x256. Divide that screen size by 64k and we'll get 4.6. That means we could fill our usual 64k space 4+ times! What the video card industry decided was to give us a means of bank switching. Realize that none of this tutorial is applicable if we have access to a mode that uses a Linear Frame Buffer. If that is the case, we use that instead and move memory normally!

Methodology

Using this strategy we can fill in the 1st screen, set our bank to 1, fill it again, set our bank to 2, fill etc. until it is full. So if our bank is set to 3, we can start writing at 0xA000 and instead of starting at (0,0) it will actually be part way down the screen.



One problem that does exist is the function that actually does that bank switching may greatly vary from card to card. The way to get around this is to access the VESA function for getting screen information and get a pointer to the function. Another alternative is to use the VESA BIOS functions for switching. If we decide to go this route, then we need to know what the screen granularity is. Basically, this tells us how large the chunks of memory are grouped together, sort of like the size of a memory page. This size isn't regulated by the VESA standard so it will vary from card to card as well. Here's some code that will get the job done, remember that we have to run the GetVESAModeInfo(..) function some time before utilizing this function, otherwise we won't know the specific granularity of the screen and it won't look correct.

Selecting the Bank

```
 void SetBankBIOS(short bank)
 {  bank<<=modeinfo.WinGranularity;
  __dpmi_regs r;
   r.x.ax = 0x4F05;
   r.x.bx=0x0000;
   r.x.dx =bank;
   __dpmi_int(0x10,&r);
 }
```

or in normal C++

```
 void SetBankBIOS(short bank)
 { bank<<=modeinfo.WinGranularity;
   union REGS regs;
   regs.x.ax=0x4F05;
   regs.x.bx=0x0000;
   regs.x.dx=bank;
   int86(0x10,&regs,&regs);
 }
```

Both functions are extremely simple. We are simply filling in the appropriate CPU registers using REGS to store them, then calling the VGA Interrupt (0x10)! Remember that we have to run GetVESAModeInfo() before attempting to change the bank because we need to know the correct granularity for the screen. There is a slight possibility that the program will dump, but the most likely event will be that the correct bank won't be switched and the screen will end up looking like garbage which is almost as bad!

VESA 2.0+

```
void (Video::*SetBank)(short);

void Video::SetBankAlt(short bank)
{  bank<<=modeinfo.WinGranularity;
   asm( “call *%0”
      :
      :”r” (pm_bank),”b” (0),”d”(bank)
      : “%eax”,”%ebx”,”%ecx”,”%edx”,”%esi”,”%edi”);
}
```

I also included a version of the new SetBank function using the Protected Mode pm_bank function. I've only found this information in one place. We have to use a special loadup scheme in order for our pm_bank function to work correctly. In our Video constructor we check to see if VESA is available and find the version number if it does. If it is VESA version 2.0+, then it will set our SetBank function pointer to the SetBankAlt routine. This doesn't involve the BIOS, making it a great alternative to our previous version!

Usually, you won't find modes that don't have Linear Frame Buffers unless the card is under VESA version 2.0. If it is, then you will find that these functions come in pretty handy! If you have any questions, comments, rude remarks please give me some feedback!

Contact Information

I just wanted to mention that everything here is copyrighted, feel free to distribute this document to anyone you want, just don't modify it! You can get a hold of me through my website or direct email. Please feel free to email me about anything. I can't guarantee that I'll be of ANY help, but I'll sure give it a try :-)

Email : Justin Deltener <deltener@mindtremors.com>
Webpage : http://www.inversereality.org

Inverse Reality