Plotting Pixels By Accessing VRAM

VRAM Organization

VRAM organization varies according to which video mode you are using at any given time. For the rest of this document, we are assuming that you are in 320x200x256 for screens under 64k or a similar linear memory mapped mode. We are also going to assume that we are talking about VRAM in graphics mode, NOT text. The reason for this is because of the ease of programming these types of modes. VRAM will start at A000:0000. From that location and beyond there is a 1:1 correspondence of memory locations to pixels on the screen. Each 8 bit value that is placed in the memory location corresponds to a color in the DAC color table at that pixel location.

For more information about the DAC color table and palette information please consult the palette manipulation tutorial.

Screen Size Under 64k

Since we have a 1:1 relationship between screen pixels and memory locations, we are able to plot pixels VERY easily. The equation is (x,y) = VRAMBase + (Screen_Width*y) + x. Or in our case here's some code:

```
void Plot(int x, int y, char color)
{unsigned char* vram = MK_FP(0xa000,0);
 vram[Screen_Width*y+x]=color;
}
```

This will get the pixel onto the screen as soon as it is called! The only problem with this is that it will immediately overwrite anything that is on the screen at that location, and we aren't using any offscreen buffer. For very simple applications, this will do just fine. If we wanted to use a dynamically allocated offscreen buffer all we would modify would be to set vram equal to the address of the buffer. For those fellas in protected mode, here's your version of the MakeFilePointer function!

```
void * MK_FP(unsigned short seg, unsigned short ofs)
 {if(!(_crt0_startup_flags & _CRT0_FLAG_NEARPTR))
   if(!__djgpp_nearptr_enable())
        return (void*)0;
   return (void *)(seg*16+ofs+__djgpp_conventional_base);
 }
```

High Resolution Screens

Remember that we can’t just start plotting pixels on the screen when it is bigger than 65536 bytes. We have to take Bank Switching into account. The only exception to this rule is if we have access to a Linear Frame Buffer. The VESA Video Modes tutorial takes care of explaining LFB. If this is the case, then the code for screens under 64k will work fine as long as Screen_Width is set correctly! Otherwise we have to use banking. If you don’t remember what this is all about, take a gander here. This should make you realize that we have a little bit more work ahead of us, nothing too major though! All we need to do is calculate which bank our pixel is going to be on and then calculate how far into it we need to go to plot the proper pixel. It should be something like this:

```
void Plot(int x, int y, char color)
{unsigned char* vram = MK_FP(0xa000,0);
 unsigned int BankNumber,BytesInto;

 BankNumber=(y*Screen_Width)/65536;
 BytesInto =(y*Screen_Width+x)-(BankNumber*65536);
 Bank(BankNumber);
 vram[BytesInto]=color;
}
```

Now several things have to happen for this function to work properly. First off we need Screen_Width to be set accordingly. If we use the functions we created previously for changing Video Modes, then this won’t pose any problem. If you are just going to cut and past this into your code, then make sure that Screen_Width is set correctly before attempting to use this code! We also need access to the Bank function. Remember in the Using Video Modes with Banks tutorial that we need to call the GetVESAModeInfo function before calling Bank(..). We need to do this to get the granularity of Video Memory in a VESA Mode. That should cover everything. I’ll admit that I’m writing this code from memory because I don’t think I’ll ever need a direct pixel plotting function in the future since I’m a pretty strict Offscreen Buffer man! If you have any questions, comments, rude remarks or if this darn code just doesn’t work for ya, please give me some feedback!

Contact Information

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 : deltener@mindtremors.com
Webpage : http://www.inversereality.org