Tweaked Video Modes

Intro

Your video card may be holding out on you! Tweaked video modes are the modifying of Video Controller registers to attain non-standard video modes. What i mean by “non standard” are those that aren’t available directly by accessing the Video Bios. Even the VESA VBE doesn’t even support these modes! The wonderful thing about TVM (Tweaked Video Modes) are that they are VERY predictable, offer extended capabilities, can be faster, and are available on any video card bought within the last decade! Each video mode requires a precise set of steps to make it work. You will also find that the means of getting to different video modes will very greatly. I will try to order these so that we start off with the easier ones 1st, and then graduate to higher levels of difficulty. One nice fact is that after we get these routines done, we can just call our function and bingo we will be in that video mode. That way we can forget *how* we got into that video mode and focus more on actually using it! Each section will cover one case that is located inside a switch() function inside our main modex video changing routine. Here’s what i mean:

```
void Video::ModeX(short Width, short Height)
 { if(Width == 320)
    { switch(Height)
         { case ...:
         }
      Screen_Width=320;
    }
  XCenter=Screen_Width>>1;
  YCenter=Screen_Height>>1;
  Move2Vid = &Video::MoveMemTweaked;
}
```

This function is going be called whenever tweaked video modes are passed to the main VideoMode(int) function. This way we can use one function for ALL video modes and just be concerned with the supporting functions to extend our functionality. This is also nice because it is easy to debug should something go wrong, and also a lot easier to read and understand. We are also going to be using 2 functions from 2 other tutorials. Don’t worry, they are really small and pretty darn easy to read, understanding, now that’s something different :) They are the StandardVideoMode() and the Unchain() functions.

Modes Covered

320x100
320x120
320x200
320x240
320x400
320x480

320x100 Unchained

```
case 100:
{ VideoMode(0x13);
  Unchain();
  asm(“ movw $0x3d4,%%dx
          movw $0x4309,%%ax
           out %%ax,%%dx”
           :
           :
         :”%ax”,”%dx”);
   Screen_Height=100;
   Screen_Size = 32000;
    break;
}
```

or if you prefer straight C

```
case 100:
{ VideoMode(0x13);
  Unchain();
  outp(0x3d4,0x4309);
  Screen_Height = 100;
  Screen_Size = 32000;
   break;
}
```

Here we are going into normal 320x200x256 mode, changing the video ram organization to unchained and then telling it that we want to display only 100 vertical lines instead of 200. We can easily do this by communicating with register 9 of the CRT. Bits 0-4 stand for the number of times to display the scanline minus 1. We can double our vertical resolution by doing an outp to port 0x3d4 with 0x4009. This is cool because when we go to tweaked modes like 320x240 we can use this technique to get 120 and 480 vertical resolutions. As you will find out this is exactly what we do! Notice how we are setting class variables Screen_Width, Screen_Height, and Screen_Size. That way, no matter what, we will always know what our screen dimensions are!

320x120 Unchained

```
case 120:
{ VideoMode(0x13);
    Unchain();
    asm(“movw $0x3c2,%%dx
            movb $0xe3,%%ax
            out %%ax,%%dx
            movw $0x3d4,%%dx
            movw $0x2c11,%%ax
```

```
            out %%ax,%%dx
            movw $0x0d06,%%ax
            out %%ax,%%dx”
            movw $0x3e07,%%ax
            out %%ax,%%dx
            movw $0xea10,%%ax
            out %%ax,%%dx”
            movw $0xac11,%%ax
            out %%ax,%%dx”
            movw $0xdf12,%%ax
            out %%ax,%%dx”
            movw $0xe715,%%ax
            out %%ax,%%dx”
            movw $0x0616,%%ax
            out %%ax,%%dx”
            movw $0x4309,%%ax
            out %%ax,%%dx”
           :
           :
          :”%ax”,”%dx”);
  Screen_Height = 120;
  Screen_Size = 38400;
  break;
}
```

or normal C++

```
case 120:
{ VideoMode(0x13);
    Unchain();
     outpb(0x3c2,0xe3);
      outpw(0x3d4,0x2c11);
      outpw(0x3d4,0x0d06);
      outpw(0x3d4,0x3e07);
      outpw(0x3d4,0xea10);
      outpw(0x3d4,0xac11);
      outpw(0x3d4,0xdf12);
      outpw(0x3d4,0xe715);
      outpw(0x3d4,0x0616);
      outpw(0x3d4,0x4309);
     Screen_Height=120;
     Screen_Size = 38400;
      break;
}
```

As I mentioned previously we are going into normal 320x200x256 mode, unchaining memory, and then changing the resolution to 320x240. Then we use our little technique that we learned above to go to 320x120!

320x200 Unchained

```
case 200:
{VideoMode(0x13);
  Unchain();
  Screen_Height = 200;
  Screen_Size = 64000;
  break;
}
```

All we are doing is changing to the standard 320x200 video mode and calling our Unchain function that turns the memory arrangement from normal Linear to Unchained! Remember that since we are changing the way video ram is organized, we have to modify the way we move our offscreen buffer to Vram and also how we directly draw pixels to the screen.

320x240 Unchained

```
case 240:
{VideoMode(0x13);
  Unchain();
   asm("movw $0x3c2,%%dx
          movb $0xe3,%%ax
          out %%ax,%%dx
          movw $0x3d4,%%dx
          movw $0x2c11,%%ax
          out %%ax,%%dx
          movw $0x0d06,%%ax
          out %%ax,%%dx
          movw $0x3e07,%%ax
          out %%ax,%%dx
          movw $0xea10,%%ax
          out %%ax,%%dx
          movw $0xac11,%%ax
          out %%ax,%%dx
          movw $0xdf12,%%ax
          out %%ax,%%dx
         movw $0xe715,%%ax
          out %%ax,%%dx
          movw $0x0616,%%ax
          out %%ax,%%dx"
          :
          :
         :"%ax","%dx");
   Screen_Height = 240;
   Screen _Size = 76800;
    break;
}
```

or normal C++

```
 case 240:
{VideoMode(0x13);
  Unchain();
  outpb(0x3c2,0xe3);            //Create Square Pixel Aspect Ratio
   outpw(0x3d4,0x2c11);         //Turn Off Write Protect
  outpw(0x3d4,0x0d06);          //Vertical Total
  outpw(0x3d4,0x3e07);          //Overflow Register
  outpw(0x3d4,0xea10);          //Vertical Retrace Start
  outpw(0x3d4,0xac11);          //Vertical Retrace End AND Write Protect
  outpw(0x3d4,0xdf12);          //Vertical Display Enable End
  outpw(0x3d4,0xe715);          //Start Vertical Blanking
  outpw(0x3d4,0x0616);          //End Vertical Blanking
  Screen_Height = 240;
  Screen_Size = 76800;
   break;
}
```

If you really want to know what's going on please read the remarks next to the C++ version of this mode. The nice thing is that we can forget what this is doing and just use it!

320x400 Unchained

```
 case 400:
{VideoMode(0x13);
   Unchain();
    asm(" movw $0x3d4,%%dx
            movw $0x4009,%%ax
            out %%ax,%%dx"
           :
           :
         :"%ax","%dx");
  Screen_Height = 400;
  Screen_Size = 128000;
  break;
}
```

or nomal C++

```
case 400:
{VideoMode(0x13);
  Unchain();
  outp(0x3d4,0x4009);
  Screen_Height = 400;
  Screen_Size = 128000;
 break;
}
```

This is pretty darn easy huh! All we are doing now is communicating with CRTC Register 9. Bits 0-4 tell it how many scan lines to use -1. This is the same thing we did to achieve 320x100 except we are writing a different value to the same port!

320x480 Unchained

```
case 480:
{ VideoMode(0x13);
  Unchain();
   asm("movw $0x3c2,%%dx
          movb $0xe3,%%ax
          out %%ax,%%dx
          movw $0x3d4,%%dx
          movw $0x2c11,%%ax
          out %%ax,%%dx
          movw $0x0d06,%%ax
          out %%ax,%%dx
          movw $0x3e07,%%ax
          out %%ax,%%dx
          movw $0xea10,%%ax
          out %%ax,%%dx
          movw $0xac11,%%ax
          out %%ax,%%dx
          movw $0xdf12,%%ax
          out %%ax,%%dx
          movw $0xe715,%%ax
           out %%ax,%%dx
           movw $0x0616,%%ax
            out %%ax,%%dx
           movw $0x4009,%%ax
           out %%ax,%%dx"
           :
           :
           :"%ax","%dx");
   Screen_Height = 480;
   Screen_Size = 153600;
   break;
}
```

or normal C++

```
case 480:
{VideoMode(0x13);
 Unchain();
 outpb(0x3c2,0xe3);
 outpw(0x3d4,0x2c11);
 outpw(0x3d4,0x0d06);
```

```
  outpw(0x3d4,0x3e07);
  outpw(0x3d4,0xea10);
  outpw(0x3d4,0xac11);
  outpw(0x3d4,0xdf12);
  outpw(0x3d4,0xe715);
  outpw(0x3d4,0x0616);
  outpw(0x3d4,0x4009);
  Screen_Height = 480;
  Screen_Size = 153600;
  break;
}
```

Here we are just going into normal 320x200x256 mode, unchaining memory, changing to 320x240 and using our little scheme we learned to achieve 320x100 to double our resolution to 320x480.

Well, there we have it, tweaked video modes! If you have any questions, comments, or some tips on how i can improve this page, please send 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