Scrolling and SEUCK file player

Introduction

This demonstration code started life as an example of how to use a multiplexor, music player and free directional scrolling. It was then expanded to demonstrate single directional scrolling using more than one frame to scroll the screen data, this resulting in lower per frame CPU usage. The single directional scrolling demonstration was then expanded to include sprite animation, sound effects and to show something like a full game loop.

The code for this project is located in git hub.

Building

To build the code use the following command from the C64\Scroller directory: ..\acme.exe ScrollEntry.a

Free directional scrolling

The ScrollEntry.a file contains various build options. Uncomment the line with Scroller_MultiDirection in it and also comment out the following Scroller_FullScreen and IRQOpenBorders options. Or run the pre-built Sroller.prg file. A joystick in port two scroll the screen.

Playing SEUCK data files

The ScrollEntry.a file contains various build options. Comment out the line with Scroller_MultiDirection in it and also uncomment the following Scroller_FullScreen and IRQOpenBorders options. Or run the pre-built DemoGame*.prg files. A joystick in port one or two will control the game. The SEUCK data file is included by the ScrollerData.a file so changing all the occurrences of “GAME3” to “GAME1” will include the other game example file. The SEUCK data files are in VICE file format which means they have $1a bytes of extra file header information, then there are 2 bytes of load address and the real memory start address is meant to be $900, hence each !bin command uses offsets starting with “$1a + 2 - $900”, the last offset is therefore a memory address for the particular part of SEUCK memory. Each part of the SEUCK data file format is extensively documented and demonstrated by their respective code routines.

Support for sideways SEUCK is enabled with the Scroller_IsHorizontal option in ScrollEntry.a and is demonstrated by the SideWays.prg file.

This SEUCK Redux code is completely written from scratch and in the public domain, so please feel free to use it, abuse it etc.

In ScrollEntry.a there is a define for Scroller_LOTD. Enabling this define will include the extra code for thr spell effects and linked enemy testing used in Legion of the Damned. The linked enemy test shows how to create end of level bosses and the spell effects show extra animated effects not seen the normal SEUCK.

This project also demonstrates using the sprite multiplexor to read VIC2 sprite collision information with the Multiplex_LogCollisions flag. This allows pixel perfect collision to be used at the expense of some raster time, which means sprite formations cannot be so closely packed on the screen as before. This greatly improves upon the sprite collision in the original SEUCK code. The original character based collision can be enabled by commenting out the Multiplex_LogCollisions line in ScrollEntry.a.

Quick recap about bug issues with REDUX and how to fix those

The phantom enemy sprite

Although this is an ongoing open source project, the SEUCK Redux source changes various parameters inside the Sideways Scrolling SEUCK engine. One issue, which is not the fault of SEUCK redux, is when you are playing the game, you will notice some unwanted enemies appear on screen. This could be the case where you may have tried to insert enemies from the top left or top border of the original SEUCK. In your original game, you cannot see the attack waves working, but it gets stored to memory. With SEUCK redux, you can see those appear on screen and it is impossible to delete that part of the attack wave. This is because you cannot see where your attack wave came from with original Sideways SEUCK. It is best to *avoid* using the top-left/top borders when inserting enemies into Sideways SEUCK for now, until that bug has been fixed. :)

Player movement speed issue (Very slow speed)

With SEUCK, if you set the player's speed to the slowest speed. There's a problem, where the player will get stuck if in redux, you move the player to it's maximum stopping position (Moving right). To fix this problem, search for .notLeftP in the PlayerControl.a source, then underneath bcc .notRightP, add beq .notRightP, and that problem will be fixed, until the next update. :)