# THE SEUCK TITLE SCREEN MAKER

**Version 1.8**

## 1: Introduction

If you enjoy developing games using the ***Shoot Em Up Construction Kit***, or the ***Sideways Scrolling SEUCK*** and would like to make new front ends, but cannot really code. The **SEUCK Title Screen Maker** will help. It allows you to load in your very own **custom font/charset**, **music**, **hires** or **multicolour logo bitmap** in **Koala paint** format, **edit the screen** with optional **animated chars**. There is even support for **including hi scores.** After you have finished making your title screen and you have saved it to disk (or D64). You can use a **freezer** cartridge or **.crt** plug-in such as an **Action Replay** or **Retro Replay** to load in your SEUCK game and **link** your new **front end** to the game and save it how you like. Alternatively, you can use VICE's own machine code monitor instead.

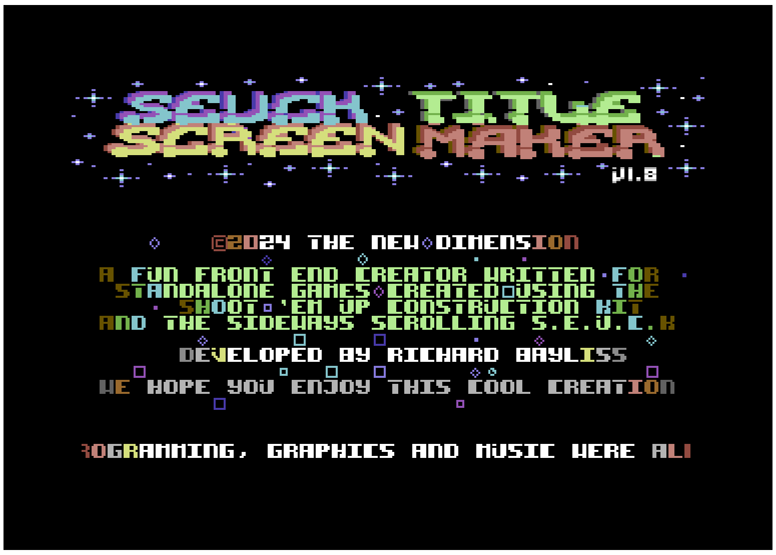
***Please take note that: This version of the Title Screen Maker is NOT backwards compatible with front ends made using the SEUCK Title Screen Maker V1.7+ or lower. This is due to the resizing of the title screen code and improvements implemented inside it. However it is possible to import text and colour data from previous versions of the title screen maker and re-load into V1.6 using an Action Replay Cartridge or Machine Code monitor.***

*NEW FOR V1.8:*

**BUG fix:** - Crash after end screen on standard SEUCK games now fixed.
**BUG fix: -** Fixed a minor display bug where the end screen did not use multi-colour characters from character set load option.
**BUG fix:** - Fixed a major display bug where the screen was garbled up if the game was aborted during play. The title screen can now return back to the correct charset, bank memory, etc.

***The parameter areas for in game enhancements are not changed therefore enhancements can be linked to the same addresses as featured in SEUCK Title Screen Maker V1.7.***

**This program uses standard KERNAL settings and should also work on SD2IEC devices.**



©2024 THE NEW DIMENSION

A FUN FRONT END CREATOR WRITTEN FOR STANDALONE GAMES CREATED USING THE SHOOT 'EM UP CONSTRUCTION KIT AND THE SIDEWAYS SCROLLING S.E.U.C.K

DEVELOPED BY RICHARD BAYLISS

WE HOPE YOU ENJOY THIS COOL CREATION

ROGRAMMING, GRAPHICS AND MUSIC WERE AL

***Example of the default front end that is featured in SEUCK Title Maker V1.8***

### What are the features in SEUCK Title Screen Maker V1.8?

1. You can make a front end for your SEUCK/Sideways scrolling SEUCK game by loading in some music (title or in game music or both). There is also an option for multi-tracks which can be assigned for **GET READY**, **GAME OVER** or **HI SCORE** tunes as well as **IN GAME MUSIC** should you wish to use them in your game.

2. You can load in either **a hi-resolution logo** in **OCP Art Studio** format, or perhaps load in a **multi-colour** bitmap logo in **Koala Paint** format. (The logo height has been raised by one full character row). So now instead of 7 x 40, the logo size is 8 x 40.
3. You can load in either a **hi-resolution charset** or **multi-colour charset**. There is also options to use c**harset animation**, should you wish to use that. *New for V1.6, it is now possible to load in **1x2**, (double height) charsets.*
4. You can include and also edit a **hi score table** for your game and **paint** the **flashing colours** for it (and the **sound options** icon, if enabled)
5. You can **edit** and **enable/disable GET READY** and **GAME OVER** text lines for your title screen
6. You can write 24 rows of **scroll text** for a message and pick a default **speed** for your **scrolling message** and **paint** the **colours** of it.
7. You can **select** a **range** of **options** that will suit your game. Also you can pick the **colour** of the in game **side border.**
8. You can **save** your **finished** title screen then **link** it to your SEUCK/Sideways SEUCK creation with aid of **Freezer Cartridge**, such as an **Action Replay, Retro Replay, Super Snapshot** or any other cartridge which has a built in machine code monitor that allows saving from lower memory to $fffa without messing any data/code.
9. You can now add **in game enhancements** like smart bomb/full boss explosion, power ups, custom score panel, level colour schemes, etc. (Code snippets have been provided on side 2 of the disk). You are only limited to three of the code snippets – but by following SEUCK School, you should hopefully work these out and be able to implement / combine more from my tips. More about this in the final chapter.
10. Multi-track music support (for those who want to assign specific tunes to their productions). Multi-track music must be *at $8000-$9FFF).*
11. You can make end screens of up to 11 lines for your games.

***Note that you cannot link a new front end to your SEUCK game, from the editor alone. You will need to save it separately, but once you read Chapter 9 your title screen will be all good to go.***

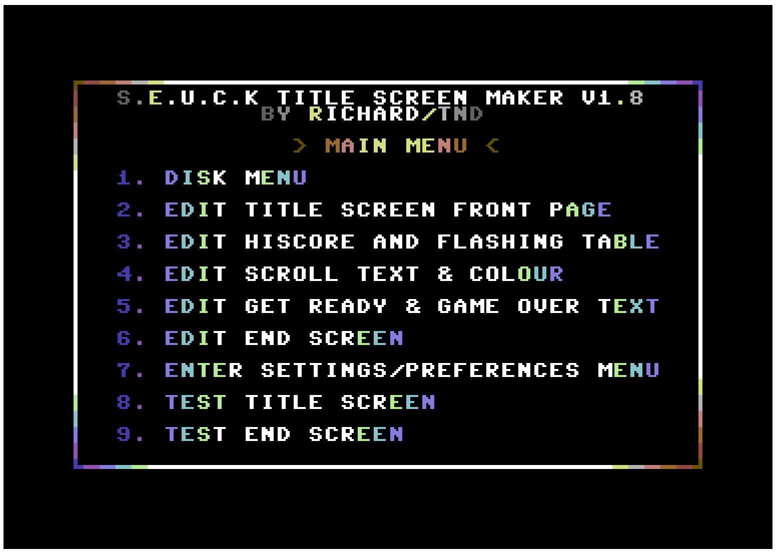
**So then, how does the title screen work?**

The SEUCK Title Screen Maker displays a **Koala Paint** multicolour bitmap (Dimension 8 rows x 40  characters) at the top of the screen. It can also display **credits** and a **scroll text** with optional **animated character sets/additional still hi-res characters**, which could be used for **borders around text**, etc. Also if enabled from the settings option. The SEUCK Title Screen Maker can **display** a **high score table** (after a few seconds) and then flip back to the **title screen credits**. There is an example title screen already built-in with the utility.

There is also a "**Get Ready**" and "**Game Over**" feature included in the enhancement. You can even set the title screen to play **music** and even play **jingles** for **Get Ready**, **Game Over** and possibly **Hi Score Table** listings. (These will be heard in your finished game, rather than the editor).

After the title screen is finished and saved. You can load the new front end manually to your finished SEUCK game creation with aid of an external **freezer** cartridge such as an Action Replay cartridge using either the fast load prompt or the **machine code monitor**. Alternatively, you can use **VICE** machine code monitor (make sure **bank ram** command is entered before loading and saving your new production). Then after finished, use either a standard C64 packer and cruncher or use Exomizer V3.1.2, Bongo Cruncher, TSCrunch, PuCrunch, ByteBoozer, SubSizer, LZMPi, Dali or any other cross-platform compression tool that is available. Use CSDB or any search engine to search for these cool crunchers. The command for compression differs to each one.

# 2: The main menu



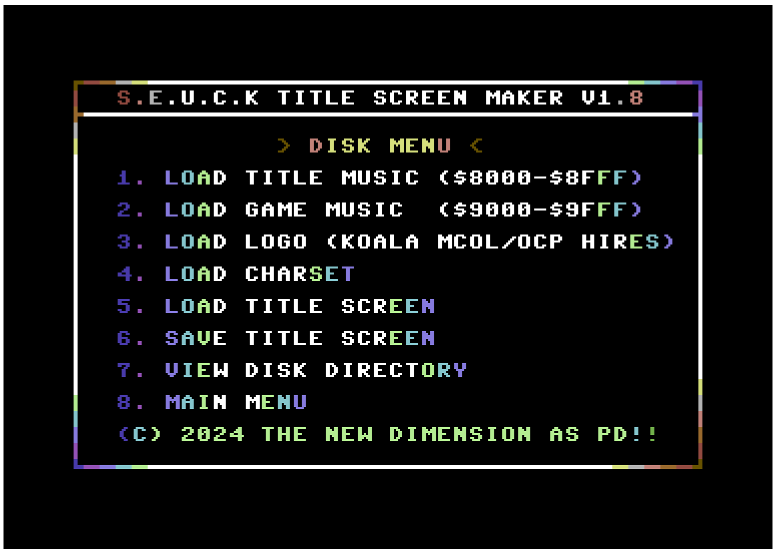
```
S.E.U.C.K TITLE SCREEN MAKER V1.8
         BY RICHARD/TND

          > MAIN MENU <

1. DISK MENU
2. EDIT TITLE SCREEN FRONT PAGE
3. EDIT HISCORE AND FLASHING TABLE
4. EDIT SCROLL TEXT & COLOUR
5. EDIT GET READY & GAME OVER TEXT
6. EDIT END SCREEN
7. ENTER SETTINGS/PREFERENCES MENU
8. TEST TITLE SCREEN
9. TEST END SCREEN
```

***Example: The main menu screen***

The menu has a set of 7 options which are as follows:

1. ***Disk menu****- Load and save specific data or display the disk directory*
2. ***Edit title screen –*** *Edit your credits/presentation page for the front end with animation or custom fonts*
3. ***Edit hi score table –*** *Edit the names, score and title name of the hall of fame. Also edit the flashing colour*
4. ***Edit scroll text –*** *Edit the text and colour of the scroll text presentation.*
5. ***Edit GET READY and GAME OVER text –*** *Write a single line message for both processes*
6. ***Enter settings/preferences menu –*** *Modify the settings of the title screen to match your game (Best done first before saving)*
7. ***Edit end screen -*** *See what it really looks like. If happy with the result save it.*
8. ***Invert charset –*** *Some fonts may have already been inverted after loading (IE Note Writer fonts)*
9. ***Exit program –*** *Aborts program.*

## 3: The Disk menu:



```
S.E.U.C.K TITLE SCREEN MAKER V1.8

         > DISK MENU <
1. LOAD TITLE MUSIC ($8000-$8FFF)
2. LOAD GAME MUSIC  ($9000-$9FFF)
3. LOAD LOGO (KOALA MCOL/OCP HIRES)
4. LOAD CHARSET
5. LOAD TITLE SCREEN
6. SAVE TITLE SCREEN
7. VIEW DISK DIRECTORY
8. MAIN MENU
(C) 2024 THE NEW DIMENSION AS PD!!
```

***Example: The disk menu options screen***

Like with with the main menu. The disk menu has a series of different options for loading saving and also viewing the disk directory. You can only use disk in SEUCK Title Screen Maker, but you can use the last disk drive you loaded the utility from. The device is automatically checked. Each menu option has been divided into sections, which are as follows:

1 ***Load Title Music*** *– Relocated music for your title screen at $8000 (Use this option to load multi-track music)*
2. ***Load In-Game Music*** *– Relocated music for your main game at $9000*
3. ***Load Logo*** *– (Koala Paint Multicolour / OCP Hires Colour)*
4. ***Load Charset*** *– (1x1, 1x2 or 2x1) – 1x2 is a double height charset*
5. ***Load Title Screen*** *– Load title screen finished/incomplete for further editing*
6. ***Save Title Screen*** *– Save title screen finished/incomplete for next session or linking to game*
7. ***View Directory*** *– Show what's on the disk.*
8. ***Main menu*** *– Return back to the main menu in chapter 2*

***If you have a problem loading in a filename, use * at the end of the correct name which you wish to use.***

## 4: Adding music to your production

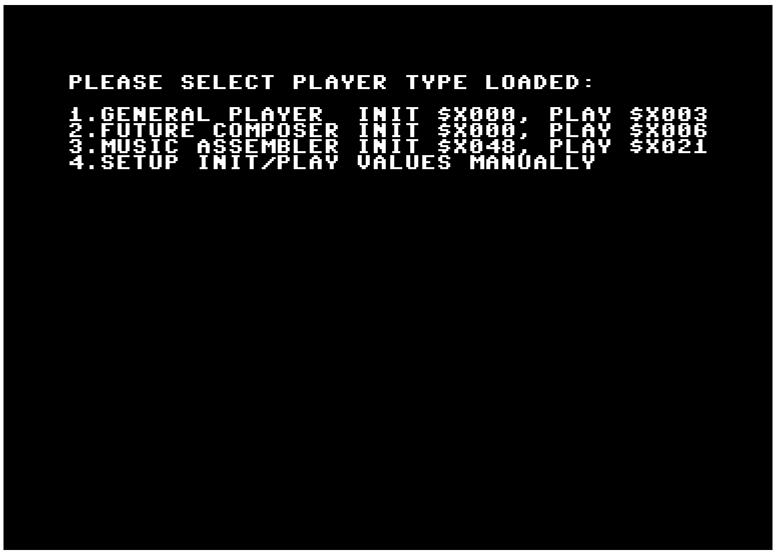
The SEUCK Title Screen Maker allows you to add music to your front end creations, and even to your game (if assigned from settings). Before you can add music to your production. It is very important that you ***relocate*** the music using a music data re-locator. If you or a group of friends have been composing music in either Goat Tracker V2, Cheese Cutter or SID Factory V2, there are some relocation/converters provided with the utilities. If however you wish to grab example tunes from HVSC, for example, my very own library. You will need to use Sidplay/W 2 by Inge to save the SID to PSID format, and make sure you download and use SIDReloc then use SidPlay again to save back in PRG / DAT format, and then import the PRG file into a D64 using the latest version of DirMaster by Style.

***Deflemask and some other players are not supported. It is also strongly advised that you don't take music from commercial games, without permission of the authors who composed the music.***

### 1. Load Title Music:

**WARNING:** *Typing a missing filename will zero-fill the music memory – try to avoid making mistakes.*

Type in the filename of the piece of music in which you wish to load into your title screen. Before you can load your music, you must have it relocated at ***$8000*** and it must not go over ***$8fff***, that is unless you just want to have one big title tune and no in-game music or assign a tune with **5 multi-tracks** from the settings menu.


```
PLEASE SELECT PLAYER TYPE LOADED:

1.GENERAL PLAYER  INIT $X000, PLAY $X003
2.FUTURE COMPOSER INIT $X000, PLAY $X006
3.MUSIC ASSEMBLER INIT $X048, PLAY $X021
4.SETUP INIT/PLAY VALUES MANUALLY
```

After loading, you will have a choice of 4 different music players. The supported players are as follows:

* **General music player (Init $x000, play $x003)**. These are mainly general music players, such as ***Goat Tracker***, ***Cheese Cutter***, ***Music Mixer v5***, ***JCH New Player***, ***Demo Music Creator (DMC)***, ***Voice Tracker***, ***SadoTracker***, ***EMS***

***V7.03, Virtuoso, SidWizard, Sid Duzz It*** etc. Basically, any tune that initialises at $1000, and plays at $1003 should be **relocated** to that address.

* **Future Composer (Init $x000, play $x006)**. Tunes composed in **Future Composer** or **Laxity's SID Factory** (V1 or V2) can be relocated to $8000 and played on the front end.

* **Music Assembler (Init $X048, play $X021)**. Tunes composed in Dutch USA Team's Music Assembler and some versions of **Voice Tracker** can be played on the front end although tunes composed using the Voice Tracker will work on the **General music player**.

* **Setup Init/Play parameters**. If you use a music editor or relocated a tune that does **not** support the **three of the above music players**. You can now edit the **custom music initialise and play parameters**. I had a go using this option using a couple of tracks that did not match the three specific players and they seem to work okay here.

***It is possible to use music up to 8K for the title screen. If you do use large tunes, please make sure you do not load in any in game music. If you do load in game music, the in game music will overlap the title music data and a chunk of it will be lost and might crash the title screen.***

After selecting the music player or editing the parameters. You should **hear** it play in the background. If you are happy with the tune, press SPACEBAR to go back to the disk menu. If however you selected the WRONG option, or a load error occurred. Then this program might crash, due to an insufficient jump location in the music file. The title music will also play inside the credits editor.

### 2. Load in Game Music

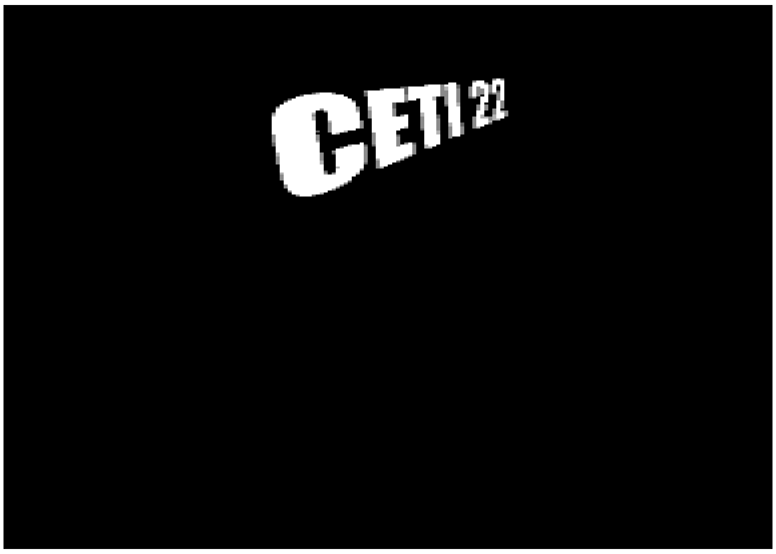
Loading in game music is a similar process as loading the front end music. However it does not support additional tunes/jingles for in game music. Music memory is $9000 -$A000 and cannot go over $1000 bytes, otherwise the title screen bitmap will display garbage over it – also there is a high possibility that large music files can cause major disruption to the graphics/presentation. *If you use multi-track music as your title music filename, or you disable in game music completely from this option is not required.*

***Do not use this option if you load in large title music files for the title screen music. This is because the in game music will overlap the large file's title music and will cause the title music to mess up with a possibility of the music crashing the program.***

## 5: Adding graphics

The SEUCK Title Screen Maker allows you to insert various specifications of graphics for your title screen. If looked at the example title screen by pressing 7 on the main menu, you will have noticed that it contains a logo, and also a character set with animation.

### 1.Loading your logo



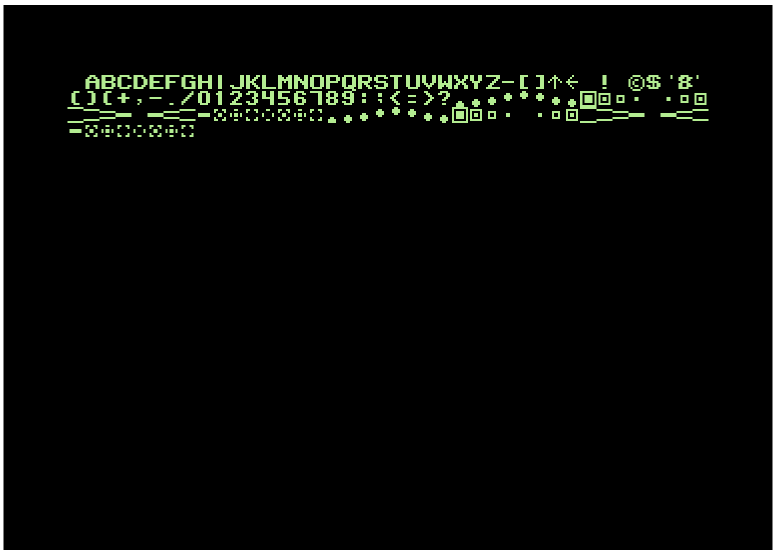
***Example: CETI 22's logo, featured in SEUCK Title Screen Maker V1.7***

SEUCK Title Screen Maker V1.8 allows you to load either an **OCP/Advanced Art Studio** hi-resolution bitmap logo or a multi-colour bitmap logo in **Koala Paint** format. The dimensions of your bitmap logo should be **40 chars width x 8 rows** height. Otherwise your logo won't show up correctly on the title screen. Also your drawn logo should be placed at the top of the screen.

After loading the filename you will be asked if your logo is hi-resolution or multi-colour. Type in **<H>** if your logo is hires OCP Art Studio format or **<M>** if your logo is multi colour Koala Paint format. Your logo will then display on screen in one of the two formats. Press space bar to exit.

***- Please take note that the background colour of the previous setting in the title screen will fill the colour RAM of the OCP hires bitmap. Ideally you should select the background colour in the settings menu before you load in the hires bitmap. Loading Koala Paint multi-colour bitmap is unaffected.***

## 2. Loading in a charset



***Example: Charset for CETI 22***

Your new front end will need a cool custom character set. The SEUCK Title Screen Maker allows you to load in your own or other custom character sets. You can only use 128 characters for your title screen. This is because the rest of the characters are over written with the screen RAM.

***Note that it is possible to use 1x2 and 2x1 fonts. However, the charset preview displayer will only display in 1x1 charset mode.***

The first 64 characters can be used as the main characters for typing in text. However the remainder of the 64 characters can either be use as additional symbols, or animation frames (8 chars = 1 animation). If your game is to include a hi score character for entering your name, you will need to make the UP ARROW character as an END indicator, and BACK ARROW as DELETE. There are plenty of character editors which you can use to make your custom character. My personal favourite is Char Editor V2.0 by Dunex.

**Fonts made as raw bin format are NOW compatible with SEUCK Title Screen Maker**. So if you created a character set using PC applications such as ***Charpad, Ultrafont PC, Cuneiform, etc.*** then you will notice this feature. A question will be asked "**Is the charset to be loaded raw binary format?"** If you enter **'Y' (Yes)** then you can load charsets exported from raw .bin files imported into your D64 (as PRG). Otherwise pressing **'N'** will result in loading character sets which are made in native C64 graphics tools, like Dunex Char Editor or any other C64 Public Domain font/character editor.

The first 64 characters can be used as **text**. However there are another 64 characters that can be used. These can be used as animated characters (each animation contains 8 frames) or symbols. Alternatively, it is possible to use 2x1 characters. For animation and/or additional symbols, "**shifted**" keys are used. Animated fonts require **8** frames from char $40 (64). You can have a maximum of **4** animations.

***If using 2x1 characters, you should make sure the hi score table is disabled in the settings menu, otherwise you'll get garbage displayed on the hi score table. The hi score table is only designed for 1x1 characters.***

### Loading and using 1x2 fonts

The second question (after loading) will ask if the character set is a "1x2" font. If you answer 'Y' to the question, it will skip the animation frames – also it will re-format the scroll text format to 1x2 characters. Where two rows are being used for the scrolling text.

***Please note that by selecting the 1x2 font option, there must be no hi score table available. Where this is the case, it must be disabled from the settings menu. Otherwise if that is enabled, you will only see half of the hi score character set on screen.***

### Hi-resolution or Multicolour?

After loading your font, press <H> or <M> to select option. The charset in that format will then display on screen. If you use multicolour charsets, in preview press 1 to change multi colour 1, press 2 to change multi colour 2. Then press space to store and return to the main disk menu.

## 5: Loading and saving a title screen

It would be pointless to not be able to load in any title screen files for displaying in your SEUCK games. Luckily after saving a finished title screen you are able to load it from disk for further updating. Unfortunately it is not possible to do this with older versions of the SEUCK Title Screen Maker, but good news is that it is possible to re-use your older SEUCK title screens from older versions of the editor.

### 1. Load title screen

To load in an existing title screen which has been made. Type in the filename of the front end. Remember to end the filename with a * for example RAYFISH.*TITLE** otherwise the filename will not load.

***Avoid using title screens made from older versions of SEUCK Title Screen Maker, because the title screen code has been altered. Using an older title screen might cause this program to crash. However if you made a title screen using older versions and wish to load it into this brand new version of the SEUCK Title Screen Maker. It is possible to load the text and scroll text into the editor via an Action Replay Cartridge or any machine code***

***monitor. (VICE monitor users use the command "bank ram" then l or s "filename" 1 xxxx yyyy when loading or saving to memory from disk).***

*To save existing text from Version 1.5/V1.5+ use:*

```
s "textandcolour",8,7160,7660
s "scrolltext",8,aa00,acc0
```

*To save existing text from Version 1.6 use:*

```
s "textandcolour",8,728a,76ea
s "scrolltext",8,aa00,acc0
```

*To save existing title screen text from Version 1.7(+) use:*

```
s "textandcolour",8,74db,79b3
s "endscreencolour",8,7e00,8000
s "scrolltext",8,aa00,acc0
s "endscreen",8,ae00,b200
s "hiscores",8,b200,b400
```

*To load existing text (Credits + colour and scroll text) from version 1.6 or lower use:*

```
l "textandcolour",8,74fa
l "endscreencolour",8,7e00
l "scrolltext",8,aa00
l "endscreen",8,ae00
l "hiscores",8,b200
```

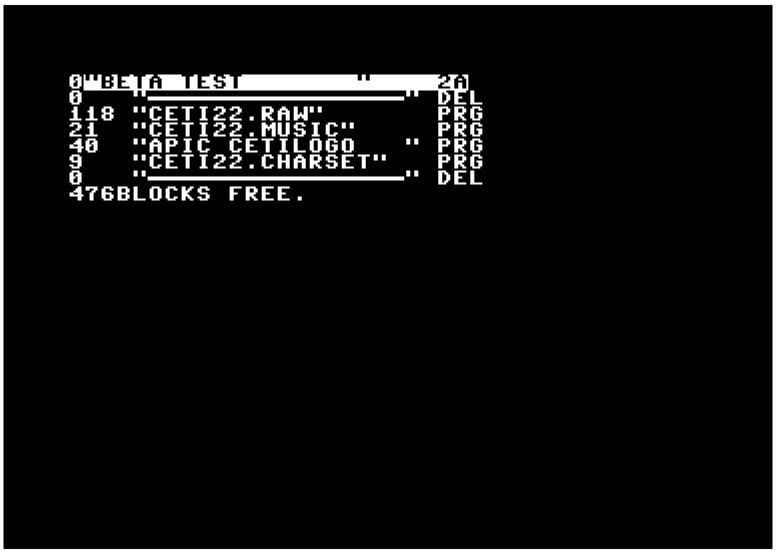
**You may need to re-edit the flashing colour data from the hi-score editor and also the char colour data from the scroll text editor.**

## 2. Save Title Screen

After you have finished with your session making a new title screen and its other bits, or decide to wait until some other day to finish it off. There is an option to save your title screen, complete with its settings, text, music, etc. Basically everything you have put in place saves to disk.

***This is also the option you need to use to save your finished title screen. Linking to your finished SEUCK game has to be done manually.***

## 3. View disk directory



```
0"BETA TEST          " 2A
0    "               "   DEL
118  "CETI22.RAW"        PRG
21   "CETI22.MUSIC"      PRG
40   "APIC CETILOGO  "   PRG
9    "CETI22.CHARSET"    PRG
0    "               "   DEL
476BLOCKS FREE.
```

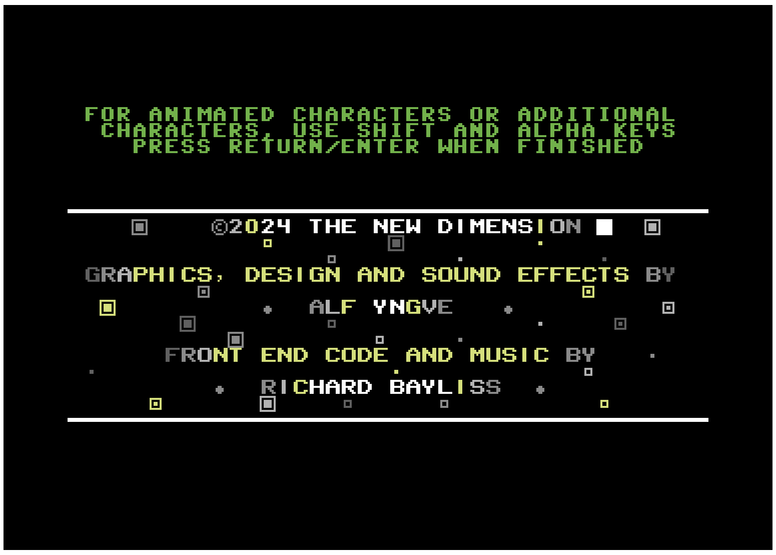
*An example of displaying a disk directory*

You will want to see how much disk space your disk/D64 has, pressing 7 will display the directory listing of all of the files that are on your disk. To slow down the display of the directory, on your C64 hold CONTROL. After the directory has finished, press SPACE BAR to return to the disk menu.

## 6: Creating your own title screen credits page

Now that the disk functions are out of the way, it is time to have some fun and create and design your very own title screen, hi score and other bits. Using a series of options from the main menu. It is made possible. All input options are literally the same.

You can now see the editor text on screen (using CBM charset). The title screen's purpose is to create a simple, colourful and wonderful looking credits screen for the front end. Alternatively you can add additional characters or animated characters, by entering the SHIFT and keys @, A-Z. Use F1 / F3 to enable/disable grid.



```
FOR ANIMATED CHARACTERS OR ADDITIONAL
 CHARACTERS, USE SHIFT AND ALPHA KEYS
   PRESS RETURN/ENTER WHEN FINISHED

      ©2024 THE NEW DIMENSION

GRAPHICS, DESIGN AND SOUND EFFECTS BY
            ALF YNGVE

     FRONT END CODE AND MUSIC BY
          RICHARD BAYLISS
```

**Example: Editing the credits page for CETI 22**

Use CONTROL+NUMBER KEY 1-7 / CBM+NUMBER KEY 1-7, to pick a colour for your cursor (Use TAB+NUMBER KEY / CTRL+NUMBER KEY if using a PC/Laptop with a C64 emulator such as VICE, CCS64, etc.).

Use F5/F7 to enable/disable grid mode. The grid mode is used to help try and work out the central area of the screen, while typing in text and that.

It is also possible to use multi-colour characters. This is only enabled if you have selected multi colour mode after loading in your character set. Multicolour charsets can be used by pressing CBM+Numbers 1 to 7.

While editing inside the space between the two white borders. You can also use SHIFT+KEYS to display symbols, or (if enabled from load charset option), you can use animated characters. Animated characters. Note that you can only have 4 different animated characters. SHIFT+CLEAR HOME will erase the title screen credits.

***Central alignment of text is not possible in this editor. You should do this manually. (Simply enable grid mode and use use INST/DEL (backspace/shift+backspace) to move the characters backward/forward.***

It is also possible to use 2x1 characters in your title screen presentation. To do this, simply use shifted characters.

For example:

using 2x1 characters

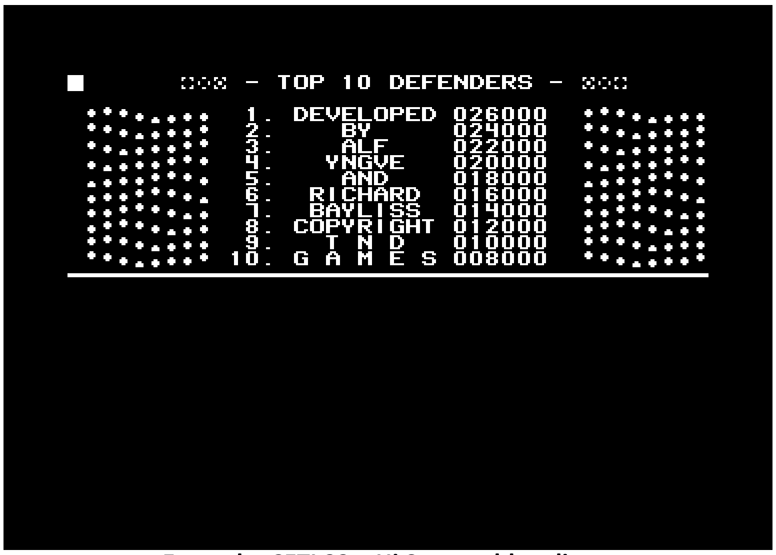
**`HheElLlLoO wWoOrRlLdD`**

or if using 1x2 characters

```
hello world
HELLO WORLD
```

After you have finished making your cool front end. Press RETURN/ENTER to store it and then go back to the main menu screen.

## 7: Creating your Hi Score table and Flashing Colour Table



```
- TOP 10 DEFENDERS -
 1. DEVELOPED 026000
 2.    BY     024000
 3.    ALF    022000
 4.   YNGVE   020000
 5.    AND    018000
 6.  RICHARD  016000
 7.  BAYLISS  014000
 8. COPYRIGHT 012000
 9.   T N D   010000
10. G A M E S 008000
```

**Example: CETI 22 – Hi Score table editor**

The Hi Score editor is yet another cool, and pretty fun option for those of you who would like a front end to not only display the credits page presentation, but also a hi score table list. This is the first non -programmable SEUCK support tool that can do this for you. It is now possible to include animation on the hi score panel.

Editing the hi score list is easy, but there are some restrictions. You can edit the names, the hi score values, but **you must not change the layout/character position of the names or hi score table**. You can still edit the heading and the names and hi scores. Max number of characters per name is **9 chars**. Valid chars allowed for names are **alphanumeric** only. Avoid making a score of 000000 points in the table, otherwise it will confuse the two player mode (even if the game is set to one player mode).

*If you mess up, don't worry. It is possible to restore a default hi score table list which you can edit.*

As soon as you are happy with the hi score list. Press RETURN/ENTER to store it.

You will then be asked to enter a value of colours for the hi score table flashing. If you don't wish to have a flashing hi score table, simply fill the values with one colour.

If you are using a multi colour character set, please use CBM+NUMBER key to select cursor colour in order to see your text properly.

### The colour table

***@ = black / A = white / B = red / C = cyan / D = purple / E = green / F = blue / G = yellow***

***H = orange/char multi colour black / I = brown/char multi colour white***

***J = pink/char multi colour red / K = dark grey/char multicolour cyan***

***L = medium grey/char multicolour purple / M = light green/char multicolour green***

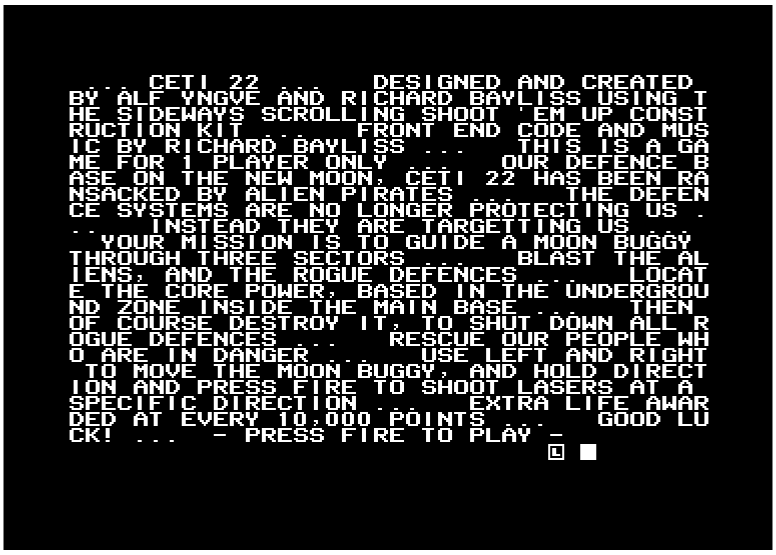
***N = light blue/char multicolour blue / O = light grey/char multicolour yellow***

After you have finished filling in the colour flash table for the hi score table (It also features in the GET READY, GAME OVER and HI SCORE text) press RETURN.

***Warning: Using a saved title screen that does NOT support the hi score table will clear it away completely and will not be possible to use. A saved title screen clears all of the hi score table data before saving to disk. Also you cannot use 1x2/2x1 character sets for the hi score table.***

***If you used a 1x2 or 2x1 charset, you SHOULD NOT use the hi score table.***

## 8: Writing your own scrolling text message



```
 ...  CETI 22 ...    DESIGNED AND CREATED
BY ALF YNGVE AND RICHARD BAYLISS USING T
HE SIDEWAYS SCROLLING SHOOT 'EM UP CONST
RUCTION KIT ...     FRONT END CODE AND MUS
IC BY RICHARD BAYLISS ...    THIS IS A GA
ME FOR 1 PLAYER ONLY ...    OUR DEFENCE B
ASE ON THE NEW MOON, CETI 22 HAS BEEN RA
NSACKED BY ALIEN PIRATES ...    THE DEFEN
CE SYSTEMS ARE NO LONGER PROTECTING US .
..    INSTEAD THEY ARE TARGETTING US ...
  YOUR MISSION IS TO GUIDE A MOON BUGGY
THROUGH THREE SECTORS ...    BLAST THE AL
IENS, AND THE ROGUE DEFENCES ...    LOCAT
E THE CORE POWER, BASED IN THE UNDERGROU
ND ZONE INSIDE THE MAIN BASE ...    THEN
OF COURSE DESTROY IT, TO SHUT DOWN ALL R
OGUE DEFENCES ...    RESCUE OUR PEOPLE WH
O ARE IN DANGER ...    USE LEFT AND RIGHT
 TO MOVE THE MOON BUGGY, AND HOLD DIRECT
ION AND PRESS FIRE TO SHOOT LASERS AT A
SPECIFIC DIRECTION ...    EXTRA LIFE AWAR
DED AT EVERY 10,000 POINTS ...    GOOD LU
CK! ...   - PRESS FIRE TO PLAY -
```

**Example of the scroll text editor being used for CETI 22**

So then, you have made a front end with credits and possibly a hi score, but you would also like to have a scroll text for your game title screen. Selecting this option allows you to do exactly that. Enter the speed of the scrolling message (1 -5) 1 is the slowest possible speed, 5 is the fastest, which 5 of course might be unreadable in 1x1 or 1x2 char mode. The scroll text editor works the same way as editing the front end. You can type up to 24 rows of scroll text for presentation. It could be used for either adding additional credits, instructions, greetings, silly stuff or whatever you like to add.

After the scroll text is finished, press @ to set a loop marker to your message. The loop marker [L] in place is then marked. If you are using a multi colour character set, please use CBM+(0-7) key to select cursor colour in order to see your text properly. Use F1/F3 to swap to CBM Character mode, if you are writing a scroll for a title screen that uses double-height characters.

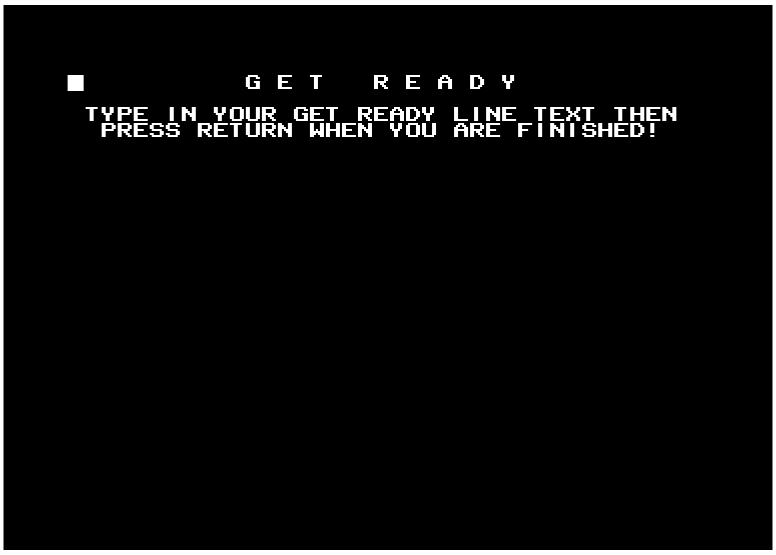
Like with the front end editor, it is possible to use animated charsets or additional characters using shifted keys, or 2x1 characters, but be very careful using them. At the end of your scroll text, type @ (which is marked with a box with a letter L inside it) then press RETURN/ENTER to exit.

**If you are writing text for a 1x2 scrolling message please use single characters, but do not use the 1x2 font until after typing in your scroll text.**

Now you can edit the 40 characters that form the colour of the scroll text. Please refer to [3] Edit hi score table option in this manual for colour chart.

***If you don't want to write a scroll text simply clear it and at the start of the screen enter a couple of spacebar characters and then place a restart pointer in the scroll text.***

## 9: Get ready and game over screens


```
■         G E T   R E A D Y
  TYPE IN YOUR GET READY LINE TEXT THEN
   PRESS RETURN WHEN YOU ARE FINISHED!
```
**Example of the GET READY and GAME OVER screen editor**

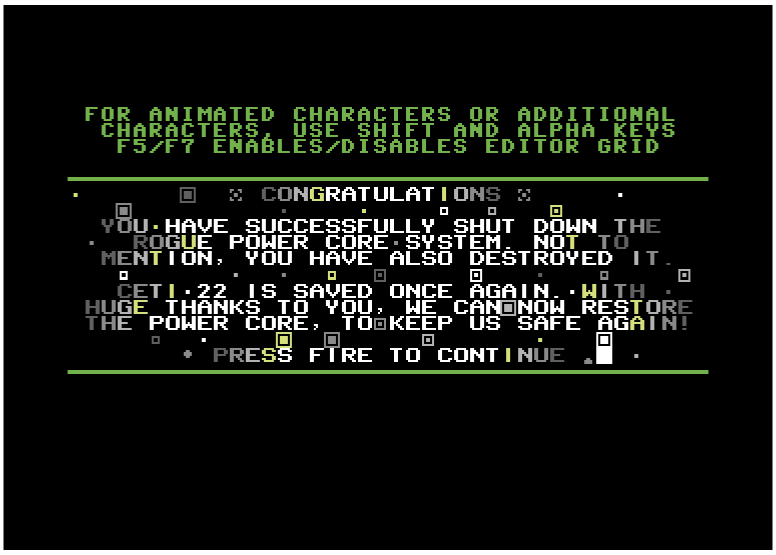
You can also edit a single line text for GET READY and also GAME OVER. There is not much that needs to be done here. First enter the GET READY message (with a maximum of 40 characters). Then press RETURN/ENTER. Next edit the GAME OVER message (also with a maximum of 40 characters). Then press RETURN/ENTER. Your scene will then be stored into memory and return to the disk menu.

It is possible to use multi colour or 2x1 chars and for this feature. **Text for 1x2 characters will be automatically formatted to that size after loading a 1x2 character set. So you can still use single characters like with the scroll text editor.**

You can also edit a single line text for GET READY and also GAME OVER. There is not much that needs to be done here. First enter the GET READY message (with a maximum of 40 characters). Then press RETURN/ENTER. Next edit the GAME OVER message (also with a maximum of 40 characters). Then press RETURN/ENTER. Your scene will then be stored into memory and return to the disk menu.

It is possible to use multi colour or 2x1 chars and for this feature. Text for 1x2 (double height) characters will be automatically formatted to that size after loading a 1x2 character set. So you can still use single characters like with the scroll text editor.

## 10: Creating an end screen



```
FOR ANIMATED CHARACTERS OR ADDITIONAL
 CHARACTERS, USE SHIFT AND ALPHA KEYS
  F5/F7 ENABLES/DISABLES EDITOR GRID

          CONGRATULATIONS

 YOU HAVE SUCCESSFULLY SHUT DOWN THE
   ROGUE POWER CORE SYSTEM. NOT TO
 MENTION, YOU HAVE ALSO DESTROYED IT.

  CETI 22 IS SAVED ONCE AGAIN. WITH
HUGE THANKS TO YOU, WE CAN NOW RESTORE
THE POWER CORE, TO KEEP US SAFE AGAIN!

       PRESS FIRE TO CONTINUE
```

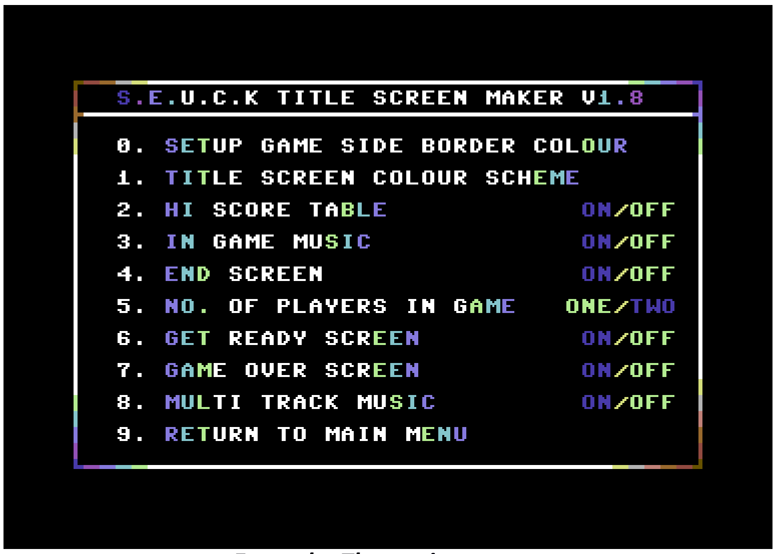
*Example of the end screen editor*

The SEUCK Title Screen Maker V1.7 now has two new main features. There is the End Screen maker and also the updated settings, to allow multi-track music. The end screen maker works the same way as the credits page editor. However, size is a little limited for the end screen, due to title screen memory taking over a lot of space – but it is still possible to let it stand out. Much better than just allowing the game to loop.

The end screen maker allows you to also include animated characters (just like with the previous version). Use F5/F7 to enable/disable grid mode. The grid mode is used to help try and work out the central area of the screen, while typing in text and that. The end screen editor allows you to edit 11 lines.

Use CONTROL+NUMBER KEY 1-7 / CBM+NUMBER KEY 1-7, to pick a colour for your cursor (Use TAB+NUMBER KEY / CTRL+NUMBER KEY if using a PC/Laptop with a C64 emulator such as VICE, CCS64, etc.). Also use SHIFT+A-Z keys to plot animated characters on to the screen or enter the second half of the text.

## 11: Setting up the title screen and game preferences / music settings, etc.



```
 S.E.U.C.K TITLE SCREEN MAKER V1.8

 0. SETUP GAME SIDE BORDER COLOUR
 1. TITLE SCREEN COLOUR SCHEME
 2. HI SCORE TABLE               ON/OFF
 3. IN GAME MUSIC                ON/OFF
 4. END SCREEN                   ON/OFF
 5. NO. OF PLAYERS IN GAME      ONE/TWO
 6. GET READY SCREEN             ON/OFF
 7. GAME OVER SCREEN             ON/OFF
 8. MULTI TRACK MUSIC            ON/OFF
 9. RETURN TO MAIN MENU
```

*Example: The settings menu*

The settings menu in the SEUCK Title Screen Maker is a handy option for helping you setup the game title screen properties and in game features that would suit your very own SEUCK game. Unfortunately this program does **not** allow you to **completely** enhance your SEUCK game with special features such as **power ups** or **background animation enhancements**. You will need to flip over to side 2 of the SEUCK Title Screen Maker and have an Action Replay/Retro Replay handy (or use VICE M/C monitor), load in Turbo Assembler (or use the pre-built assembler in

Retro Replay by typing in **TASS** in Fastload as it will also work with that.). The source would then need to be loaded into the Turbo Assembler and then assembled to disk. Then you manually install the features yourself (More about this later on). The SEUCK Title Screen Maker does however give you simple options to suit your game, before you add enhancements.

## 0. Setup game side border colour

The side border colour in default SEUCK games have always been black. However it is now possible to change the side border colour by selecting option 0 in the settings menu. Then type in a single character to represent the side border colour. (Keys @, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O).

## 1. Title screen background colour

This option helps you pick out the colour you would like your title screen to use. It is important that this feature is selected before loading a hi- resolution OCP Art Studio bitmap in order for the colour RAM to be filled in correctly with the background colour. Pressing 1 will cycle the border colour. After you have finished picking the colour, you can either stop pressing the '1' key or return to the main menu. Either way, the background colour is automatically stored.

## *2.* Hi score table ON / OFF

By pressing '2' in this menu, it will toggle whether or not you would like a hi score table enabled/disabled for your game. If you are using 2x1 characters containing 128 characters this MUST be disabled otherwise your hi score table will just display a load of garbage.

## *3.* In game music ON / OFF

Most C64 SEUCK games contain only in game sound effects. If this is the case, you might want to disable the in game music feature. Also if you want to use a Title music file (with or without sub-tunes) but it uses more than $1000 bytes in memory, but less than $2000 bytes then in game music should be disabled.
Sometimes having just title music (after completely cleaning the title screen) will result to saving more memory for additional code for in game enhancements. Pressing '3' in this menu will toggle in game music option.

***MULTI TRACK tunes is a feature which replaces in game music and may use an assigned track, if Multi-Track mode support has been enabled.***

## *4. End Screen* ON / OFF

SEUCK games are very common for looping games after completion of the whole game, in order to encourage the player to build up more points in its score panel. Not everybody likes that idea, and would prefer the game to end. Just like CETI 22 does. This option allows you to loop or end the game, by pressing '4'. If you disable loop in the options the game will end directly to a small end screen, designed in SEUCK Title Screen Maker.

## *5. No. of players ONE / TWO*

If you are making a SEUCK game which caters for one player only, but player two is used as the player only. This feature needs to be set as ONE player, in order to prevent confusing the hi score table name entry routine. Pressing '5' will toggle/select how many players your game uses. ***Be warned, in SEUCK either player 1 or player 2 must be disabled****.*

***If you do not wish to have two players starting at the same time, when Joystick in Port 1 is pressed to start a 2 player game. There is a code snippet (on side 2 of the disk/image) which can be linked to the title screen as an enhancement. You'll need to assemble and run this first. Otherwise two players will start at the same time on starting a 2 player game on joystick port 2.- It was intended to be that way.***

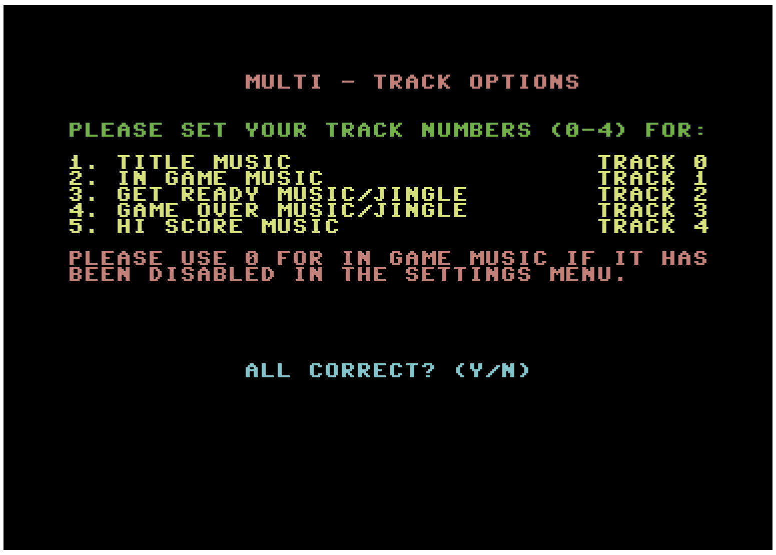
## *6. GET READY screen ON / OFF*

This option selects whether your game should enable/disable the GET READY sequence from the title screen. If it is disabled, pressing FIRE on the title screen will jump directly to the game instead.

## *7. GAME OVER screen ON / OFF*

Pressing '7' will toggle between on/off settings for the GAME OVER screen. If you disable the GAME OVER screen option, make sure to also disable the HI SCORE table option as well as the code will just jump straight on to the front end.

## *8. Multi Track Music Support ON / OFF*



```
MULTI - TRACK OPTIONS

PLEASE SET YOUR TRACK NUMBERS (0-4) FOR:

1. TITLE MUSIC                  TRACK 0
2. IN GAME MUSIC                TRACK 1
3. GET READY MUSIC/JINGLE       TRACK 2
4. GAME OVER MUSIC/JINGLE       TRACK 3
5. HI SCORE MUSIC               TRACK 4

PLEASE USE 0 FOR IN GAME MUSIC IF IT HAS
BEEN DISABLED IN THE SETTINGS MENU.


ALL CORRECT? (Y/N)
```

***Example of Multi-Track Music support***

Pressing '8' will ask if you would like to use multi track tunes. Pressing 'N' will revert back to the old title at $8000 and in game music at $9000. Otherwise pressing 'Y' will allow you to set up the correct tracks to play for the following features:

- Title music
- In game music
- Get ready music
- Game over music
- Hi score music

In order to be able to use this option, all of the sub tunes must be stored in the same file as the Title music. Music editors which support this feature are: Future Composer V4.0, Cosine EMS V7.03, Demo Music Creator (Any version), Goat Tracker V2, Cheese Cutter.

Known music editors that do not support this feature are Voice Tracker, Music Mixer, Dutch USA Team music assembler, Sid Factory V2. However you might be able to find an alternative solution to this problem, like for example combining multiple music files and make an own player to support sub tunes using all of them. Note that it will take up memory using this method. So be very careful what you do.

Another good thing about this Multi-Track support option is that you can have up to five different tunes (tracks 0-4) set up in any order. You can assign these five tracks by entering numbers 0 to 4 to represent the track values (1-5) in your music file. ***Your music file must be loaded as a title music file first.***

***CETI 22***'s music was composed using DMC V5.0+ and has four different tunes. It has been intended that the game does not use in game music. So the following is assigned:

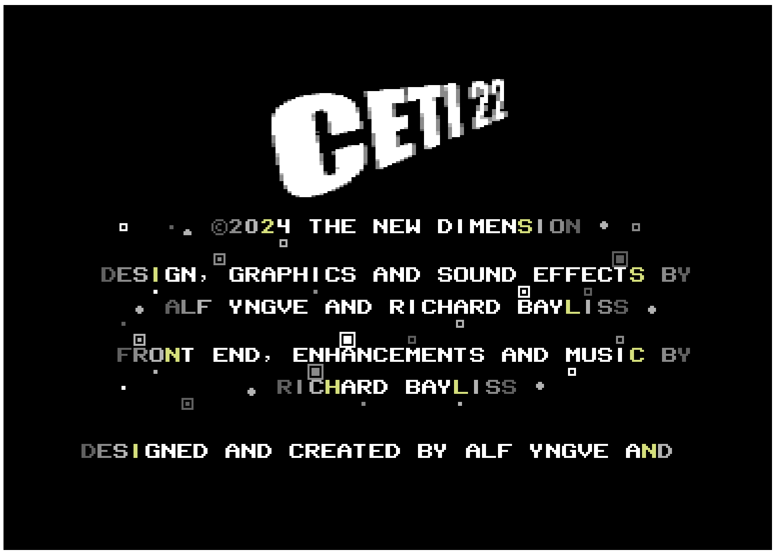
***Title music: Track 0***
***In game music: Track 0***
***Get ready music: Track 1***

***Game over music: Track 2***
***Hi score + end music: Track 3***

### 9. Return to main menu

This option leaves the settings/preference options and returns back to the main menu screen.

## 12: Testing the title screen and end screen



©2024 THE NEW DIMENSION

DESIGN, GRAPHICS AND SOUND EFFECTS BY
ALF YNGVE AND RICHARD BAYLISS

FRONT END, ENHANCEMENTS AND MUSIC BY
RICHARD BAYLISS

DESIGNED AND CREATED BY ALF YNGVE AND

***Example of the custom front end with new features implemented***

### 8. Test title screen

After making your front end, you'll want to see what it looks like after you have finished making it. Pressing 8 will show the front end. First the credits page appears on screen. After several seconds, the hi score table will display on screen (if of course you have enabled it). At the bottom you will see a scrolling message. Pressing fire on either joystick port will then test GET READY, and pressing again will display the GAME OVER message. That is if those two options have been enabled. Pressing fire on these will then return to the main menu screen.

### 9. Test end screen

This final option allows you to test the end screen to see what it looks like. The end screen will display a plain screen with 11 lines of text/characters. This is to say “well done” to the gamer, or if you want to, you could just create something else from it.

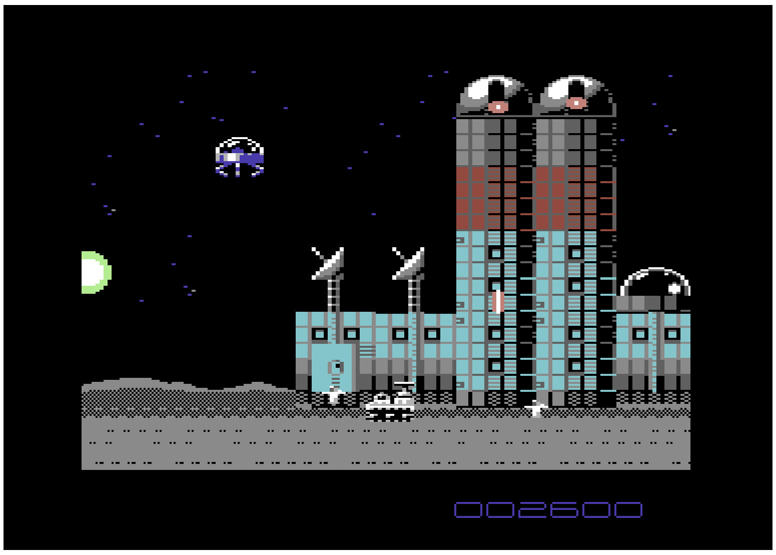
## 13: Linking your title screen to your game

You may have noticed that you can save your finished title screen, but there is no option to link the front end to your game. Why is this? That is because it would be literally impossible to link the front end directly from the editor. A whole SEUCK game will overwrite the memory if unpacking or loading. There is no option in the disk menu of **SEUCK Title Screen Maker** to load a SEUCK game anyway.

There is an alternative solution however. For this you will need a freezer cartridge such as an ***Action Replay / Retro Replay****. It might also be possible to use a* ***Super Snapshot*** or TLR's ***Super Fluid*** *plug-in,* although you might need to find an alternative solution. It is also possible to use **VICE's** built in monitor to link your front end to your SEUCK game. How you decide to make an executable of your finished production is entirely up to you. Also you will need a stand-alone SEUCK or Sideways scrolling SEUCK game (saved as finished game state). It does not matter if your game gets crunched or frozen to an executable runnable program on your C64.

### METHOD 1: FREEZER CARTRIDGE (ACTION REPLAY), FASTLOAD, AR FREEZER AND BACKUP

1. First of all **load** and run your **Shoot Em Up Construction Kit** or **Sideways Scrolling SEUCK** game. It can either be from the **stand-alone game** file, the **save finished game** file state or **directly into the SEUCK editor** itself. Start the game and leave it in play mode. This is because we do not want the score panel to be inverted.



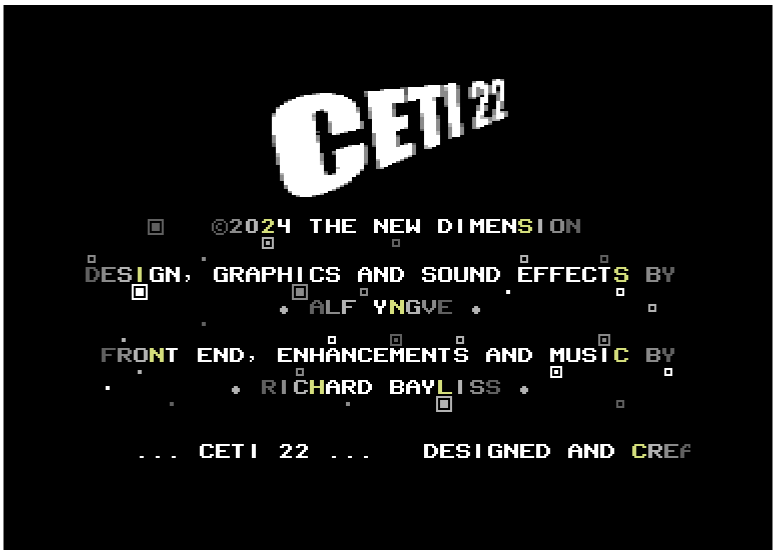
**CETI 22 – Start up game**

2. While still in play mode, press the **reset** button on your freezer cartridge and select either **normal reset** or **fast load** option. (***Do not select F1 Configure Memory, because that will wipe everything from memory***). Now load in the **title screen** from your disk.

3. Load in the **AR Freezer** tool (**LOAD "AR FREEZER",8,1**) from fast load then type **RUN**. This will turn the screen black for a short period of time.

4. On the ***black*** screen, press the **FREEZE button** on your **freezer cartridge** (**In VICE use ALT+Z**) then enter the **Freezer Backup Menu** (Pressing F1) and then **save** your production to **disk** or **tape** (or digital D64/TAP image).

5. After saving has finished, return to the program. It will unfreeze and your new front end will display. Just like this example below:



CETI 22

©2024 THE NEW DIMENSION

DESIGN, GRAPHICS AND SOUND EFFECTS BY

ALF YNGVE

FRONT END, ENHANCEMENTS AND MUSIC BY

RICHARD BAYLISS

... CETI 22 ... DESIGNED AND CREA

6. (Optional). Load in a cruncher like **Abuze Crunch V3.2+**, or any other cruncher, and then crunch the program, use **$080D** as the jump address and **$37** as **$01** value. Save the file, Then load and run the crunched game.

***Congratulations, you have just enhanced your SEUCK game with a new front end. Now share it with your friends, family or even the online C64 community.***

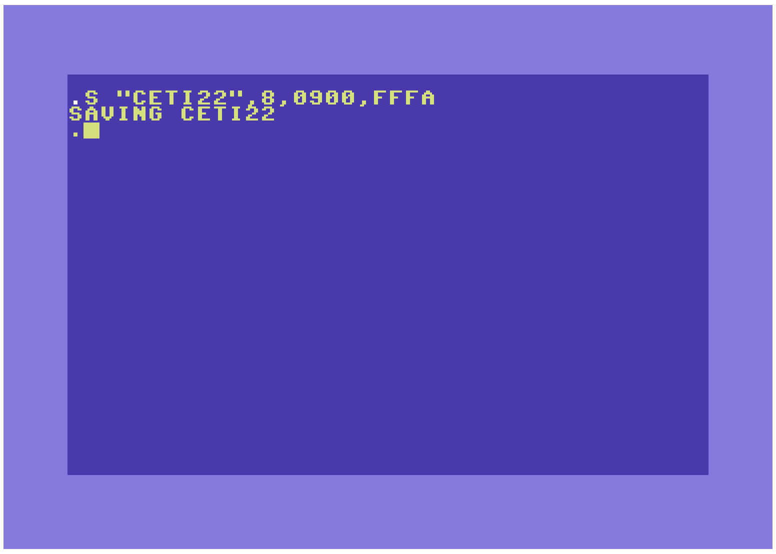
## *METHOD 2: FREEZER CARTRIDGE MACHINE CODE MONITOR (ACTION REPLAY), PACKER AND CRUNCHER*

1. Go to **FAST LOAD** or **NORMAL RESET**, load and run your SEUCK game or load the SEUCK editor with your work file, then start your game in **play mode**.

2. While the game is in play mode press the **RESET** button on your Action Replay (or any other freezer cartridge) and then go to **FASTLOAD**. Enter the machine code monitor (Press F8). Insert your title screen disk and then type in the following command:

***L "TITLESCREEN*",8 (Or whatever device you have available)***

3. Insert a disk containing 250 blocks or more free. Then save the whole game+title screen using

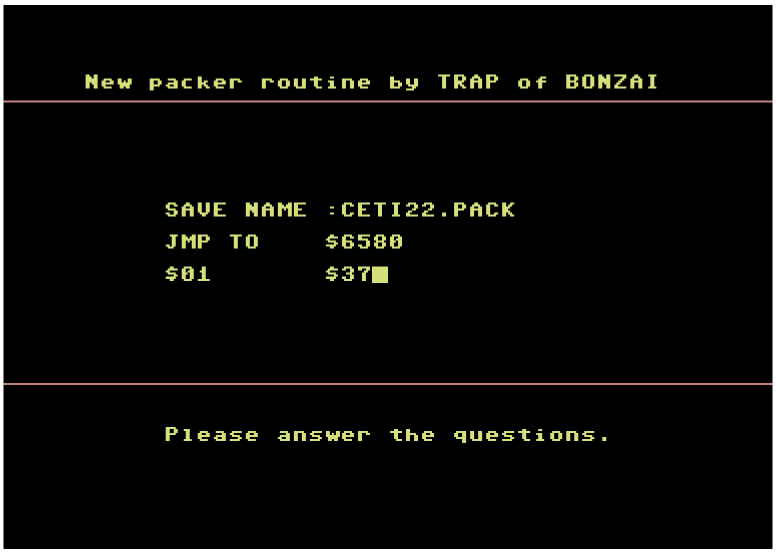


```
.S "CETI22",8,0900,FFFA
SAVING CETI22
.
```

***S "NAME OF GAME",8,0900,FFFA***

***Example of loading and saving in Action Replay's machine code monitor***

4. Load in a C64 packer from CSDB or any Public Domain disks and use **$6580** as the jump address for the title screen, and **$37** as **$01** value. *Easiest option would be to use **Sledgehammer II**. When asked to use SEI/CLI in any packers that have that option, select SEI. Wait patiently until packing is finished, then save your program to disk when prompted to. Make sure to do a test run before selecting option 5.*


```
New packer routine by TRAP of BONZAI

SAVE NAME :CETI22.PACK
JMP TO     $6580
$01        $37

Please answer the questions.
```

*5. Example Using Sledgehammer 2 for packing*

6. Load in a cruncher from CSDB or from any Public Domain disks for example **Abuze Crunch V3.2+** if you do not have 256K+ REU to use very fast and heavy crunchers such as **Cruncher AB V1.0+ (Crest version) or Byte Boiler V1.0**. Use the jump address of the de-pack routine from the packer for the start address, and use **$01** value as **$37**. Load and wait a few minutes for crunching to finish. Then save to disk. Programs that are packed with Sledgehammer II tend to use $080D as the jump address. Most *C64 crunchers will take some time. Patience is recommended.*

***An alternative solution is to use Style's DIR Master on PC and a cross-platform cruncher instead of a packer and a cruncher – it saves a lot of time unless you like to be very old school and pack and crunch like back in the 1980s and 1990s.***

## METHOD 3: VICE and cross-platform cruncher (Exomizer)

There are a few different cross-dev crunchers available, but Exomizer is the main target for this part in this chapter.

*This option does not require any additional cartridge images. However you will need a cross-platform cruncher like Exomizer V3.1.2 (or lower). There are other cross-platform crunchers which will allow to do the same thing, but may use different commands for compression.*

1. Load and run your SEUCK game like normal and start the game in play mode.

2. Do a **SOFT RESET** from the VICE menu, or shortcut keys provided (ALT+R in VICE V2.4-V3.2, or SHIFT+F9 in VICE V3.3 or higher)

3. Enter the **VICE** machine code monitor

4. Before doing anything, type in ***bank ram*** to enable all ram areas for loading/saving. This is needed so that all 4 VIC banks are used. SEUCK games use exactly that.

5. Type `L "TITLESCREEN*" 8 6580` to load in your new title screen into the SEUCK game

6. Type `s "NAMEOFGAME" 8 0900 FFFA` (if using memory below $0900 for additional code or enhancements, use the start of that memory address instead)

7.Type `g 6580` to run new title screen. This will run the new front end. Make sure you detach your D64 image.

8. Use D64 exporting software, such as Style's **DIR MASTER** and copy the PRG file to **Exomizer's** directory (or which other cruncher you are using). Then using the **command prompt**, go to **Exomizer's** directory. For

example:

```
cd\c64\tools\exomizer\win32
```

9. Use `exomizer sfx $6580 mygame.prg -o mygame+.prg -x2`
   `(-x2 gives flashing border. If you want no effect, use -n instead)`

   If using Crunchy instead (from Turbo Coder 64 menu at https://retro64.ca)
   **Compression type:** Self Expanding with BASIC RUN line
   **Code location:** $6580
   **Target Computer:** Commodore 64
   **De-crunch Effect:** Select any of the decrunch effects available or none
   **File to crunch:** nameofgame.prg

5. After crunching has finished (or you downloaded your crunched file from Crunchy), **export** mygame+.prg to a fresh **D64** and load into **VICE**.

***If everything is working. Congratulations, you have successfully added a brand new title screen to your SEUCK game.***

## 14: Adding in game enhancements

It is now possible to enhance your SEUCK game even more, by adding power ups and other special features. What makes things even cooler is that V1.7 and V1.8 of the SEUCK Title Screen Maker has **reserved 3 jump subroutines** for **initialising** and also **playing** additional routines. This means that it is now **possible** to add your very own in game enhancements to games that have been made with the standard or sideways scrolling SEUCK. As a matter of fact, I have done **exactly this** for CETI 22. All of which has been included as part of the SEUCK Title Screen Maker disk.

There are three parameter addresses available in memory in which you can link jump subroutines to routines in other memory areas. The Parameter areas are as follows:

Subroutine initialize:

**`PARAM1INIT = $6EB9`** (Always start with this parameter when adding in game enhancements)
**`PARAM2INIT = $6EBC`** (Always make sure the second parameter is called before running the third)
**`PARAM3INIT = $6EBF`** (This is the last one, but for more subroutines create new parameters and link to this one.

Subroutine play:

**`PARAM1PLAY = $6C70`** (Always start with this when adding the first in game enhancement)
**`PARAM2PLAY = $6C73`** (Always use as second)
**`PARAM3PLAY = $6C76`** (The last one)

If you take a look at these in the M/C monitor disassembler, you will see that all of those addresses are marked with a RTS (Return to Subroutine) command. These are spared for calling enhancements.

***By installing in game enhancements using the code snippets on side 2 of the disk, you can create real masterpieces, just like Stormbird. It is also possible to code your own enhancements that are not featured in SEUCK Title Screen Maker or SEUCK School. If you can code like a pro. Go for it :)***

On Side 2 of the SEUCK Title Screen Maker disk is a selection of **code snippets** in which at first will need to be loaded into **Turbo Assembler** or **Turbo Macro Pro** (or can be exported for use with a cross assembler like TMPx. The code is very adjustable and can be extended/shortened with other subroutines you want to create.  You can even create your own subroutines, providing that you set a template for your enhancement. The template om this page looks something like this:

**If making an enhancement of your own. Here's a code template to get you started.**

```
;STM V1.8 Enhancement
;Code Template by
;Richard Bayliss

;Set up initialise and play
;parameters for linking
;to the in game code

param1init = $6eb9 ;
param2init = $6ebc
param3init = $6ebf
param1play = $6c70
param2play = $6c73
param3play = $6c76

;     *** LOCATION TO LINK TO ***

paraminit = param1init ;← Pick paraminit1, paraminit2 or paraminit3
paramplay = param1play ;← Pick paramplay1, paramplay2 or paramplay3

;Set Start address to run and
;install enhancement

startaddr = $(your chosen address where spare memory lies)

    ;Start code
    *=startaddr
```

```
;Init/Play registry
;into one of the SEUCK Title Screen
;Maker's spare routines.

        lda #$20
        ldx #<init
        ldy #>init
        sta paraminit
        stx paraminit+1
        sty paraminit+2

        lda #$20
        ldx #<play
        ldy #>play
        sta paramplay
        stx paramplay+1
        sty paramplay+2

        rts ;Or use JMP $6580

        ;Enter your enhancement initialize code here
init

        rts

        ;Enter your enhancement play code here

play    jsr myroutine

        rts

        ;Main custom routine

myroutine
        ;flashing sideborder
        inc $d020
        rts
```

The example game from SEUCK Title Screen Maker V1.7 V1.8, **CETI 22** features 2 chosen enhancements, that have been set up and configured for the game. I'll explain about those later on.

In order to add your enhancements to games mastered with the SEUCK Title Screen Maker. The following C64 programs are recommended:

1. A **Freezer Cartridge**, such as an **Action Replay MKVI** or Retro Replay or any similar cartridge plug-ins.
2. An assembler such as **Turbo Assembler** or **Turbo Macro Pro.** Alternatively you can use DIR Master to export the source code and modify/assemble it to work with a cross assembler should you wish to.
3. A fully compiled SEUCK game with a new front end made with the SEUCK Title Screen Maker V1.6 or higher. It will not work with older versions.

Okay. Let's get started …

**Instructions for all enhancement code snippets**

You will need Turbo Assembler, Turbo Macro Pro or perhaps Dir Master and TMPx. All of these can be found at: https://turbo.style64.org

You will also need an Action / Retro Replay freezer cartridge or plug-in and also a native packer with cruncher. Alternatively you can use a cross-platform cruncher like the latest version of Exomizer. You can easily find these by googling C64 Packer, Cruncher. Alternatively, you can try out an online version of Exomizer called **Crunchy** by Retro64 which can be found at: https://retro64.co/crunchy. Other cross-development crunchers you can use are Bongo, PuCrunch, ByteBoozer, SubSizer, TS Crunch V1.3, LZMPI, or Dali. These tools can be found on CSDB.

Using Turbo Assembler or Turbo Macro Pro (or whatever), you should load a desired source file in which you would like to edit, setup and implement into your SEUCK game creation.

**Loading/Saving source listings**

SEQ file versions of source listings can be loaded by using ← and E. Enter filename (with .tas extension if loading the example code snippets on side 2 of the SEUCK Title Screen Maker V1.8 disk).

SEQ files of updated source listings can be saved by using ← and S. Enter filename and the source will save to disk.

PRG file versions of source code edited by yourself can be loaded by using ← and L. Enter the filename.

PRG file versions of source code edited by yourself can be saved by using ← and S. Enter the filename to save to disk.

**Assembling the source code**

**YOU CANNOT ASSEMBLE AND RUN THE CODE DIRECTLY FROM THE ASSEMBLER USING ← and 3 and start the code**. If you try this after loading your SEUCK game and title screen and / or the assembler could crash. Instead **you should assemble the source code to an object file directly onto disk**. This can be done by pressing **←** and **5**. After the object code has been assembled to disk. It will be ready to load after loading in the new SEUCK Title Screen which you have linked to your SEUCK game.

**Setting up the Parameters**

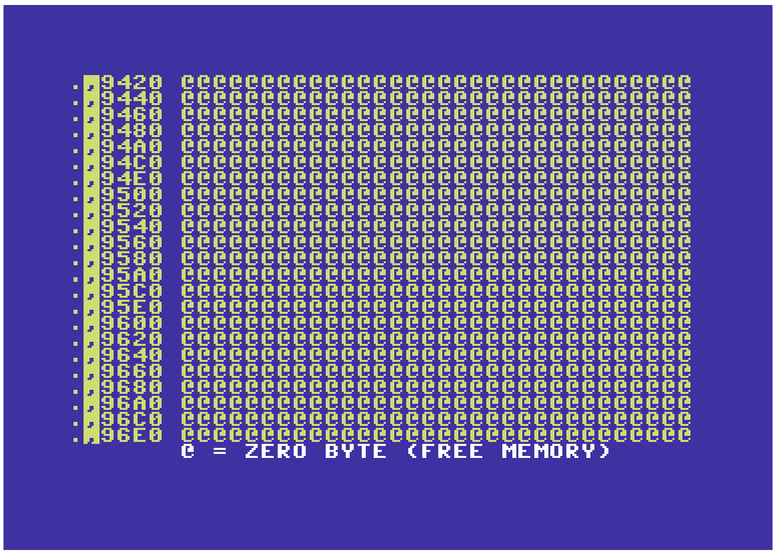
In all source files, you will need to set up the subroutine parameters by changing PARAMINIT and PARAMPLAY to initialise and play the enhancement parameters. For example:

```
PARAMINIT  = PARAM1INIT
PARAMPLAY  = PARAM1PLAY
```

This must always be set as the starting parameter for your SEUCK enhancements otherwise if you start with PARAM2INIT and PARAM2PLAY or PARAM3INIT and PARAM3PLAY otherwise enhancements will be ignored. You can set up to three different enhancements into your SEUCK game.

**Finding spare memory (Locate this before loading the assembler, and source code).**

The important thing in which you must also do is set up the **START ADDRESS** where spare memory is available for your SEUCK game. The easiest way to find which memory you need to use is to load in your game with STM title screen included. Then use the **machine code monitor** to **scan** for **zero bytes** or large chunks of identical bytes that are identified as junk) and then **scroll** through the memory to see how much spare memory there is. I normally use the **I*** prompt on the Action Replay MKVI freezer M/C monitor.



```
:9420 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9440 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9460 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9480 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:94A0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:94C0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:94E0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9500 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9520 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9540 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9560 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9580 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:95A0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:95C0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:95E0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9600 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9620 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9640 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9660 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:9680 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:96A0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:96C0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
:96E0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
      @ = ZERO BYTE (FREE MEMORY)
```

**Installing the enhancements**

Installation of the enhancements is split into three parts.

1. Load and run your SEUCK game creation.
2. Enter the machine code monitor (Action Replay/Retro Replay) by pressing F8 in fast load) and then enter `L "TITLESCREEN*",8`. To load the new title screen.
3. Load in your enhancement using `L "nameofenhancement",8,(STARTADDR)` (Where the

If you want to add another two in game enhancements to your game, repeat the same procedure as above, but make sure correct parameters have been set before assembling the source. For example:

**enhancement 1: PARAM1INIT/PARAM1PLAY**
**enhancement 2: PARAM1INIT/PARAM1PLAY**
**enhancement 3: PARAM3INIT/PARAM3PLAY**

After you have installed your enhancements use **G 6580** in your cartridge/VICE monitor to test your production. If all is working great and well, then save the finished production as you did before in the STM manual and then compress the production as an executable.

***Do not get the Parameters Init and Play code mixed otherwise your game is likely to not work properly.***

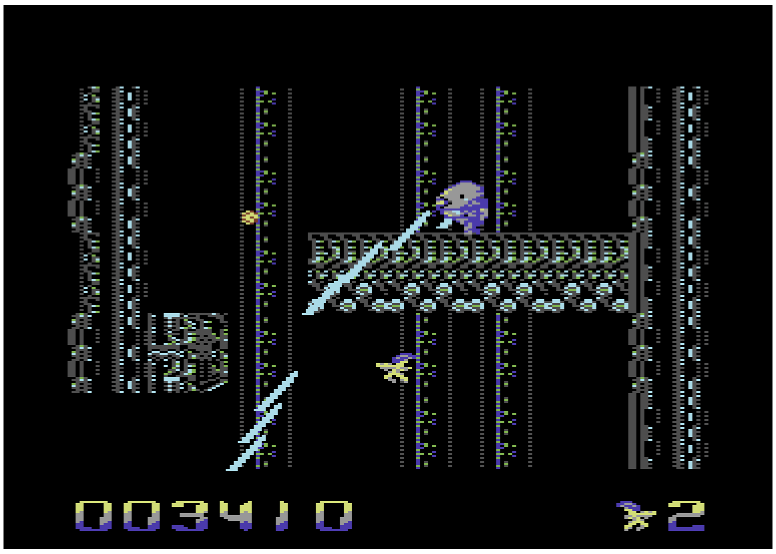
Okay. That's enough instructions on how to set up the parameters and install the enhancements. Here are the enhancements that are on offer for you all.

## SCORE PANEL (1 Player)

**by Richard Bayliss**

*File: scorepanel1p.tas*

*MEMORY USED: $00EA bytes*



**Example: Yauzeras by Gibranx**

This source code creates a custom score panel for SEUCK and Sideways scrolling SEUCK games. The score will consist of 10 sprite frames to represent numbers 0 to 9 and 8 hardware sprites are used in the score panel.

***Beware: The size of the score sprites might cause some flickering above the panel. If you want to try and reduce this, simply make the sprite height size smaller. Also try to limit the number of sprites at a time on screen. SEUCK is known to flicker the panel sprites above the score panel if multiple sprites reach the lower border where the panel lies.***

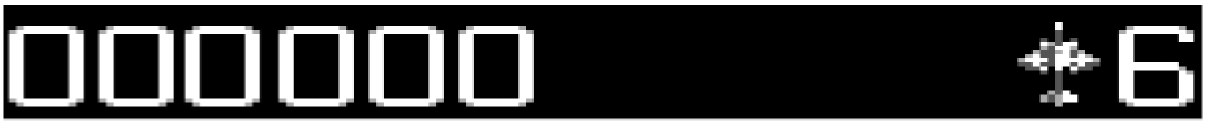
You can edit the sprite multicolour mode binary value settings. They are labelled as **SPRMCOLMODE**.
The values represent the sprite multicolour that can be on / off.

**%11111111** represents all sprite using multicolour mode. Where as **%00000000** represents all sprites using hi res colour mode instead.

**SPRITENUM** represents the **sprite number** in **SEUCK**. This is used as a **LIVES** object in the score panel. For example, the **player** sprite.

**SPRITECOL** represents the sprite colour of the whole score panel. Values can be between 0 to 15 according to the C64 colour palette.

By default the s**prite panel** layout will look something like this:



The score in digits are based at the left of the panel and the lives indicator is based on the very right of the score panel. Although Stormbird uses this layout, it is **possible** to move the sprites to another **X** position, by changing the values of the X/Y position table near the very bottom of the code snippet. $07 must remain as the Y position in the table (Like in this example below).

```
;Position table for the score and
;lives sprites.

postbl
         .byte $18,$07,$30,$07
         .byte $47,$07,$60,$07
         .byte $78,$07,$90,$07
         .byte $27,$07,$40,$07
```

Another important thing to take note of is that the 2 sprites at the far right of the panel above use $D010 X MSB positioning. The C64's sprite position by default is usually restricted to 256 pixels, but MSB doubles the pixel positioning for the sprites. The score panel uses sprites 0-5 and the lives sprite uses sprites 6 and 7. If you take a look at $D010 inside the code snippet, it has the sprite value set where sprites 0-5 use no MSB expansion and 6 and 7 uses MSB. $D010 value is set to %00000011. You can alter the MSB value of the sprites at $4562 by adding inside the initialise code:

```
lda #%00000011 ;(Or other binary values)
sta $4562
```

After modifying the source code, make sure you save it using ← W or ← S and in Turbo Assembler/Turbo Macro Pro Assembler.

Remember to set a valid start address for installing the code. Use ← and 5 to assemble the source code to your project work disk .

***After assembling the source***

You need will 10 sprites drawn to represent a score panel. This can be done using any native C64 sprite editor or SpritePad (With sprites exported to PRG format). If using multi-colour, the sprite multi-colour must match the two colours of which the SEUCK game uses. ($D025, $D026). The sprites must form of digits 0 to 9 in order.

As an example, the scoring sprites should look something like this:

***Loading the custom sprite panel***

The sprite data needs to be loaded at $f080, because this will of course replace the old SEUCK score panel with the new one. Using Action Replay or Retro Replay machine code monitor, enter the following command:

```
L "PANELSPRITES",8,f080
```

Load in the assembled object file (as an example SCOREPANEL1P.OBJ). Using

```
L "SCOREPANEL1P.OBJ",8
```

… and finally install the score panel (which you loaded to the set load address) and type in

```
G ($load address) (Example: Stormbird uses G 9400 to install the new panel)
```

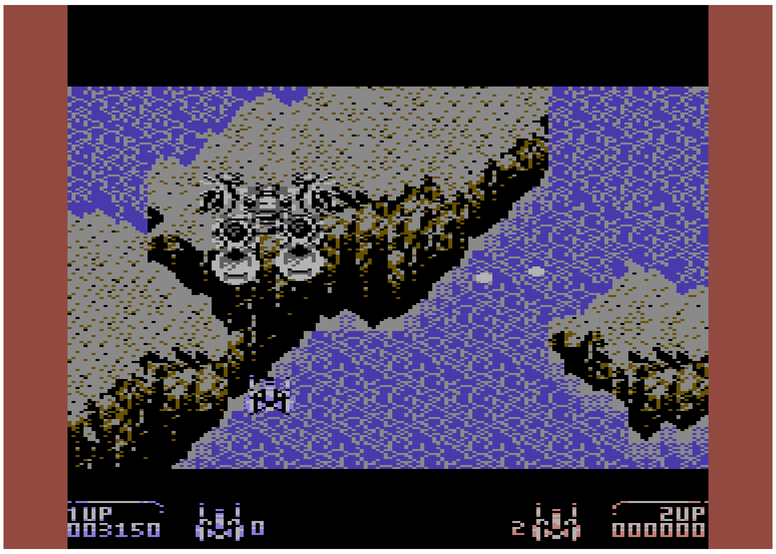
For example if your source was set to location $3402, G 3402 is used to install and configure the enhancement.

## SCORE PANEL (2 Player)

**by Richard Bayliss**

*File: scorepanel2p.tas*

*MEMORY USED: $026A bytes*



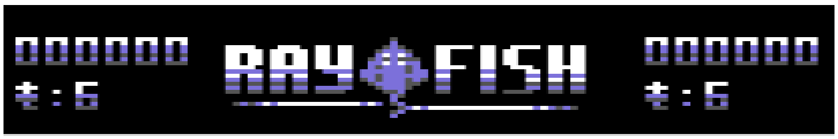
**Example: VIOS by Carl Mason**

The previous new score panel code was made for 1 player. Now here's another great custom score panel code snippet. This time it is for SEUCK games that are suitable for 2 player games. If you remember Ray Fish and Mega Tank Blasta, you'll definitely will want to try and make cool new score panels for your games.

**Warning:** ***Just like with the 1 player sprite panel, you might get flickering above the panel and the new panel code might also cause slow down if too many sprites are on screen.***

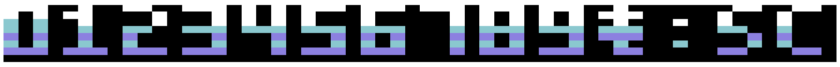
**The score panel**

The score panel for 2 player games will look something like this:



**The graphics type used**

This type of score panel uses two different C64 graphic files. The main panel decoration consists of **8** sprites. The



score and lives indicators consist of a **1x1 charset** that is **embedded** inside the **sprites**. The charset should very small and be set in the following order:
Just like with the sprite score panel. The sprite data must be loaded at **$F080** and this time use **eight** sprites instead of 10. The **charset**, however can be set to a desired address where free memory is available.

There isn't much in which you need to edit inside the source code, but let me give you some ideas about everything:

**Sprite Colour**

The labelled variable **sprcol** will set the colour of the whole score panel (The changeable sprite colour). Other colours (multi colour 1 and multi colour 2 must match the colour of the game sprites – otherwise multicolour mode should be set to hi-res mode for just one colour). Values 0-15 ($00-$0F) can be used to represent the desired colour from the C64's palette.

**Hires/Multicolour Mode**

The sprite **hires/multi colour** mode can be set by using binary values in the variable: **mcolmode**. It is possible to mix hi-resolution and multi-colour sprites if say you wanted a multi-colour logo as decoration and the blank areas use a hi-resolution text for score and lives indicators.

**Setting addresses of the score character set**

The variable **scorechar** should be set to the correct address in which you intend to load in the character set for your custom sprite panel. For example **scorechar = $0800** points at the address $0800 for the score panel charset.

**Embedding the position of the characters within the 8 sprites**

If you want to place custom scores in different sprite positions. It is possible to do exactly that by changing the value of the sprite position for the charset plotting. These are indicated with:

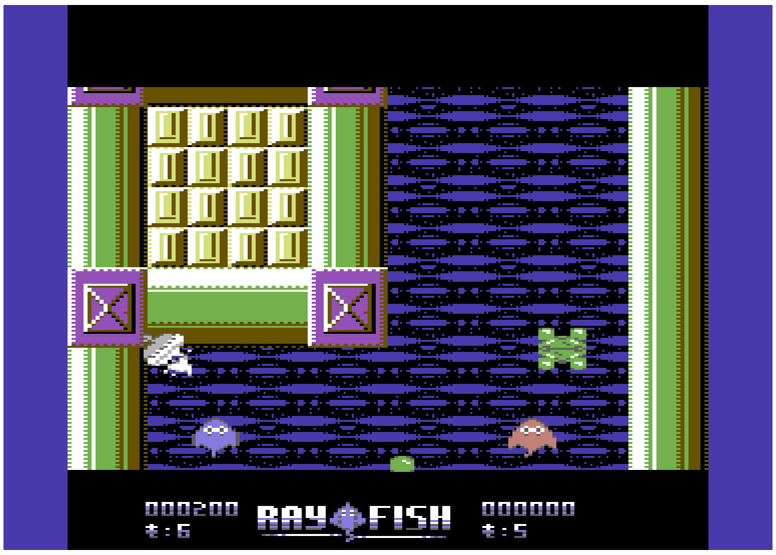
```
spplayer1 = $f080
spplayer2 = $f200
```

**How to add the enhancement**

As previously mentioned. After setting the SEUCK player parameters. Press ← and 5 to assemble your modified source code to disk (remember to save the source listing first, because you will need it in future).

Load in your SEUCK game as normal, and install the title screen. Now do as follows

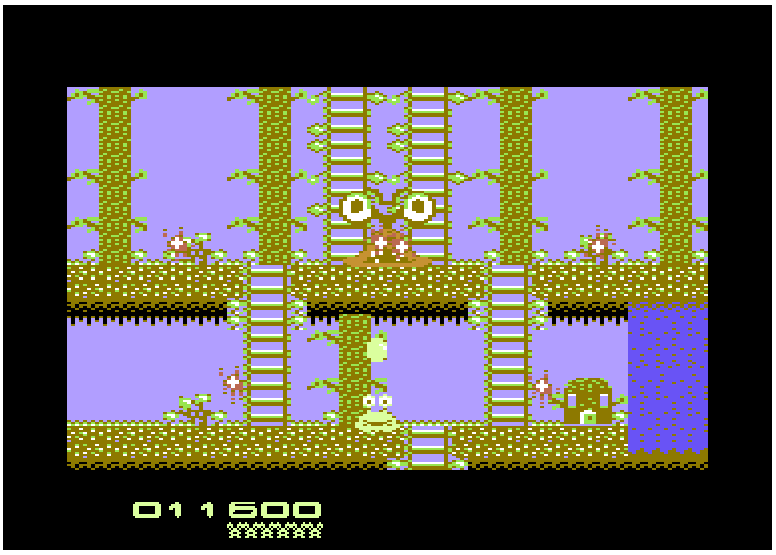
Load in your 8 sprites to **$F080** (`L "SPRITEDATA",8,f080`)
Load in your mini charset to your chosen free memory address (For example: `L "CHARDATA",8,0800`)
Load in the enhancement object code (For example: `L "OBJECTCODE",8,9400`)
Execute the object code.
Execute the title screen using `G 6580`
Start the game and you should see a new score panel, similar to this example below (Rayfish)



## INSTANT CHAR SCROLLING BACKGROUND ANIMATION

**by Richard Bayliss**
*File: charscroller.tas*
*MEMORY USED: $00B4*

The next enhancement is something which I love to add in SEUCK games. This is because these type of games lack that type of feature. **Single char scrolling background animation** is a perfect feature for creating simple fun effects like **water flow** (Rivers, waves or even waterfalls like Squibblies 3 used – pictured below), **lava,** a **1x1 scrolling void,** etc. The source on disk allows you to pick **4** characters in the SEUCK background in which you would like to roll. The scrolling can be either **up**, **down, left** or right. It is also possible to mix the character scrolling so that you can use **diagonal movement** for single character scrolling.



**Setting up which SEUCK background characters to scroll**

The background animation routine features a choice of four characters in which can be picked to be animated. The variables are declared as below:

```
char1 = 1 ← Char ID: 001 is assigned as animated character 1
char2 = 2 ← Char ID: 002 is assigned as animated character 2
char3 = 3 ← Char ID: 003 is assigned as animated character 3
char4 = 4 ← Char ID: 004 is assigned as animated character 4
```

By default, char1 is character 001 in SEUCK. If you change the values of the characters for scrolling as the 4 labels. They can be any character chosen in the background character editor in SEUCK. Let us take for example, you make a game where characters 027, 032, 076, 094 are to be scrolling character objects. We simply re-label the values of **char1, char2, char3** and **char4** as the example below:

```
char1 = 27 ← Char ID 027 is assigned as animated character 1
char2 = 32 ← Char ID 032 is assigned as animated character 2
char3 = 76 ← Char ID 076 is assigned as animated character 3
char4 = 94 ← Char ID 094 is assigned as animated character 4
```

**Picking the speed of the scrolling characters (best used with vertical scrolling only)**

It is possible to also set whether or not you would like a slow or a fast scrolling speed of the characters in which you wish to scroll. This can be done by setting the values with the labels below:

```
speed1 = 2
speed2 = 2
speed3 = 2
speed4 = 2
```

Hint: (2 is the set speed for speed 1. This is the delay rate. 1 is the fastest speed. The higher the delay value (speed1, etc) is set, the slower the speed of the scroll will become. This will only work on up and down scrolling because left and right won't look right with delay put in place.

**Picking the direction of the char scroll**

If you want your background character to scroll another direction other than the set direction. You can comment the code listings with the directions you do not wish to scroll the background (Using ; before the code listing) and remove the comment for the listing in which you wish to use for scrolling. Or alternatively, you can remove the whole listing for the other scrolling code.

For example. Should you wish to scroll the chosen character (**char1**) right. You would need to put ; next to the whole code which reads the code that make the char scroll up, down and left. Then remove ; next to the whole listing that scrolls the character to the right. It should look something like this example below.:

```
scroll1

;Comment, or delete the unwanted scroll direction

;Scrolling direction up

dirup1
;         lda bgdelay1
;         cmp #speed1
;        beq doscroll1a
;         inc bgdelay1
;         rts
;doscroll1a
;         lda #0
;         sta bgdelay1
;         lda chars+(bgchar1*8)
;         sta temp1
;         ldx #0
;scrup1
;         lda chars+(bgchar1*8)+1,x
;         sta chars+(bgchar1*8),x
;         inx
;         cpx #8
;         bne scrup1
;         lda temp1
;         sta chars+(bgchar1*8)+7
;skip1a
;         rts

;Scrolling direction down

;dirdown1
;         lda bgdelay1
;         cmp #speed1
;         beq doscroll1b
;         inc bgdelay1
;         rts
;doscroll1b
;         lda #0
;         sta bgdelay1
;         lda chars+(bgchar1*8)+7
;         sta temp1
;         ldx #7
;scrdown1
;         lda chars+(bgchar1*8)-1,x
;         sta chars+(bgchar1*8),x
;         dex
;         bpl scrdown1
;         lda temp1
;         sta chars+(bgchar1*8)
;         rts

;Scrolling direction left

;dirleft1
;         ldx #$00
;scrleft1
;         lda chars+(bgchar1*8),x
;         asl
;         rol chars+(bgchar1*8),x
;         inx
;         cpx #8
;         bne scrleft1
;         rts

dirright1
         ldx #$00
scrright1
         lda chars+(bgchar1*8),x
         lsr
         ror chars+(bgchar1*8),x
         inx
         cpx #8
         bne scrright1
         rts
```

**Diagonal char scrolling**

It is also possible to combine two directions to make instant diagonal scrolling of your character in which you wish to animate. This can be done by simply combining two directions of the scrolling. (X and Y). Below is an example on how to set a diagonal scroll (down and right)

```
;Comment, or delete the unwanted scroll direction

;Scrolling direction up

dirup1
;         lda bgdelay1
;         cmp #speed1
;        beq doscroll1a
;         inc bgdelay1
;         rts
;doscroll1a
;         lda #0
;         sta bgdelay1
;         lda chars+(bgchar1*8)
;         sta temp1
;         ldx #0
;scrup1
;         lda chars+(bgchar1*8)+1,x
;         sta chars+(bgchar1*8),x
;         inx
;         cpx #8
;         bne scrup1
;         lda temp1
;         sta chars+(bgchar1*8)+7
```

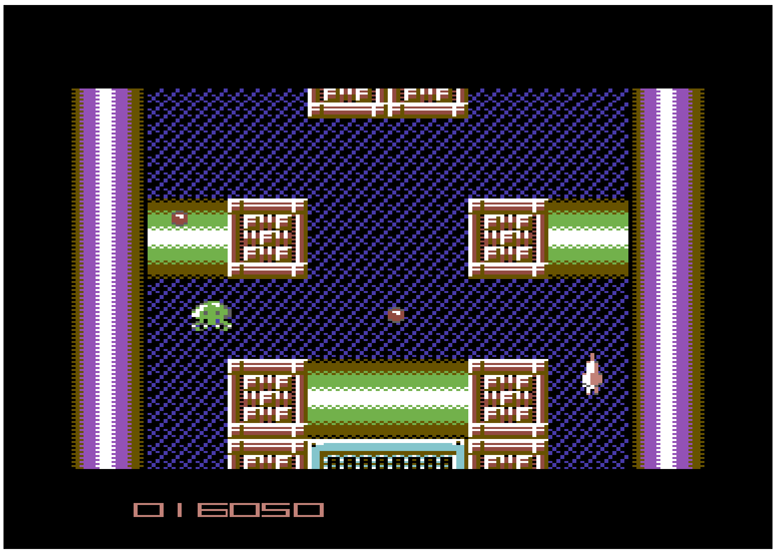
```
;skip1a
;         rts

;Scrolling direction down

dirdown1
        lda bgdelay1
        cmp #speed1
        beq doscroll1b
        inc bgdelay1
        rts
doscroll1b
        lda #0
        sta bgdelay1
        lda chars+(bgchar1*8)+7
        sta temp1
        ldx #7
scrdown1
        lda chars+(bgchar1*8)-1,x
        sta chars+(bgchar1*8),x
        dex
        bpl scrdown1
        lda temp1
        sta chars+(bgchar1*8)
        ;rts ← RTS must be disabled in first char and mix with second

;Scrolling direction left

;dirleft1
;         ldx #$00
;scrleft1
;         lda chars+(bgchar1*8),x
;         asl
;         rol chars+(bgchar1*8),x
;         inx
;         cpx #8
;         bne scrleft1
;         rts

dirright1
        ldx #$00
scrright1
        lda chars+(bgchar1*8),x
        lsr
        ror chars+(bgchar1*8),x
        inx
        cpx #8
        bne scrright1
        rts
```

## SCROLL/PUSH PARALLAX CHAR SCROLLING (VERTICAL or HORIZONTAL)

**by Eleanor Burns and Richard Bayliss**

*File: parallaxv.tas / parallaxh.tas*

*MEMORY USED: $0088 / $0089*



**Example: Rocket and Roll** used a parallax scrolling effect

This routine performs a vertical charset up scrolling parallax subroutine. It is suited for games that use either an up scroller or instant vertical scrolling. The purpose of this effect is to give a cool static effect to inner background (Like Hades Nebula game – although that is NOT SEUCK). The effect can also be useful for giving a 3D visual from a birds eye view. There is also a special version for Horizontal scrolling SEUCK as well as standard SEUCK games. Example: Although not featured in the SEUCK Title Screen Maker. The static void feature was used in the SEUCK Game, Rocket 'N Roll (See SEUCK School: Invincibility Cloak at: https://tnd64.unikat.sk/SEUCK_School.html#Invincibility).

**Setting up which SEUCK background characters to scroll**

Like with the previous background animation routines, you can use up to 4 characters. Simply set their values using **char1, char2, char3, char4** variables. (See **Picking the SEUCK Chars to Scroll** to set these in the previous category).

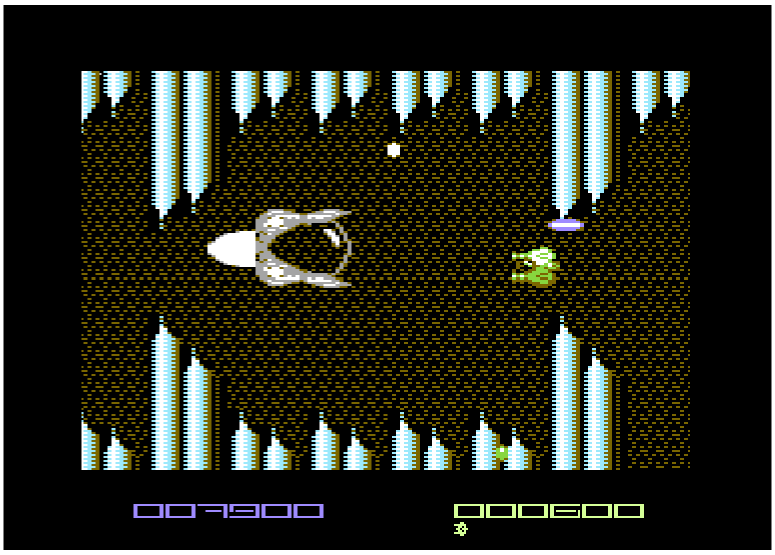
## CHANGING BACKGROUND COLOUR ACCORDING TO LEVEL

**by Richard Bayliss**

*File: levelcolour.tas*

*MEMORY USED: $00D0*

This next routine is yet another special and very handy enhancement. It allows you to master the colour scheme for your level parameters from SEUCK. SEUCK games use 22 level parameters, which always use the same background colour for every level. There may be times in which you wish to change the colour scheme for each level set in the parameters. The three colours are based on chart **background colour, multicolour 1, and multicolour 2.**



Example: Guillotine – The Doom Machine (created using Sideways Scrolling SEUCK) used level background colour scheme changes.

It is always important to take note of the the position of your game map by taking a look at the level parameters. Then take note of which background colour scheme you feel would suit your own level. Once you feel happy about your colour scheme. Scroll all the way near to the bottom in the assembly source code and change the byte values (according to level parameters) of which colours you wish to change the background for your game. Each .byte on each table has a label beside it with the colour that is being used. For example if you wish to set level 1's colour scheme where background colour is green, multi-colour 1 is light green and multi-colour 2 white. The value would look something like this:

```
bgcoltbl
        .byte 5 ; (or $05) - Level 1
bgcoltb2
        .byte 13; (or $0d) - Level 1
bgcoltb3
        .byte 1; (or $01) - Level 1
```

If you wish to edit a later level, say for example LEVEL 7 to make a new colour scheme, then it can be found at as the seventh byte of the table. Simply change the colour value there.

***Hint: It might be possible to also switch between in game tunes according to levels. For example switching from a game tune, to end of level boss tune. However this will require one time trigger pointers, and an additional table to initialise music from another track of the same tune, although this source code has not provided it, it is definitely possible to do exactly that!***

## DESTROYING ALL ENEMIES ON SCREEN
(set by default for Stormbird)

**by Jon Wells**

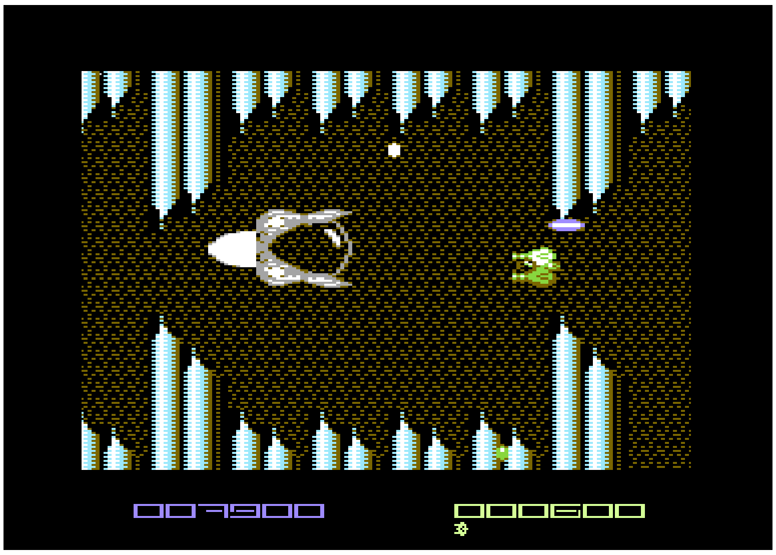
*File: smartbomb.tas*

*MEMORY USED: $0149*

Have you written a SEUCK game in which features an end of level boss. Are you fed up with shooting parts of the boss enemy one by one? Do you wish the boss enemy can explode or do you want to make a collectable object make all existing enemies explode? Well, this special integrated routine installs a feature in which you can destroy all of the enemy boss objects in one go or make instant smart bomb power ups.

***Warning: If you want to destroy linked enemies (IE Bosses), make sure there are no other enemy sprites on***

*screen, otherwise they will also explode on screen.*



*By using this enhancement you can destroy large enemies in one go. Give 'em several shots and wave bye, bye to those bad guys. Just like that evil alien ship featured in Guillotine – The Doom Machine*

Not only can you just destroy boss enemies in one go, you can also make collectable objects work like smart bombs. Should you wish to include this kind of feature in your game.

**Setting up the feature**

The example which is currently available for the exploding boss/smart bomb feature is already configured for Stormbird. However, it is possible to configure the explosion feature to suit your very own game creation. The source code has a list of variables called **enemyobj** with a value of the **object number** set to that variable. It is possible to make more or less enemy objects. The variables look something like this:

```
;chain reaction on other existing
;enemies when hit).

enemyobj1 = 22 ;Large plane part 1
enemyobj2 = 23 ;Large plane part 2
enemyobj3 = 24 ;Large plane part 3
enemyobj4 = 25 ;Large plane part 4
enemyobj5 = 26 ;Large plane part 5
enemyobj6 = 27 ;Large plane part 6
enemyobj7 = 28 ;Large plane part 7
```

*.. and so on*

The code below shows an example of detecting the object to call the full enemy/smart bomb based explosion inside your game. It looks something like this:

```
;Detect object type, and ensure it is
;the enemy enemy objects or an object
;that can create a smart bomb



objdetect

         lda $bc00,y
         sta objecthit
```

```
          lda $09
          sta playerscored
          lda $bc06,y
          sta scoretype

;OBJECT HIT CHECKER.

;Check object ID hit

          ;Check enemy ID 1
          lda objecthit
          cmp #enemyobj1
          bne notbomb1
          jmp trigger

          ;Check enemy ID 2
notbomb1
          cmp #enemyobj2
          bne notbomb2
          jmp trigger

          ;Check enemy ID 3
notbomb2
          cmp #enemyobj3
          bne notbomb3
          jmp trigger

          ;Check enemy ID 4
notbomb3
          cmp #enemyobj4
          bne notbomb4
          jmp trigger

notbomb4 … and so on.
```

The code and object identification will need to be set to your very own SEUCK game creation. If you do not want to check 23 enemy objects. You can comment out the other checks (after the label) using the ; symbol before each command. Make sure the command no more is enabled.

**Example: Using 4 objects to trigger a smart bomb:**

At the start of the listing, edit the first four variables to set as object 42, 43, 44, 45 (Actual enemy 21, 22, 23, 24)

```
;Set four enemy objects for smart bomb feature

enemyobj1 = 21 ;Assign actual enemy 21
enemyobj2 = 22 ;Assign actual enemy 22
enemyobj3 = 23 ;Assign actual enemy 23
enemyobj4 = 24 ;Assign actual enemy 24
```

Now go straight to the code underneath the comment **;Check ID Hit**. Make sure the checks for enemy object 1, object 2, 3 and 4 are available like as below:

```
;Check object ID hit

          ;Check enemy ID 1
          lda objecthit
          cmp #enemyobj1
          bne notbomb1
          jmp trigger

          ;Check enemy ID 2
notbomb1
          cmp #enemyobj2
          bne notbomb2
          jmp trigger

          ;Check enemy ID 3
notbomb2
          cmp #enemyobj3
          bne notbomb3
          jmp trigger

          ;Check enemy ID 4
notbomb3
          cmp #enemyobj4
          bne notbomb4
          jmp trigger

          ;Check enemy ID 5
notbomb4
          cmp #enemyobj5
          bne notbomb5
          jmp trigger
```

Change what is highlighted in red on this page to bne nomore. Then add a ; before the rest of the check routine – or delete it all the way until you reach the command nomore (Make sure you do not add ; or delete the nomore command). Just like the example below

```
;notbomb5
          ;cmp #enemyobj6
          ;bne notbomb6
          ;jmp trigger

;notbomb6 … and so on
          . . .
nomore
          lda $bd06,y
          rts
```

## POWER UPS (Including invincibility, bullet upgrades and smart bomb feature)

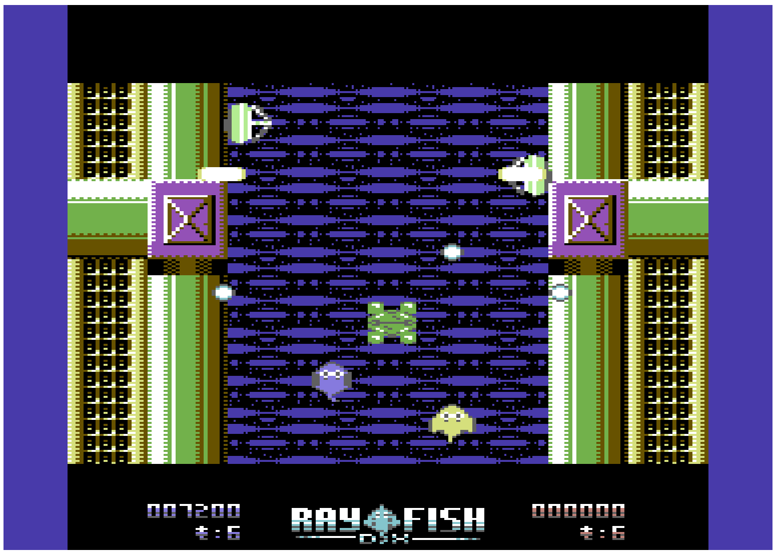
(Set by default for Ray Fish)

**by Jon Wells and Richard Bayliss**

*File: powerups.tas*

*MEMORY USED: $04DB*

SEUCK games have always lacked special features such as power ups. This code listing is quite big and will require about $0500 bytes free. This is because there have been a lot of cool features implemented into the code. Although the power ups code features the smart bomb feature in the previous description. This piece of code features an even better power up system. It allows the player to kill specific enemy objects in SEUCK and upgrades the player's firepower. As an extra bonus, it even upgrades the bullet by changing the sprite ID. Make sure that you have spare sprites available to implement the power ups feature.



***Rayfish DX (SEUCK Redux) used this special feature in SEUCK Redux which can now be implemented into your game projects without SEUCK Redux.***

Another feature which is implemented in the power ups routine is a temporary shield for the player.

**Setting up the power ups**

You will need to assign the objects in which you wish to set to trigger specific power ups or smart bomb explosion. There are two variables in which can be set up for random power ups, and a few more variables to be set as smart bombs/full boss explosion. They look something like this (Based on the Rayfish DX objects)

```
;Define object number as power up

powerupid = 57 ;Object ID

;Shield is shared with power up ID

shieldid = 44 ;Object ID

;Define OBJECT numbers as boss
;objects.

bossid1  = 42
bossid2  = 45
bossid3  = 46
bossid4  = 47
bossid5  = 48
bossid6  = 49
bossid7  = 50
bossid8  = 52
bossid9  = 53
bossid10 = 56
```

**Defining the shield power up**

As well as modifying the variable **shieldid** to the object ID that represents the enemy type. There are also a few important parameters in which must be set to the player. Should you wish the player to be able to move over the background while the shield power up (or player re-spawn) is active. The sprite to background collision value should be set by altering the two variables below.

```
;SPRITE TO CHARSET/BACKGROUND
;COLLISION

;Define COLLISION CHAR VALUE for
;PLAYER 1 - SET TO STOP/DIE

collcharp1 = 60

;Define COLLISION CHAR VALUE for
;PLAYER 2 set to stop/die

collcharp2 = 60
```

Also, you should set the maximum value of the player to background char collision in shield mode. If you want the player to NOT allow the player to move over the background in shield mode, simply set it to the value of the defined collision char value.

```
maxccharp1 = 255
maxccharp2 = 255
```

**Setting the default colour of the player after shield mode**

The player colour default can be set to variables **colourp1 and colourp2** (Player 1 colour and player 2 colour). However, it is not all that simple to just assign a colour to the player. This is because the player uses different animation type values.

**$00 to $0f –** represents colour value + animation type **01 – 18**
**$e0 to $ef –** represents colour value + animation type **directional**
**$f0 to $ff –** represents colour value + animation type **directional hold**

So, let us say for example you want to have a space ship painted in light blue for player 1 and player 2 is painted in light red (pink), but both space ships can tilt when you move the ship a specific direction. Then after letting go of the control of the ship it restores to its central state. The variable for **colourp1** or **colourp2** will be:

```
colourp1 = $fe
colourp2 = $fa
```

**Setting up the player firepower variables**

The power ups code snippet also allows the player to update at random the bullets. There are some variables in which can be assigned to how many bullets each player can fire at a time. The player will be able to assign a minimum of 1 bullet at a time (set as value 0) all the way to 3 bullets at a time (set as value 2). The variables look something like this below:

```
;PLAYER 1

p1numbulls1 = 0 ; Default
p1numbulls2 = 0 ; Power up 1
p1numbulls3 = 1 ; Power up 2
p1numbulls4 = 2 ; Power up 3
p1numbulls5 = 2 ; Power up 4

;PLAYER 2

p2numbulls1 = 0 ; Default
p2numbulls2 = 0 ; Power up 1
p2numbulls3 = 1 ; Power up 2
```

```
p2numbulls4 = 2 ; Power up 3
p2numbulls5 = 2 ; Power up 4
```

It is not only the number of bullets (p1numbulls1 to 4, and p1numbulls1 to 4) in which can be assigned as bullet power ups. You can also assign whether or not you would like a power up or default bullet to use directional fire or not. This can be done by setting the variable **p1dirfire1 to 4, and p2dirfire1 to 4** to 0 (disabled) or 1 (enabled).

Finally the distance of firing (fire rate) can be edited to suit a specific power up. Let us say for example you are making a **Spy** type of game. Your player by default is armed with a short distance rifle, and then you pick up a more powerful weapon. The fire rate can be increased to a higher value by setting up the bullet duration.

```
p1rate1 = 99
p1rate2 = 99
p1rate3 = 99
p1rate4 = 99

p2rate1 = 99
p2rate2 = 99
p2rate3 = 99
p2rate4 = 99
```

Simply change the values to the above variables to setup the distance firing.

**Changing the sprites of the power ups**

Near the very bottom of the code are .byte data tables. These are all set to represent the sprite table to be used for when a new player bullet has been assigned on collecting the power up collectable. Each table consists of 18 bytes. The 18 bytes represent the order list of the power up sprites. (Hint: To get the correct values for bullets which use directional/directional hold – set anim type to 18 and then follow the order table there).

The power up sprite type table look something like this:

;Default bullet object - PLAYER 1

```
;Note that values in bytes 085 = 85
;                          001 = 1
; ... etc

pow10
         .byte 85,85,85,85,85,85
         .byte 85,85,85,85,85,85
         .byte 85,85,85,85,85,85

;Power up bullet 1 - PLAYER 1

pow11    .byte 52,52,52,52,52,52
         .byte 52,52,52,52,52,52
         .byte 52,52,52,52,52,52

;Power up bullet 2 - PLAYER 1

pow12    .byte 53,53,53,53,53,53
         .byte 53,53,53,53,53,53
         .byte 53,53,53,53,53,53

;Power up bullet 3 - PLAYER 1

pow13    .byte 86,72,87,74,72,75
```

```
          .byte 88,73,89,86,72,87
          .byte 74,72,75,88,73,89
```

```
… and so on!
```

**Boss explosions / smart bomb feature (again)**

The feature exists in the previous sub chapter, however there is one extra feature this power up code uses. There is a background explosion table. This can be set to create a cool explosion effect. There is a colour table already in the code. However, the variables for restoring the background colour must be set. This can be done by setting variables **bgcolour** (background colour), **bgmcol1** (background multi-colour) and **bgmcol2** (background multicolour 2).

After assembling to disk and assigning it to your STMV1.6 game project, you'll be able to enjoy your game creation with weapon upgrades, exploding bosses/smart bombs and flashing exploding background colour. How awesome is that?

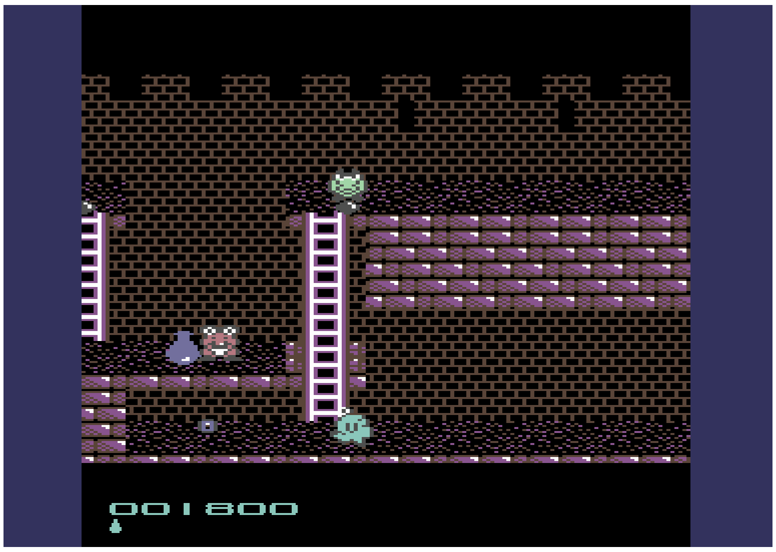
## SAFE PLAYER RESPAWN

**by Richard Bayliss**

*File: respawn.tas*

*MEMORY USED: $0078*

Do you have a problem with push scrolling SEUCK games where after the player dies, it re-spawns on a piece on stop/die assigned background characters and causes the player to get stuck? The thing the gamer least wants in their game is where player is unmovable after it spawns from its death. To our personal opinion, it is classed as an unfair spawn position. So this code snippet performs a safe spawn of the player. After the player dies, it will re-appear on exactly the same spot from where it last died. This is best used with push scroll games.



***Imaginator was one game in which had lacked safe spawn. It would have made sense to spawn the player at exactly the same position it last died.***

**Setting up the routine**

There is not much that needs to be done here. Set three variables for player 1 X, player 1 XSMB and player 1 Y start position. Also do the same for player 2 if the game is also for 2 players. You might want to load in your SEUCK game first and use the Machine Code Monitor on your cartridge to search for the byte values inside SEUCK player start positions. For example use **M 40A3**, and note down the first three bytes for player 1. Then use **M 40B6** and note down the first three bytes for player 2.

```
player1x = $a8 ;Look at $40a3
player1xmsb = $00 ; Look at $40a4
```

```
player1y = $98 ; Look at $40a5

player2x = $c8 ;Look at $40b6
player2xmsb = $00 ;Look at $40b7
player2y = $b4 ;Look at $40b8
```

After assembling and implementing into your SEUCK game, featuring the STM. You will be able to enjoy a hassle free push scrolling game and the player will **no longer** get stuck on stoppable background after it spawns again.

## STABLE SIDEWAYS SEUCK SCROLLING

**by Richard Bayliss**

*File: stablesseuck.tas*

*MEMORY USED: $002C*

*Example: Delta Run was one game to have used the stable scrolling with Sideways scrolling SEUCK*

The next enhancement is used for stabilising the background scrolling more in games that have been created using Sideways SEUCK. It does not take much memory, and will not be called inside the main initialize and play routines in the SEUCK title screen maker. Instead, it fixes the bug inside sideways scrolling SEUCK games.

What this routine does is makes the horizontal scrolling SEUCK game feel much more stable. It covers the top screen flicker and covers more of it in black. Then afterwards sets the background colour set to your game (Which you must assign using the variable **bgcolour** otherwise you might get an incorrect background colour).

After assembling and setting up with a Sideways SEUCK game with a new front end made using the SEUCK Title Screen Maker, you can enjoy your creations more without a big flicker inside your scrolling scenery or throughout the game.

***Note: By activating this feature, enemies and player will display 1 or 2 pixels lower than previously.***

## FORCE 1 PLAYER AT A TIME TO SPAWN IN 2 PLAYER MODE

**by Richard Bayliss**

*File: force1player.tas*

*MEMORY USED: $006E*

The SEUCK Title Screen Maker has a function that spawns both player 1 and player 2 on screen at the same time when starting the game with player 2's fire button. This enhancement will **prevent** player 1 from spawning directly when player 2 starts the game. It triggers a quick time function that will allow player 2 to spawn alone, if fire in joystick port 1 has been pressed to start the game. Of course, during play, player 1 can press fire and join in at any time. One player mode also will allow player 2 to spawn at any time during play.

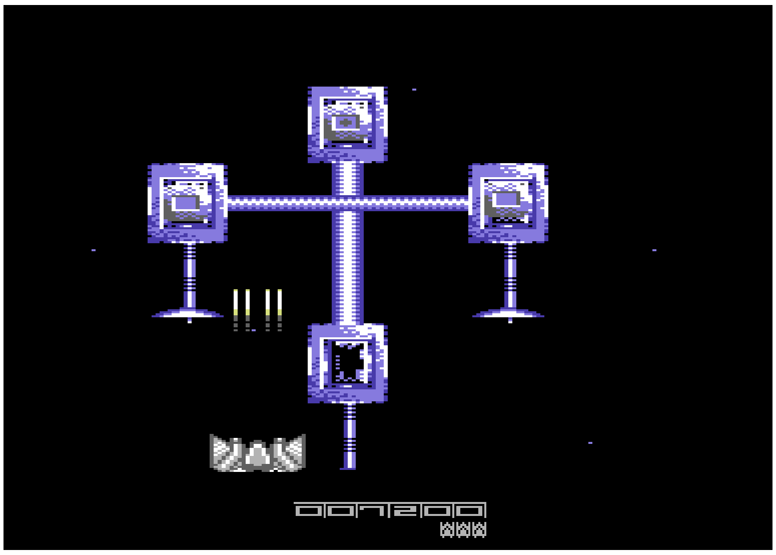
***Note: The enhancement should only be triggered if you enabled 2 player mode in the SEUCK Title Screen Maker settings menu.***

## LINKED PLAYERS AND SCORING

**by Richard Bayliss**

*File: linkplayers.tas*

*MEMORY USED: $018A*



***Example: Dark Force –** a SEUCK Light-Force inspired game to use linked players and shared scoring and lives*

Have you ever wanted to make a game where you wanted to have a bigger player, joined together. For example, by creating a beat 'em up or perhaps make a game that involves a bigger space ship just the way I did with Dark Force?

Well, this code snippet bolts two players together and shares the score and lives. It also links the death together should enemies destroy the player. With this cool enhancement, it is possible to make games like Lightforce or Narc using SEUCK. To use  this enhancement you need at least $0200 bytes available. When you use this feature. Your SEUCK game must DISABLE extra lives. Otherwise things will not work out that way. If you want to include extra lives, you will need to create an object that will allow that and use the enemy detection trick.

There are just a few values in which needs to be set in order for this to take effect more accurately.

You need to set the starting X, X MSB and Y position of the player spawning at the start of the game. These are set as variables:

**defaultx** = Default horizontal start position for the player
**defaultxmsb** = Default horizontal start position expanded sprite position
**defaulty** = Default vertical start position for the player

Also, when player 2 is linked to player 1 you should also set the X and Y offset values. This is in order to give a more accurate linking of player 2 with player 1. This depends on the type of game in which you are writing. The values are as follows:

**xoffset** = The calculation of pixels horizontally of where player 2 should fit next to player 1
**yoffset** = The calculation of pixels vertically of where player 2 should fit next to player 1

Finally there is the background collision settings and properties that should be set onto both players. Linked explosion triggers the deadly background to die mode on the player, and then restores back to background collision characters and stop/die assigned pointers. These are as follows:

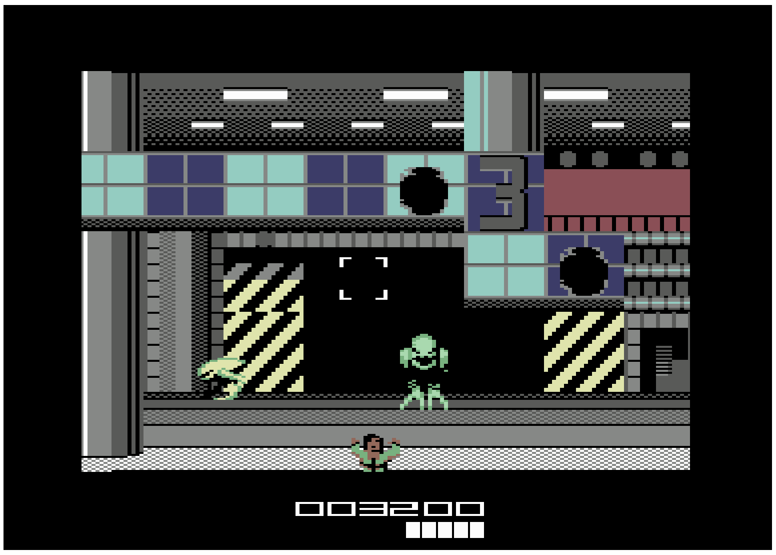
**bgchar** = The background collision char value, which the player should stop/die during play. Values can be between 1 and 255

**stopdie** = The effect which takes place after the player collides into that background object.

***Note: If you are linking the linked players mode to the custom score panel (scorepanel1p.tas). You must change the address being read ($5ea3) to read the address of where you have placed the custom score panel (store). Also extra lives at 10,000 should be disabled.***

## TWIN MODE WITH SHARED SCORING AND LIVES

**by Richard Bayliss**
*File: twinmode.tas*
*MEMORY USED: $00F6*



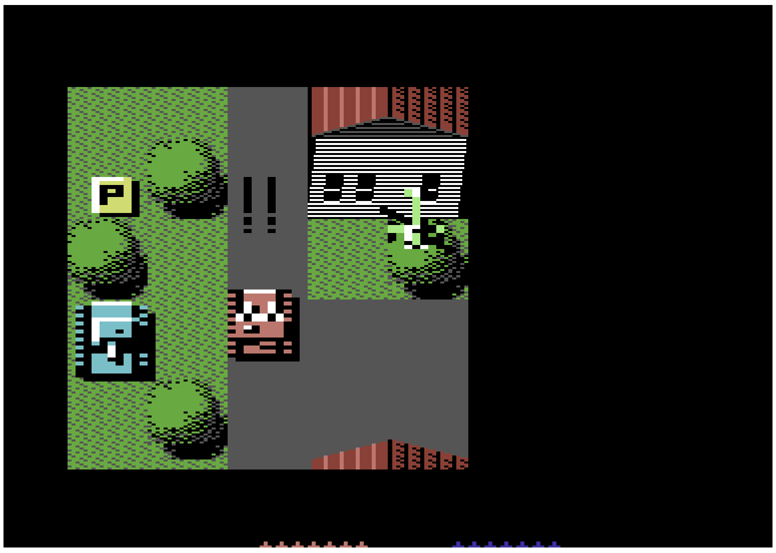
***Example: Re-Alienator by Alf Yngve used shared scoring and lives, and twin mode operations***

This second shared scoring/lives method involves twin mode players. This is where both players are controlled using one joystick, but the difference is that the players can move according to the SEUCK settings. This method is very handy for games where one player can play the role of a soldier, or vehicle, and another where another player can be a cross-hair target. You can also enable/disable permanent invincibility on either player, should you wish to set the game where the player does not want to die. The clause is that the player's lives are both shared. After all lives have been lost, the code jumps to the GAME OVER prompt.

***Note: If you are linking the linked players mode to the custom score panel (twinmode.tas). You must change the address being read ($5ea3) to read the address of where you have placed the custom score panel (store). Also your game should disable extra lives for every 10,000 points as this will cause confusion over extra lives.***

## SPRITE PRIORITY MODE

**by Richard Bayliss**
*File: sprpriority.tas*
*MEMORY USED: $006E*



**Example: Panzer Patty's Pink Tank Adventure used sprite changes**

This cool enhancement plays around with the sprite priorities. It can trigger three different things once installed. Your SEUCK games can now use hi-resolution sprites (which you should create first).  There is also the sprite-background priorities, where the sprites can hide behind the background or stay in front. Finally there is expanded sprites X or Y position. This means during your game you can use super sized sprites. However, this is best used in Sideways SEUCK because sprites that enter or leave the top border shrink back to their normal size.

Use the following pointers to set up the sprite priorities:

**gamemc** = Multi colour mode for main game sprites
**panelmc** = Multi colour mode for score panel

**sprbg** = Game sprites in front/behind background

**sprxpdx** = Game sprite horizontal expansion
**sprxpdy** = Game sprites vertical expansion

## 15: A practical example: Enhancing CETI 22

For this final part of the chapter, I am going to show you how to enhance Ceti 22. This is a fun Left to Right-Scrolling SEUCK game written by me and Alf. You saw in the manual how the game title screen is designed. I'm going to show you how to install the new front end and also include the enhancements in simple steps.

CETI 22 consists of a couple of ready-made in game enhancement code snippets. These were based on the code snippets from Side 2 of the disk. The first thing which we will want to do is load and run in Turbo Assembler (or Turbo Macro Pro). (If you are using Retro-Replay, there might be a built-in turbo assembler. Enter TASS in fastload to enter it.

The first enhancement code snippet we are going to load in and assemble to disk is the custom score panel sprite display code. The enhancement will display eight sprites, and use the player sprite as the lives indicator. I have already provided the score panel sprites.

1. In Turbo Assembler. Enter ← and then **E.**
2. Enter as the filename SCOREPANEL1P.TAS
3. Check the parameters to ensure that the very first parameters for init and play are set. Also make sure Player 2's score and lives are set for the custom panel code. Select any start address where free memory is available. I have chosen $9500. My CETI 22 project has $9500-$9FFF free. So it shouldn't be a problem to have enhancements plugged in that range.
4. Enter ← **and 5** to assemble to disk.

5. Save as **SCOREPANEL1P/9500** (This is the filename, and also the load address in which the filename is based).

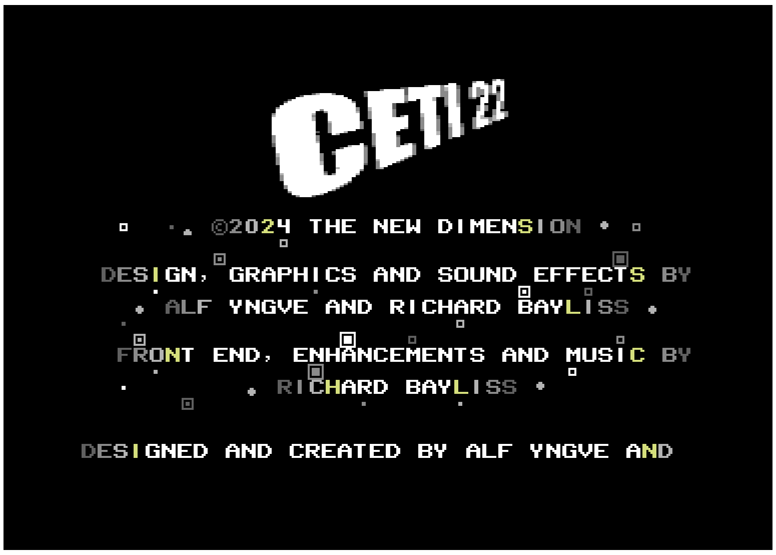
Okay, now we do the same for the smart bomb feature, where assigned enemy objects explode as a whole if the player destroys them – In this game, this routine should only work on large enemies.

1. Run a cold start in turbo assembler. Enter ← and C. Then select Y (yes) when asking for cold start prompt.
2. Enter ← and then E once again.
3. Enter as the filename SMARTBOMB.TAS (This is the smart bomb feature, which is only triggered on the large enemies if the player destroys them. This feature can also be used with collectibles in specific events, if assigned to those objects).
4. Check to ensure that Parameter 2 is set up for this specific feature. Select any start address where free memory is available. I have chosen $9800 for this enhancement. My CETI 22 project has $9500-$9FFF free. So no problems to have an enhancement plugged in that range.
***5.*** Press ← and 5 to assemble to disk.
6. Save as **SMARTBOMB/9800** (This is the filename, and also the load address in which the filename is based).

Now we have these little routines assembled saved to disk. It is time to load in the game, link the new front end and integrate the in game enhancement features.

1. Load in the game (**CETI 22.RAW**) and run it.
2. Press fire to start the game and pause it.
3. Press the RESET button on your Action/Retro Replay cartridge
4. Go intro fastload, and then type in MON or MONITOR to enter the machine code monitor (You need to use this for this part).
5. Load in the title screen using **L "CETI22.TITLE*",8,6580** this will automatically load the title screen to **$6580**
6. Load in the custom panel sprites (10 sprites representing numbers 0 to 9) using **L "CETI22.SCORESPRTS",8,F080** (This will replace the old score panel sprite char data).
7. Load in the score panel code subroutine using **L "SCOREPANEL1P",8,9500**
8. Install it using **G 9500**
9. Load in the enemy smart bomb feature code using **L "SMARTBOMB/9800",9800**
10. Install it using **G 9800**
11. Now save the complete game with the new title screen using **S "CETI22ENH",8,0900,FFFA**
12. Type **G 6580** to test it. If all is working, then pack the filename saved from (11) using a packer, or use Exomizer (Command is: **exomizer sfx $6580 ceti22enh.prg -o enhanced_ceti22.prg**).

If all is working, *Congratulations – You have enhanced the game.*


CETI 22
©2024 THE NEW DIMENSION
DESIGN, GRAPHICS AND SOUND EFFECTS BY
ALF YNGVE AND RICHARD BAYLISS
FRONT END, ENHANCEMENTS AND MUSIC BY
RICHARD BAYLISS
DESIGNED AND CREATED BY ALF YNGVE AND

## 16: Credits

### SEUCK Title Screen Maker V1.8

Programming, charset, logo, design and music by **Richard Bayliss**

### Example game: CETI 22

Created and designed using **Sideways Scrolling Shoot Em Up Construction Kit (Left-Right version)**
Graphics, sprites, front end logo, sound effects and maps by **Alf Yngve**
Design and concept by **Alf Yngve and Richard Bayliss**
Front end charset, music and game attack waves by **Richard Bayliss**

Should you wish to use this software for your own game production publicly, please do not forget to credit me for the front end code. This can either be done on your web site, intro or on your game production, itself. *You are also welcome to use this software to develop front ends for your own magazine e-cover mount games, but the same rules apply.*

For future updates please visit this web site address https://richard-tnd.itch.io/seuck-title-maker

**Have fun.**