



---

# GRAPHIC TOOLKIT MANUAL

---

Foenix F256 JR/K



[HTTPS://GITHUB.COM/ECONTRERASD/F256-GRAPHICTOOLKIT](https://github.com/econtrerasd/f256-graphictoolkit)

# Table of Content

<b>THE GRAPHIC TOOLKIT.....</b>	<b>2</b>
Installation.....	2
Using the launcher application .....	3
<b>SPRITE EDITOR.....</b>	<b>4</b>
DESCRIPTION .....	4
FUNCTION ICONS .....	5
COLOR PALETTE.....	7
SPRITE EDITOR GRID AND SPRITE REPRESENTATION.....	10
DRAWING TOOLS.....	10
.SPR FILE STRUCTURE .....	13
LOADING A SPRITE FROM BASIC .....	14
MEMORY REQUIREMENTS .....	15
<b>TILE MAP EDITOR.....</b>	<b>16</b>
MENU FUNCTIONS .....	18
LAYER CONTROLS .....	20
TILE SELECTOR SECTION .....	23
TILE MAP AREA.....	24
.MAP FILE STRUCTURE .....	25
LOADING A TILEMAP FROM BASIC.....	25
Memory Considerations.....	26
<b>FONT EDITOR .....</b>	<b>27</b>
DESCRIPTION .....	27
EDITOR FUNCTIONS.....	28
USING CUSTOM FONTS FROM BASIC.....	28
<b>OTHER UTILITY PROGRAMS.....</b>	<b>30</b>
Convert Sprites to SUPERBASIC format.....	30
Importing Bitmaps.....	32
Importing Tiles .....	34
Explore Tile Maps .....	35
<b>CONTACT INFORMATION .....</b>	<b>36</b>

# THE GRAPHIC TOOLKIT

## Preface

The Graphic Toolkit is integrated by a series of programs, written in BASIC, destined to help new owners of a Foenix F256 computer to create graphical assets for your new machine. The toolkit includes 3 *Main* applications to create new assets (Sprite Editor, Tile Map Editor and Font Editor) as well as some *Transmute* applications (sorry I just like the name!), oriented to import or transform somehow digital assets.

### MAIN APPLICATIONS

File Name	Application	Description
Toolkit.bas	Main App Launcher	Allows you to launch any of the main applications, and return to the launcher when you quit the selected application
spreeditjr.bas	Sprite Editor	Allows you to define color palettes and edit all the different sized sprites (8, 16, 24 and 32 by 32 sprites) supported by the Foenix F256
fontjr.bas	Font Editor	Allows you to redefine the default character set.
mapeditjr.bas	Tile Map Editor	Allows you to load tile sets and create tile maps from up to 2 tile layers.

### IMPORT APPLICATIONS

File Name	Application	Description
spr2bas.bas	Sprite Converter	Converts sprites into a binary format supported by the included sprite manipulation BASIC commands.
Importbmp.bas	Import Bitmap Files	Read 256 Indexed Color windows bitmaps and save its color palette and its graphical data in a binary 'Foenix Bitmap'.
bmp2tile.bas	Slice Bitmap into Tiles	Slices a bitmap image into 8x8 or 16x16 tiles and save it as a binary tile file.
mapexplore.bas	Tile Map Viewer	Allows you to load a tile set & a map from the Tile Map editor and navigate it with the cursor keys.

## Installation

Please Unzip the latest release in an SD Card, this will unzip the Launcher application (Toolkit.bas) and a few subdirectories to organize the apps and each file type generated (see table below) along with examples in each subdirectory.

SUBDIRECTORIES CREATED AND THEIR CONTENT DESCRIPTION.

SubDirectory	Content
toolkit	Applications from the Graphic Toolkit
sprites	Subdirectory for saved sprites
fonts	Subdirectory for saved fonts
palettes	Subdirectory for saved/imported color palettes
tiles	Subdirectory for imported Tile sets
maps	Subdirectory for saved Tile maps
bitmaps	Subdirectory for imported bitmaps (.FBMP)
bspr	Subdirectory for sprites converted to BASIC format

Please consider that all programs in the Toolkit will use these subdirectories for reading and storing each of the file types that they require or produce.

The only exception are source bitmap Images used when importing bitmaps and Tiles into the Foenix, these should preferably be in the root directory (after importing, most likely these files will be deleted anyway).

### Using the launcher application

Access the apps from the Toolkit through the toolkit.bas program which should be the only program in your root directory (until we come up with a changedir command in SUPERBASIC I believe this is the best way to unclutter your root directory), this program will help you launch all the other applications from the toolkit.



Whenever you are using one of the invoked apps and quit, you'll be taken back to the launcher.

# SPRITE EDITOR

The Sprite Editor requires the following to operate:

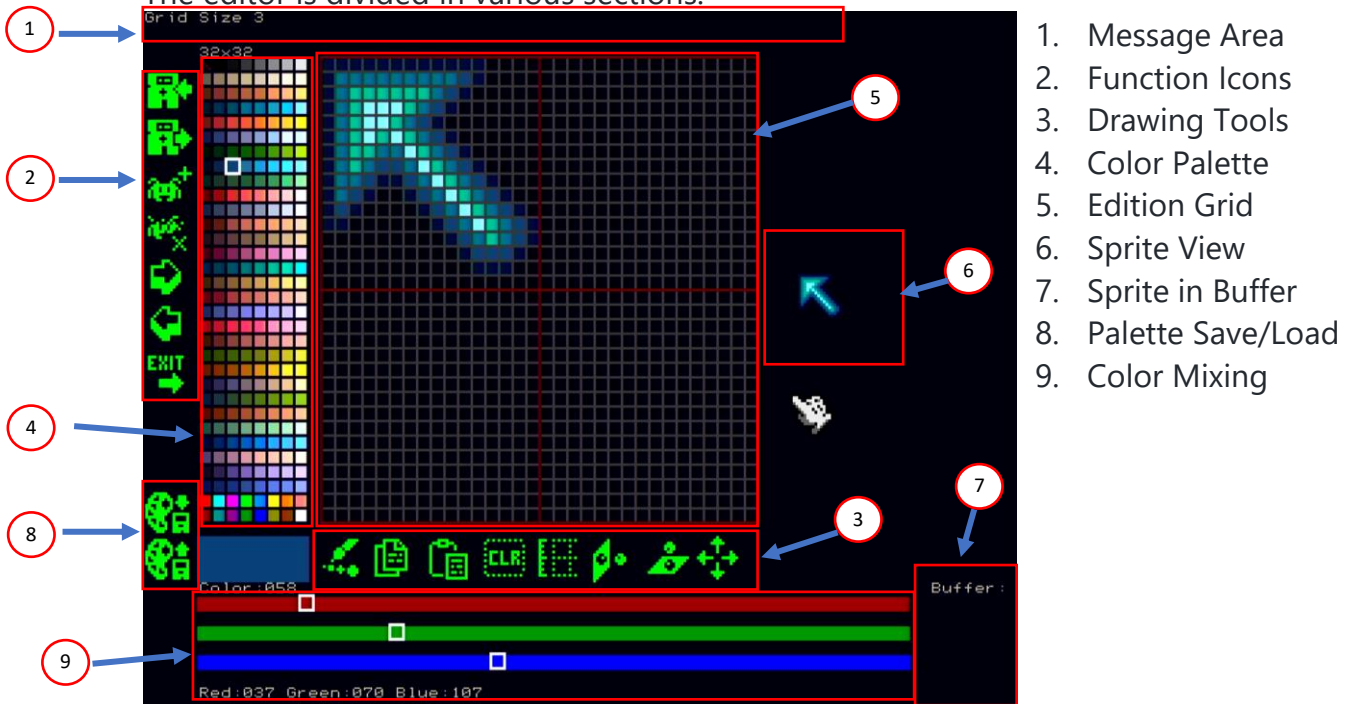
1. Foenix F256 (JR or K models)
2. Joystick or PS/2 Mouse to operate.
3. SD card or IEC Drive if you want to save your creations.

While most of the program is in BASIC (SUPERBASIC) a few Machine Language Routines are used for speed, like drawing the Sprite representation on the bitmap screen, mirroring sprites, shifting sprite pixels around, etc.

## DESCRIPTION

SPREDITJR is a simple sprite editor written in BASIC for the F256 series of Computers created by Stefany Allaire. The Program is either mouse or joystick driven and operates on the 320x240 graphical screen mode; this resolution allows the users to see a reasonable sized representation of the current sprite you are working on.

The editor is divided in various sections:










The program operation will be discussed as we review each one of the sections of the screen.

## FUNCTION ICONS

The Function Icons allow you to handle the main operations needed to administer your Sprites:

Default Menu icons description

Icon	Function	ICON DESCRIPTION
	SAVE	5.25 Disc with incoming arrow
	LOAD	5.25 Disc with outgoing arrow
	ADD SPRITE	Space Invader with '+' sign
	DELETE SPRITE	Space Invader with 'x' sign
	MOVE TO NEXT SPRITE	-> arrow
	MOVE TO PREVIOUS SPRITE	<- arrow
	EXIT PROGRAM	Exit Sign



### SAVE

Clicking this ICON allows you to save all your sprites, a prompt appears at the top of the screen asking for a filename\*. Consider that a default extension (.SPR) is added to your file.



**WARNING:** If you use a filename from an already existing file, the program overwrites the existing file without warning!



## LOAD

Clicking this ICON allows you to load all sprites from an existing sprite file (.SPR extension). A prompt is shown to request the file to load, type a filename (without the extension) and press ENTER.

- If the file is located the file is loaded and the first sprite is shown
- An error is reported if the file doesn't exist.



## ADD SPRITE

When you first enter the Sprite editor there is only one sprite to edit, if you press the MOVE TO NEXT SPRITE or MOVE TO PREVIOUS SPRITE a message will let you know that you are at either the *Last sprite* or *First sprite* already.

To Add an additional sprite, click the ADD SPRITE ICON, which will cause a new sprite to be **appended after the last sprite** and moving to the last sprite.



## DELETE SPRITE

When you Click this Icon **the last sprite of the set is deleted**, and you are positioned at the new *Last Sprite*.

If you only have one sprite a message will remind you that you can't delete your *Last Sprite*.



## MOVE TO NEXT SPRITE

When you have many sprites, this ICON allows you to navigate to the next sprite, the grid automatically changes and reflects the sprite definition of your next sprite.

If you click continuously, the sprites change fast enough to do some rudimentary animation tests.



## MOVE TO PREVIOUS SPRITE

When you have many sprites, this ICON allows you to navigate to the previous sprite, the grid automatically changes and reflects the sprite definition of your next sprite.

If you click continuously, the sprites change fast enough to do some rudimentary animation tests.



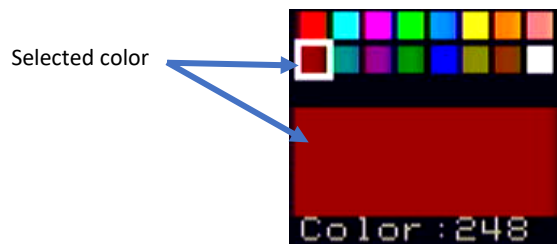
Click to exit the Sprite Editor, please remember to save your work before exiting the program, if you exit and enter the program again all sprites will be cleared as part of initializing the Editor.

## COLOR PALETTE

The color palette includes a pre-selection of color Gradients, this color palette is currently hardcoded in the Editor.

### Picking a Color from the Palette

Left Clicking on any color on the palette will change the brush color to the selected color, a rectangle will show the currently selected color on the palette, along with the color index number under it for easy identification of the selected color.



### RGB Value

Red:128 Green:000 Blue:000

Below the Palette controls the red, green, and blue components of the selected color are shown, just in case you need this RGB values.

### Transparent Color

The top left color in the Grid is the transparent color, or better said the background color, this is denoted by an / symbol over this color.



## Picking a Color directly from the GRID

When you are editing a Sprite and change colors is easy to forget which color you were using on another section (yes even with a limited palette of 256 colors things can get confusing!) and hunting down for the right color on the palette can consume precious time.

To improve the color selection workflow, a color pickup function was added to the spacebar, when you press it, you'll pick up the color pointed by the hand icon (pointer), this helps a lot in getting a previous selected color faster, if you are using the program with a mouse, you can click the right mouse button to pick the color as well.

## Editing new colors

A new section at the bottom of the screen shows the color components (RGB) of the current selected color with handles on the corresponding bar relating to the value of each component (RGB).

Moving the handles with the joystick/mouse modifies the current color value and change the color in real time.

As an additional help, the red, green, and blue components values change as you edit the colors.



👁 **Note:** If you edit the Transparent color, you are changing the background color of the screen, which is not part of the palette, but is useful for visualizing possible contrast issues

## Loading / Saving Palette Colors

The Icons below the Menu Icons allow you to either load a palette from disk or save a palette to disk.

*Load & Save palette icons.*



An extension (.PAL) will be appended automatically to the name you provided when loading or saving a palette file.



**WARNING:** If you use a filename from an already existing file the program overwrites the existing file without warning!

## Default Palette Color Credits

- The initial color palette was borrowed from the public palettes published in the following link:

(<https://lospec.com/palette-list/duel>)

Although it has evolved since version 1, with a major revamp to make smother gradients in some cases and add true black, true white and 16 primary and secondary colors at the end.

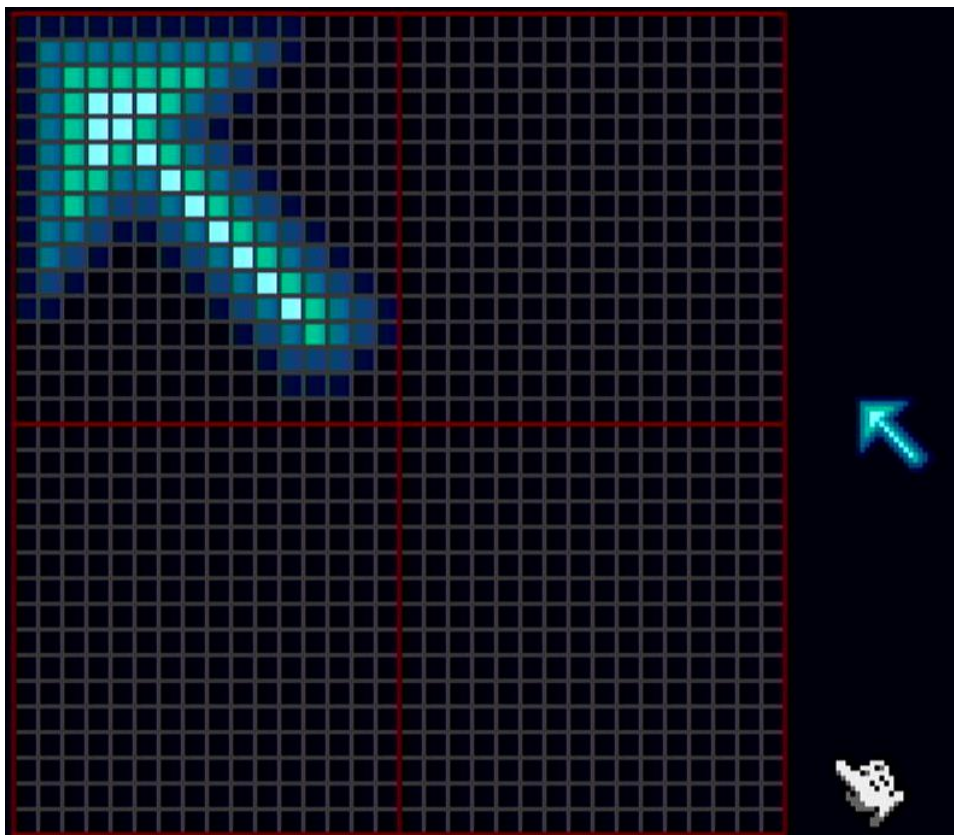
- Credits to the original creation of this palette are due to ARILYN@ARILYNART

<https://t.co/3Kxr6ZPzmu>

## SPRITE EDITOR GRID AND SPRITE REPRESENTATION


Once you have selected a color, just left click on any square on the grid to assign the current color to the pixel, you can hold the mouse or joystick button and move the pointer around to draw faster.







As you edit the sprite in the grid, an actual sprite on the right side of the grid is updated in real time, so yes WYSIWYG!



## DRAWING TOOLS

The Drawing tools contains tools that help you in drawing your sprites (description of the icon in parenthesis):

	Tool	Icon Description
	BRUSH SIZE	A Brush with increasing square point size (1x1,2x2,3x3,4x4)
	COPY	Copy Icon

	PASTE	Board with document ready to be pasted
	CLEAR	Empty space with CLR letters
	GRIDLINES	Ruler with guidelines
	MIRROR X	Ball reflected in Standing Mirror
	MIRROR Y	Ball reflected in Flat Mirror
	SCROLL	Arrows Left/Right/Up/Down




## BRUSH SIZE

Using the BRUSH SIZE Tool will change how many pixels are drawn on the SPRITE EDITOR GRID.

The default brush size is 1 pixel, but available options are 2x2 / 3x3 / 4x4 pixels\*.

Every time you click in the BRUSH SIZE icon the BRUSH SIZE toggles to the next option and the size is shown on the message area.

If you click enough times the option wraps around to go back to the default pixel size (1).

 **Note:** When the BRUSH SIZE is 4x4 it lays down more pixels but the responsiveness when you move while drawing is somewhat reduced.



## COPY

When you are editing sprites, chances are that you are animating something and having the option to copy an already completed sprite into a new frame of animation is essential.

## What did I just Copy?

When you click the COPY Icon a representation of the current sprite is copied into the buffer memory, you might continue to edit your current sprite and choose to Paste the untouched copy latter.

To make it easier to remember what you copied into memory the sprite you copied appears in the bottom right corner of the screen under the "Buffer" section, so you can see what is in the buffer.



When you click PASTE the current sprite in the buffer replaces whatever is in your current sprite.



When you click this icon *-not surprisingly-* the current sprite is wiped clean, sometimes is better starting from scratch. (And yes, this function was added since erasing a 32x32 sprite pixel by pixel takes a while)



Clicking this ICON will show a color grid on your sprite editing area, the gridlines toggle it's spacing with each click between 4x4, 8x8, 16x16 and removing them completely.

This function is designed to help aligning or centering sprites, calculate height, or plan displacement in animation frames.



Clicking this ICONS flips your sprite from left to right.



## MIRROR Y

Clicking this ICONS flips your sprite from top to bottom.



## SCROLL

Clicking this ICONS enables the scroll sprite function, moving the joystick will move the sprite image around the editing area, if you move past one of the sprite boundaries the sprite image will wrap around on the other side. This functionality is useful to align and/or recenter sprites.

## .SPR FILE STRUCTURE

The Save file structure is very simple, but beware the format is not compatible with how sprite commands in SUPERBASIC expect the data.

SPRITE INDEX BLOCK – 1024 BYTES

Section	Description
First byte	Number of sprites in file (up to 255)
768 Bytes	Sprite Index data
'n' data blocks	Data used for each sprite in the file

INDEX ENTRY – 3 BYTES

Section	B7	B6	B5	B4	B3	B2	B1	B0
1 <sup>st</sup> Byte	Unused						Sprite Size 0=32x32, 1=24x24 2=16x16, 3=8x8	
2 <sup>nd</sup> Byte	Offset to Sprite Initial Data Block (High Byte)							
3 <sup>rd</sup> Byte	Offset to Sprite Initial Data Block (Low Byte)							

The Header and index Data included are required, this allows any program in BASIC or ASSEMBLER using the file to know:

- Know how many sprites are in the file.
- What Sprite Size use for each sprite in the file.
- Where to point the sprite registers for High Byte / Low byte for each image.

## LOADING A SPRITE FROM BASIC

SUPERBASIC uses a different structure to sort how many sprites are in memory and what size they are. I'll cite the SUPERBASIC manual:

At present there is a very simple data format.  
 +00 is the format code (\$11)  
 +01 is the sprite size (0-3, representing 8,16,24 and 32 pixel size)  
 +02 the LUT to use (normally zero)  
 +03 the first byte of sprite data"

The size, LUT and data are then repeated for every sprite in the sprite set. The file should end with a sprite size of \$80 (128) to indicate the end of the set.

This translates into a "Header", "Sprite Data Entries" and "End marker":

Header

Byte	Field	Description
0	Header	One magic byte with value \$11 (17 decimal)

Data Entry for each Sprite

Byte	Field	Description
0	Sprite size	Value 0-3: representing 8x8, 16x16, 24x24 and 32x32 size
1	LUT	Palette number to use (0-3)
2..n	Sprite Data	n bytes of data (dynamic length, according to sprite size)

End of Data Marker

Byte	Field	Description
0	End Marker	One byte with value \$80 (128 decimal)

You can convert sprites from the sprite editor to SUPERBASIC using "spr2bas.bas"

1. Once converted, you can use BLOAD to load the sprite file into memory, for example:

```
BLOAD "SPRFILE.BSPR", $30000
```

This example loads the sprite file into RAM beginning at Address \$30000, the default area where SUPERBASIC expects sprites.

You can use then SUPERBASIC commands to show the first SPRITE in the file.

```
SPRITES ON  
SPRITE 0 TO 100,100  
SPRITE 0 IMAGE 0
```

## MEMORY REQUIREMENTS

- Theoretically the sprite editor is limited to editing 255 sprites (not that I have tried to define that many sprites... yet)
- The maximum memory required for 256 sprites of 32x32 size is \$40000 (262,144 bytes), the Foenix F256 (Jr or K models) have 512k of included memory so you could define this many sprites.
- Sprites on the F256jr can take different sizes 8x8, 16x16, 24x24 and 32x32, so memory requirements might vary depending on how many sprites of each size you define.



# TILE MAP EDITOR

The Tile Map Editor requires the following to operate:

1. Foenix F256 (any model)
2. Joystick or PS/2 Mouse to operate.
3. SD card or IEC Drive if you want to save your creations.

You need these two files to execute the program (BASIC program and default color palette):

- MAPEDITJR.BAS
- DEAFULT.PAL

The complete program is written in SUPERBASIC.

## DESCRIPTION

MAPEDITJR is a simple Tile Map Editor written in BASIC for the F256 series of Computers created by Stefany Allaire.

The Program is either mouse or joystick driven and operates on the 320x240 graphical screen mode; this resolution allows the users to see a reasonable portion of a 16x16 Tile Map onscreen (20x12 tiles per screen, considering the menu & tile selection area).

The editor divides the screen in various sections:










- |   |               |
|---|---------------|
| 1. Tile Map Area                        | 5. Status Bar |
| 2. Layer Selection & Preferences        | 6. Tiles      |
| 3. Tile Set Controls (Load/Next Page)   |               |
| 4. Tile Map Section (Options/Open/Save) |               |

The program operation will be discussed as we review each one of the sections of the screen.

## MENU FUNCTIONS

These Icons will allow you to handle the main operations needed to administer your Tile Map files and Tile Sets

Menu icons description

Icon	Function	ICON DESCRIPTION
	MAP PREFERENCES	Wrench on Document
	OPEN MAP	Open Yellow Folder
	SAVE MAP	Yellow Floppy Disk
	OPEN TILESET	Open Folder w/Tiles
	IMPORT TILES	Open folder w/Space invader
	NEXT TILESET PAGE	-> Arrow pointing at tiles
	EXIT PROGRAM	Exit Sign



### MAP PREFERENCES

Clicking this ICON allows you to setup the Map size as measured in tiles, the default value is 20x15 (This is equivalent to one 320x240 screen). Be aware that when you change dimensions all layers will be synchronized to this size.



### OPEN MAP


Clicking this ICON will prompt you for a Map name, if this file with the extension .map exists it will load all Layers into memory for editing or viewing in the Tile Map Editor.



## SAVE MAP

When you click SAVE the program asks you how many layers you want to save, currently 3 layers are supported, 1 and 2 being tile layers and 3 a property layer. After selecting the number of layers, you are prompted to provide a name.

All layers are saved as individual map files with consecutive numbers after the file name. An extension (.MAP) will be appended automatically to the name you provided to each layer and an extension (.PMAP) to the property layer.

 **Note:** The first 2 bytes of the first map file contains the size of the map (X,Y)




**WARNING:** If you use a filename from an already existing file the program overwrites the existing file without warning!



## OPEN TILESET

When you Click this Icon, you are prompted for the name of a Tile set file (extension .set), the program will try to open and load into memory this Tile set. Automatically the program will also look for a matching palette file with the same name but with a .pal extension, if it is found it will be loaded as the Tile set palette.

 **Note:** Although the Foenix platform can support up to 8 different Tile sets (each with 256 tiles) the Tilemap Editor currently exposes only one



## IMPORT TILES

When you click this icon, you are prompted for the name of a Sprite File (extension .spr) the program will try to open and load into memory the sprites. Please consider that all sprites in the set should be in 16x16 resolution for the tiles to load correctly.

The program will search for a palette file with the same name as the sprite file and load it immediately, if no palette file is found the default palette will be used.



## MOVE TO NEXT TILE PAGE






Tile sets can have a lot of tiles and the tile selection section can only show 45 tiles per page, so to view more tiles, you need to click this ICON to show the next page of tiles. If you go past the last page (page 5) it moves to the first again (page 0).



## EXIT

Click to exit the Tile Map Editor, please remember to save your work before exiting the program, if you exit and enter the program again all tiles on the map will be cleared as part of initializing the Editor.

## LAYER CONTROLS

Icon	Function	ICON DESCRIPTION
	SPECIAL LAYER CONFIG	Wrench on Tile
	SELECT LAYER 1	Layer 1
	SELECT LAYER 2	Layer 2
	SELECT SPECIAL LAYER	Layer S
	SPECIAL LAYER PROPERTY	Selected Property Number

The Foenix JR works with Layers to form “interesting” Tile Maps, be aware that sprites can be on top or hidden by each layer, so multiple layers can be used to play with the visibility, giving images (sprites) the illusion of being above or below each layer, skillful use of the layers and sprite layering system will add “dimension” to your games, demos, or programs.

The Map editor can draw on 2 tile layers (out of 3) and has logic to create a special layer with up to 8 properties so that your program can have additional information about any part of the map and how to treat it. For example, is the tile impassable terrain? would the tile reward treasure? or maybe quests? or is it a monster generator? usage of the 8 properties is only limited by your creativity!



## **ESTABLISHING PROPERTIES OF SPECIAL LAYER**

Allows you to set the name of the 8 properties that can be used in the special layer, these properties will be stored in a special file with an extension .PMAP (properties Map)



### **SELECT LAYER 1**

This is considered the Bottom Layer. When you click on it a small red marker is shown on top of it, so that you know which layer you are working on. All tiles painted from this moment on are associated to this layer.



### **SELECT LAYER 2**

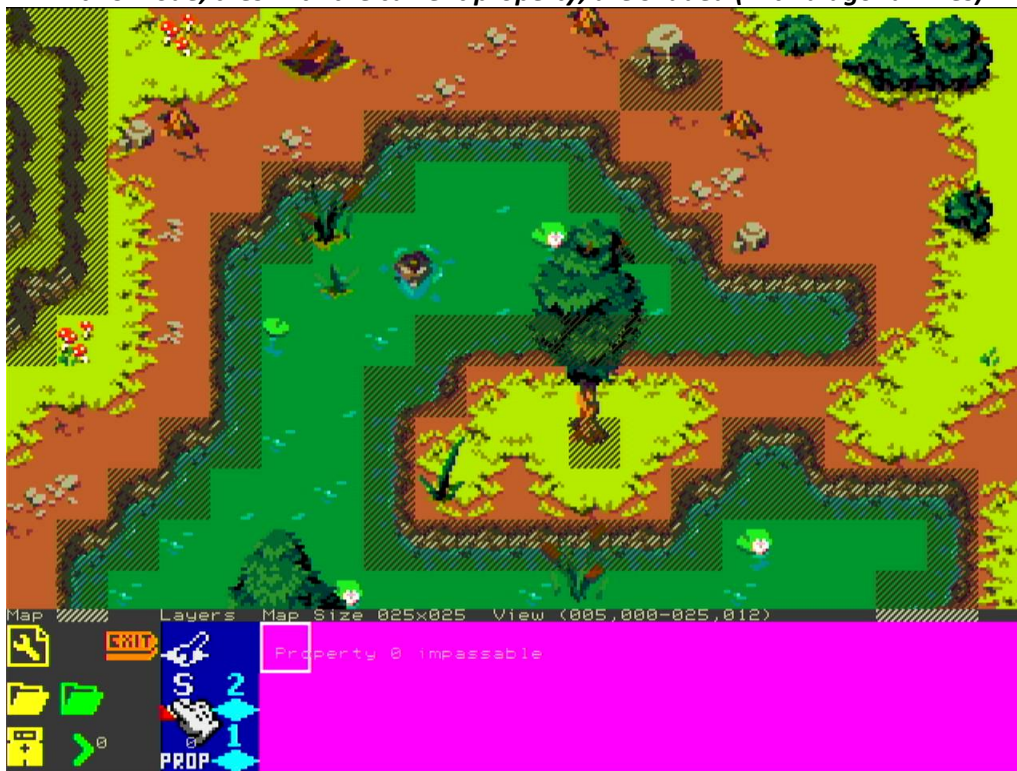
This is considered the top layer. When you click on it a small red marker is shown on top of it, so that you know which layer you are working on. All tiles painted from this moment on are associated to this layer and appear on top of tiles in layer 1.



### **SELECT SPECIAL LAYER**

This is a logical layer, when you click on it a small red marker is shown on top of the special layer icon, all tiles from the tile selection area disappear and instead the name of the current selected property is shown.

*In this mode, tiles with the current property, are shaded (with diagonal lines)*



Depending on your Tileset and palette you might need to change the shading color to make it more visible.

**Note :** To change the shading color press 'F1'



The idea of a property layer is that each tile can be assigned up to 8 different properties, this allows a program to control specific behaviors per tile, for example property 0 could imply impassable tiles, property 1 could be configured to be a respawn position, and so on...



## SET PROPERTY FOR SPECIAL LAYER

Clicking this ICON shifts to the next property and re-shades tiles with this new property (once you try to go beyond property 7 the property shown will reset again to 0).

## TILE SELECTOR SECTION

Once you have loaded a Tile set the tile selector area will show the available tiles, a white border will be shown around the selected tile. You can select different tiles by clicking on them. The current selected tile will be used to "paint" in the Tile map area.

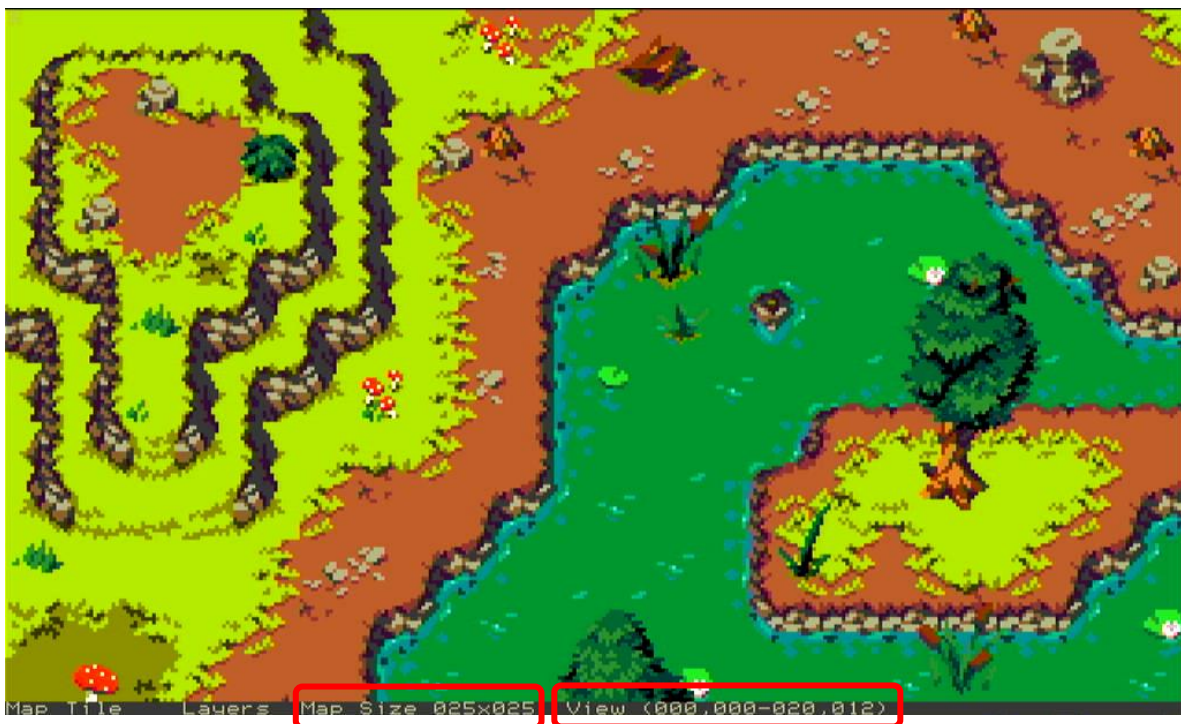




 **Note:** The transparent color in the tiles is shown as bright pink

## TILE MAP AREA

Once you have selected a Tile you can go to the Tile Map area and “paint” with this tile by pressing the button and dragging either with the joystick or mouse over the area.



The Tile Map might be bigger than the image shown on the Tile Map editing area, to understand where you are exactly there is information on the status bar:

1. Map Size : Shows the map dimension in tiles (width x height)
2. View : Shows the tile (col,row) in the upper left corner – lower left corner

Example (001,000-020,012)

This should be interpreted that you are looking at a window that goes from the tiles at (1,0) to (20,12) in the Tile map.

## NAVIGATING THE MAP

If you want to move the viewport around to a different section of the map you can use the cursor keys to scroll the window to a new position

If instead you want to move faster through the tile map you can use the WASD keys to move up, down, left, or right but at one full screen at a time.

## .MAP FILE STRUCTURE

On the first map file the first two bytes specify the map dimensions

Section	Description
First byte	Map width
Second Byte	Map height

All other bytes in the map file contain the map data (2 bytes per tile)


Tile Definition Data (2 bytes)

Byte	Description
0	Tile Number to display (0-255)
1	Tile Set (bits 0-2), Tile Palette (bits 3-5)

## LOADING A TILEMAP FROM BASIC

An example program is provided:

Program	Description
MAPXPLORE.BAS	This program allows you to load a tile set, palette, and tile map in memory, poking the necessary registers to setup and manipulate the layers, setup the tile map & control the portion of the tilemap shown along with the smooth scrolling

 **Note :** Currently there is no example for using SUPERBASIC Tile commands since they are designed to use 8x8 tiles and this Tile Map Editor uses 16x16 tiles.

## Memory Considerations

Resources are limited by the F256 memory, the internal memory organization of the program is as follows.

Memory Address	Description
\$10000-\$22BFF	Bitmap for Menu overlay
\$30000- \$3FFFF	Sprites used for cursor, pointer, and Tiles in Tile selection area
\$40000- \$4FFFF	Tile Set memory (up to 255 tiles)
\$50000- \$5FFFF	Tile Map Layer 1 memory (enough for 256x128 tiles or equivalent)
\$60000- \$6FFFF	Tile Map Layer 2 memory (enough for 256x128 tiles or equivalent)
\$70000- \$7FFFF	Property Map memory (enough for 256x128 tiles or equivalent)

Due to memory restrictions, some compromises have been taken, such as allowing only one Tile set, two tile maps and one property map.

Memory could be distributed differently to allow for example more Tile sets, or more tile map layers. But I believe that this design offers a lot of flexibility.

## Other Tips

When Creating a Tile Map your best shot is to create a Map of the intended size, but if for some reason you require to re-dimension your map later, consider the following guidelines:

1. Changing the width of an existing map alters the fundamental map geometry and all existing tiles in the old map will be scattered in the resized map!
2. Changing the Height of an existing map is safe, additional rows are added and you can edit the new tiles without disturbing the existing tiles.
3. You can start with a one-layer map and then decide to add a second layer or a special property layer without disturbing the previous layers.

# FONT EDITOR

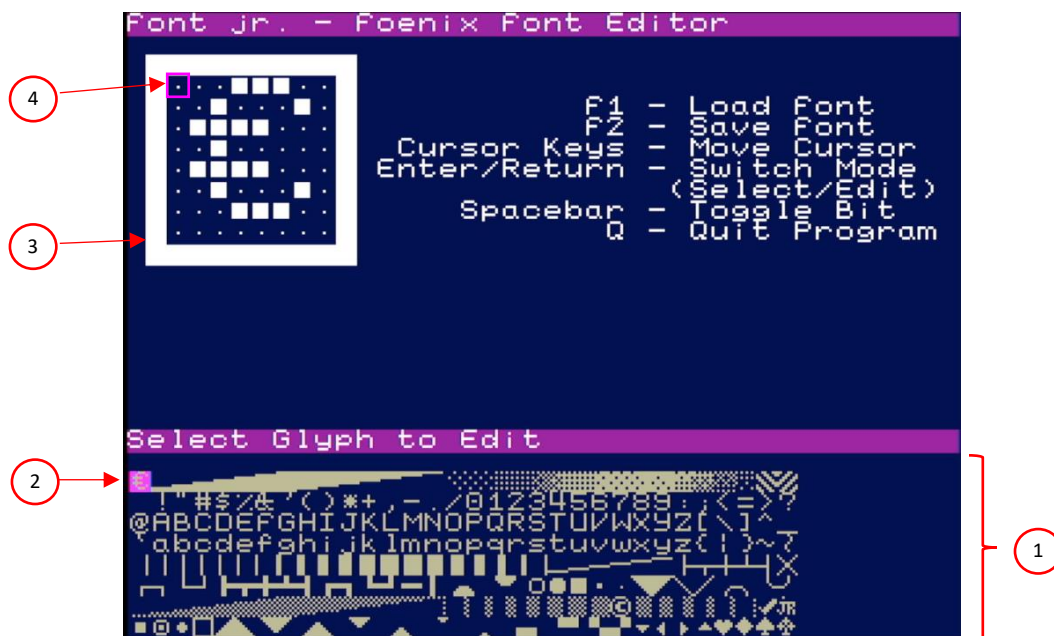
## DESCRIPTION

The Foenix F256 has support for two separate character sets, so you can easily have two different fonts, even one designed by you! the Font Editor allows you to edit the default character set, see the results as you make the changes in real time and save the font in a file for usage in your own programs.

Fonts in the F256 are inspired and compatible with other retro 8-bit computers, it uses monochrome fonts, 8x8 pixels in size. The font Editor uses the 40x30 text mode, since it provides a bigger representation of each character.

Since the Font Editor is the simplest of the main applications in the toolkit you can operate by using only the Keyboard.

Let's review how the Font editor screen is divided:



SECTION	DESCRIPTION
1- Character Display and Selection Area	Shows all the 256 available characters in the set, shows changes in real time
2- Character Selector	This cursor allows you to select the character to Edit
3- Character Edition Area	The representation of the character being edited
4- Edition Cursor	Allows you to move in the character representation and toggle bits on and off

## EDITOR FUNCTIONS

### Editing Fonts

When entering the Editor, you are by default in the *Character selection area* (1), use the CURSOR KEYS to move to the character you want to edit.

Press ENTER | RETURN over the character you want to edit, and it will appear in the *Character Edition Area* (2).

Use the CURSOR KEYS to move around the pixels in the character and press SPACE to toggle the current bit either on or off. As you edit the pixels the selected character also changes in the *Character selection area*.

Once you are finished editing the current character, press ENTER | RETURN to move to the Character Selection Area

### Saving Fonts

To save the complete character set press F2 and provide a name, please note that an extension (.FONT) is added to your provided filename.

👁 **Note :** All fonts are loaded from, and saved to the subdirectory /fonts



**WARNING:** If you use a filename from an already existing file the program overwrites the existing file without warning!

The format of a FONT file is a binary file of 2048 bytes.

### Loading Fonts

To load a font press F1, you will be prompted for a file name, please note that an extension (.FONT) is added to your provided filename.

## USING CUSTOM FONTS FROM BASIC

You can use the following routine to Load a font in one of the two memory areas reserved for fonts.:

```
1000 proc loadfont(a$,slot)
1010   if (slot<0)^(slot>1) then print "Only slots 0-1 are available":end
1020       a$="fonts/"+a$:try bload a$,$7800 to ec
1030       if ec=0
1040           ?1=1:rem "Activate font memory under address $C000"
1050           for c=0 to 2047
1060               ?($C000+(slot*2048)+c)=?($7800+c)
1070           next
1080           print "font ";a$; "successfully loaded!"
1090       else
1100           print "file ";a$;" could not be loaded."
1110       endif
1120 endproc
```

### Usage Example 1: Loads "angle.font" into font slot 0 (default font)

```
loadfont("angle",0)
```

### Usage Example 2: Loads "single.font" into the second font slot and activates second font.

```
loadfont("single",1)
poke $d001,32: rem "Set bit 5 on to activate font slot 1)
```

## Other Tips

You can obtain binary Font files for 8-bit platforms such as the commodore 64, Atari 8-bit Series of computers, add a .font extension and load them on the Editor

[GitHub - patrickmollohan/c64-fonts: Fonts for the Commodore 64.](https://github.com/patrickmollohan/c64-fonts)

Also, if there is an application that changes the Font in the F256 you can use the reset button to initialize the machine, but you will notice that the Font won't be restored to the original one (this only happens on full powerup, which forces it to be loaded from FLASH) , load the font editor directly with the following command:

```
LOAD "toolkit/fontjr.bas"
```

Run the editor, and voila! the current font will be shown for editing, and you can save it using F2.

## OTHER UTILITY PROGRAMS

A few utilities are included along with the Toolkit, Under the Transmute Apps option:



### Convert Sprites to SUPERBASIC format

SUPERBASIC uses a different structure to sort how many sprites are in memory and what size they are. I'll cite the SUPERBASIC manual:

```
At present there is a very simple data format.  
+00 is the format code ($11)  
+01 is the sprite size (0-3, representing 8,16,24 and 32 pixel size)  
+02 the LUT to use (normally zero)  
+03 the first byte of sprite data"  
  
The size, LUT and data are then repeated for every sprite in the sprite  
set. The file should end with  
a sprite size of $80 (128) to indicate the end of the set.
```

This translates that in memory SUPERBASIC expects a "Header", "Sprite Data Entries" and an "End marker":

## Graphic Toolkit

### Header

Byte	Field	Description
0	Header	One magic byte with value \$11 (17 decimal)

### Data Entry for each Sprite

Byte	Field	Description
0	Sprite size	Value 0-3 Representing sizes 8x8, 16x16, 24x24 and 32x32 size
1	LUT	Palette number to use (0-3)
2..n	Sprite Data	n bytes of data (data length, according to sprite size)

### End of Data Marker

Byte	Field	Description
0	End Marker	One byte with value \$80 (128 decimal)

You can convert sprites from the sprite editor to SUPERBASIC using the Toolkit's option "Sprite to BASIC " available in the Transmute Apps Menu

1. The program prompts you for a filename (the sprite file is expected to be in the /sprites folder)
2. If the file is Found the SPRITES are loaded and converted to the new format
3. Finally, a file with the same name as the original file but with an extension (.BSPR) is saved in folder /bspr

Once converted, you can use BLOAD to load the sprite file into memory, for example:

```
BLOAD "SPRFILE.BSPR", $30000
```

This example loads the sprite file into RAM beginning at Address \$30000, the default area where SUPERBASIC expects sprites.

You can use then SUPERBASIC commands to show the first SPRITE in the file.

```
SPRITES ON  
SPRITE 0 TO 100,100  
SPRITE 0 IMAGE 0
```



## Importing Bitmaps

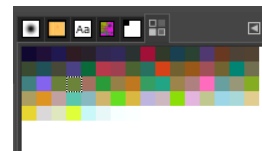
### Preparing Images to be Imported

To turn an Image into a suitable Bitmap for importing with the tools provided in the Toolkit we are going to be using the program "Gimp", since it's a powerful Image Editor that works on Windows / Linux / MAC and best of all its Free!

- Download Windows & Linux Gimp versions from [GIMP - Downloads](#)
- Download the MAC version from the developer's version. [GIMP - Development Downloads](#)

Once Installed, launch Gimp, and follow these instructions:

1. Open your image in GIMP.
2. If needed resize it, since the maximum bitmap size in the Foenix F256 is 320x240 (or 300 x 200 in 70Hz mode)
3. Convert the Image to Indexed 8-bit color by using the following Menu Options: **Image | Mode | Indexed.**
  - On the dialog box select the following options:
  - *Generate Optimum palette.*
  - *Maximum number of colors 255*
4. Save the Indexed Bitmap file by using the following Menu Items:  
**File | Export As...**
  - Remove the current extension of the file and change it to **.bmp**
  - A message will appear informing that "Cannot export indexed image with transparency in BMP format – alpha channel will be ignored."
  - Don't mind the warning and click on OK.
  - The file will be created correctly.
5. Optional Step - To View the Indexed File Palette generated select the following Menu Options:  
**Windows | Dockable Dialogs | Color Map**
  - On the left side the palette will appear



- You can alternate click over any color to show a context menu, and then click on rearrange color map to show all colors and be able to change the order by dragging them and dropping them.

## Importing the prepared Bitmap

You can convert 8-bit Indexed Bitmaps to a Foenix binary format by using the Toolkit's option "Import Bitmap" available in the Transmute Apps Menu

1. The program prompts you for a filename, (the .BMP extension will be added automatically) the file is expected to be in the root folder of the SD card)
2. If the file is found the File is loaded and parsed, and if indeed it's an 8 bit indexed file it's imported
3. Finally, 2 files with the same name as the original file but with extensions (.PAL) and (.FBMP) will be saved in the folders /palettes and /bitmaps respectively.

To load an imported FBMP file you can use the following SUPERBASIC code:

```
10 cls:bitmap on:bitmap clear 0
20 input "name of bitmap to load (FBMP will be added)? ";n$
30 loadbitmap(n$)
40 print "done!.."
50 end
1000 proc loadpalette(name$,n)
1010 name$="palettes/"+name$+".pal"
1020 try bload name$,$7900 to ec
1030 print "Loading ";name$
1040 if ec<>0
1050 print "File "+name$+" Not Found!"
1060 end
1070 endif
1080 ?l=1:for c=0 to 1023:?( $D000+c+n*$400)=?( $7900+c):next: ?l=0
1090 endproc
1100 proc loadbitmap(a$)
1110 name$="bitmaps/"+a$+".fbmp"
1120 try bload name$,$10000 to ec
1130 if ec=0
1140 loadpalette(a$,0)
1150 else
1160 print "File ";name$;" not found":end
1170 endif
1180 endproc
```

*\*This file exists in the toolkit release as:  
toolkit/showbmp.bas*

## Importing Tiles

### Preparing Images for Tiles

Open or Create a Source Image for your tiles in Gimp.



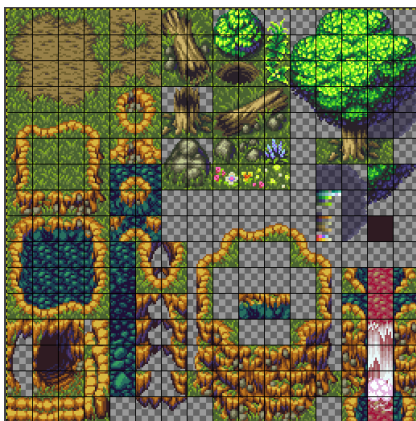
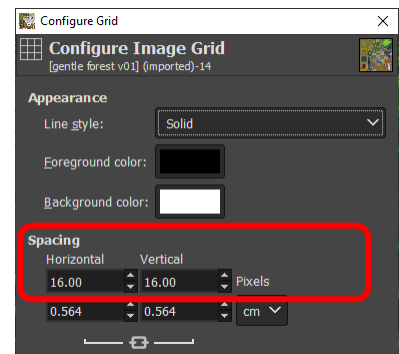
1. The obvious size for a Tile Map would be to make the image 256x256 pixels (16x16 tiles), but the F256 resolution is 320x240, so the Y size won't be enough to fit the tile image in one screen!
2. An alternate size that would fit in one screen is 272x240 (17x15 tiles), this size is ideal for use with the Tile Editor from the *graphic toolkit* since it allows up to 255 tiles out of 256 always forcing Tile 0 as a blank tile.
3. Consider creating or rearranging your tiles to match this file size.

### Optional Step - Configure Grid | Snap to Grid

Select the following options from the Menu:

#### **Image | Configure Grid**

- On the Dialog that appears adjust grid to 16x16 or 8x8 on pixels.



- To show the Grid select the following Menu Items: **View | Show Grid**
- Having the grid helps if you are creating a tile set from scratch to dimension your images properly.
- If you are rearranging your tiles in a different size image (as suggested on step 1), you would benefit from enabling Snap to Grid, this way Gimp ensures that when you are copying & pasting images they are always aligned with

the Grid. Do this with the following menu items  
**View | Snap to Grid**

## Importing The Prepared Bitmap into Tiles

You can import your 8-bit Indexed Bitmap to your F256 and slice into tiles by using the Toolkit's option "Import Tiles" available in the Transmute Apps Menu

- The program prompts you for a filename, (the .BMP extension will be added automatically) the file is expected to be in the root folder of the SD card)
- If the file is found the File is loaded and parsed, and if indeed it's an 8-bit indexed file it's imported first as a bitmap.
- After the Bitmap finishes loading the program slices the image into 16x16 images beginning from the top left corner, moving left until the end of the image, and continuing through all remaining rows.
- Two files with the same name as the original bitmap file but with extensions (.PAL) and (.TILE) will be saved in the folders /palettes and /tiles respectively.

The .TILE File can be directly loaded in the "Tile Map Editor".

## Explore Tile Maps

The last of the Transmute Apps is a lightweight application that allows you to explore Tile Maps built with the "Tile Map Editor".

While this is not exactly a Transmute application it serves as an Example on how to use layers, tiles, and Tile Maps to create vivid scenes.

Please feel free to use any code in this app for your own purposes.

An Example Map named "mapdemo" is included, to use it follow these instructions:

- Select the "Explore Tilemap" option in the Toolkit.
- Once The application loads it prompts for a Tile Set to load, type "mapdemo" to load the "mapdemo" Tile Set (Loading the Tile Set also loads any palette with the same name, if the palette does not exist it defaults to the palette from the Sprite Editor)
- After the Tile Map & Palette finish loading a new prompt appears to load a Tile Map, please type "mapdemo" again to load the mapdemo Tile Map
- You'll notice that 3 files are loaded, this is correct since each map correspond to each layer that composes the map.

- Finally, the program informs you that you can navigate the tile map by using the Cursor Keys or press Q to quit.

## Navigating the Tile Map

After the Tile Map appears you can navigate with the cursor keys, you'll notice that the tiles move smoothly thanks to the smooth scrolling support.

Feel free to study the routines to understand the smooth scrolling.

## CONTACT INFORMATION

Hopefully you'll enjoy using this toolkit as much as I enjoyed creating it, if you have any comment or suggestion feel free to contact me: ***Ernesto Contreras***

Email	eacontrerasd@gmail.com
Discord Server	<a href="https://discord.com/channels/691915291721990194/1008139105386889346">https://discord.com/channels/691915291721990194/1008139105386889346</a> username: <b><i>econtrerasd</i></b>