EnvisionPC

Atari Font Editor v0.8

Programmed by

Mark Schmelzenbach

EnvisionPC: Release 0.8
("The March Hare Edition")

by Mark Schmelzenbach
Copyright ©1997-2006

Introduction

Many years ago, APX released a program called Envision. It was THE game design tool. It had an integrated character editor, map maker and many other useful utilities. Having recently returned to the Atari 8-bit scene, due mostly to the high quality emulators now available, I felt like pounding out a quick game. I scrounged around for a copy of Envision-- and was rather surprised to be unable to find one.

As such, I set out to write my own version. However, this time it runs on the PC (either Linux or Windows). I can already hear the gasps of horror rippling through the crowd. Why write a design tool for the 8-bit that runs on a different platform? The answer is twofold: first, it is useful to switch between applications (i.e. the Atari Emulator, and EnvisionPC). Second, this type of editor begs to use the mouse. Since I haven't seen an emulator that uses the mouse--and since my Atari ST mice are all dead, I decided to write this a cross-platform tool.

Addendum:
Over the nearly 10 years that have elapsed since the first release of EnvisionPC, it became increasingly obvious that the program was in need of a face lift. The original version ran in a 320x200 graphics mode, which closely matched the original resolution of the Atari. Although this is the original VGA mode, many modern-day graphics cards only nominally support it... and if you are using an LCD monitor, you probably will wish that your graphics card didn't support this resolution at all! However, the biggest problem with the previous release was that I could no longer even compile the program. EnvisionPC depended on a graphics library called GRX, which worked well for DOS and SVGA Linux programs, but did not play nicely in a true windowing environment. In addition, maintenance for GRX pretty much had ground to a halt and even finding the library is a task to challenge your Google-Fu.

As such, I finally ported the program to use the SDL graphics library. (Note: It is a strange thing to look back on C code after nearly a decade gathering dust). The SDL library is currently supported by an active community, and supports both fullscreen and windowed modes.

The new release of EnvisionPC has a few command-line arguments:

```
envision [-f|-full] [-z <n>|-zoom <n>]
```

where
-f or -full: runs EnvisionPC in fullscreen mode

-z <n> or -zoom <n>: sets the zoom level of the screen. For instance, running "-z 4" magnifies the screen four times. By default, the zoom level is set to 3, which puts it pretty close to 1024x768.

Also, EnvisionPC now supports both character maps (the default) and tile maps.

Installation

The zip file includes the following files:

envision.txt: this file
envision.exe: the Windows executable, now built with MingW.
SDL.dll: the SDL library.
LGPL.txt: the license for libSDL
envision: the Linux executable

src/*.*: The source code for EnvisionPC, including Makefiles. Notice that compiling this program requires the SDL library, which is available at http://www.libsdl.org.

Usage

This editor is based loosely around memories of the original Envision program. However, since I haven't actually used the program in over 8 years, don't expect it to be the same.

The character editor screen is divided into four main sections:
Along the bottom of the screen is the font currently being edited. The current 'cell' actually being changed is highlighted by a blue square. Clicking on any letter in this section will select that letter for editing.

The right hand side of the screen is a gird...this is an enlarged version of the selected cell. Left clicking on the grid will fill in a pixel, while right clicking will erase the pixel. The mouse button can be held down for continuous drawing. Alongside the grid is an example of what the character will look like in Antic mode 6 (Graphics 1), 7 (Graphics 2) and 4.

The left hand side of the screen contains the command menu. All commands are accessible through either the keyboard or the command menu. To activate a command, click on the command button with the mouse or press the letter that is highlighted. There are multiple command menus, accessible by clicking on the tabs along the top. Notice that the keyboard commands are always active, even if the equivalent button is not displayed.
The center of the screen is a copy of the upper left-hand corner of the map. You can edit this box (and change the graphics mode) from the map editor.
The font editor can hold up to 10 font 'banks' in memory at a time. To change between banks, press the numbers 0-9 to select the appropriate font. Also, the arrows next to the bank indicator can similarly be used.

Currently the commands available in the character editor are:

'b': blank
clears the currently selected character

'i': inverse
inverts the currently selected character...this toggles the status of all pixels

'u': undo
restores the character to the state it was in before the character was selected. This command is a little twitchy, so you can undo the undo...

'a': Atari
restores the character to the original Atari character.

'A': All Atari (shift-a)
This restores the entire bank of characters to the default Atari character set.

'h': horizontal flip
flips the character horizontally

'v': vertical flip
flips the character vertically

'r': rotate
rotates the character 90 degrees counter-clock wise

'c': copy
This copies the character to another cell. Select the destination cell by clicking on it along the bottom. Notice that copying between font banks is allowed--encouraged even.

'x': extended copy
This command copies a range of characters. The source beginning cell is the current editing cell, select the ending cells, then the select the beginning destination cell. Again, copying between banks is allowed, as well as overlapping source/destinations.

't': transparent copy
This command overlays the current character onto another cell. Select the destination cell by clicking on it along the bottom. This mode is useful mainly for designing ANTIC 4 and 5 characters

'g': toggles the GTIA register <currently not implemented>

'p': poke
This command allows you to set the color registers to desired values. The colors are taken from a header file from David Firth's Atari 800 emulator. He includes a note that these colors were provided by Chris Lam. I think that they are fairly accurate, although I could have sworn that the color 70 was red. Here is seems to be a lavender. Can anyone confirm and/or deny this?

'n': numbers
This displays the values that make up the currently defined character. Pressing it one shows decimal values, pressing again shows hex values and again turns off the number display.

arrow keys: slide
The arrow keys (left/right/up/down) slide the character the corresponding direction

0-9: select font bank
The number keys select the appropriate font bank

'o': options

This is where the basic options for file input/output are set. You will be prompted for a disk image file name (or none, if you wish to use the hard drive instead), the export format, and line numbering parameters, if appropriate.

'l': load a font

Loads a font, either from the hard drive or from an .XFD disk-image

's': save a font

Saves a font, either to the hard drive or to an .XFD disk-image

'e': export a font

This command allows the font to be written to disk in a form useful for programming. You can select a range of characters to export, or the entire font. If you are writing to the hard drive, the format will be ASCII, if you are writing to an .XFD disk-image, it will be written in ATASCII

Currently available formats are:

BASIC -- written out as DATA statements, separated by commas from a given line number, incrementing by a given step.
MAE -- written out as .BY statements, separated by commas, no line numbers
MAC/65 -- written out as .BYTE statements, separated by commas from a given line number, incrementing by a given step.
Action! -- written as a byte array.

The format is set in the default options menu, accessed by the 'o' command described above. Let me know if you think there are other formats that would be useful.

'm': enters map mode

Pressing Escape or control-q will exit the program.

The Map Maker

The map screen is divided into three sections: the character palette, the command menu and the map itself. Clicking on a character will select that as the current drawing character. Left clicking on the map display will place the character at the current cursor position.

Available commands in the map editor:

'f': find

This performs a search/replace on the current map. You will be prompted for the character to search for, and it will be replaced by the current draw character. This command is effected by the current Ratio (see below). So, to replace 10% of all 'A's with 'B's, set the Ratio to 10, set the draw character to 'B', then execute this command and select 'A'.

'c': clear

This fills the entire map with the current draw character. This command is effected by the current ratio (below). A ratio of 100 will fill the entire map with the draw character. Smaller ratios will result is characters 'spotting' the map.

'r': ratio

This commands sets the ratio value for the find and clear commands. A ratio of 100 means that the character will always be replaced/set. A ratio of 10 means that it will be replaced/set 10% of the time.

'u': upper/under

In ANTIC modes 6 and 7 (graphic modes 1 and 2), there are only 64 characters available. Normally, the upper 64 are visible. This command toggles between the upper 64 and lower 64 possible characters.

't': type mode

Entering this mode allows you to move the cursor with the cursor keys. Pressing any other key will result in typing that character at the current position on the map. The ALT-key acts as the ATARI key for inverse characters. To exit this mode, press escape. (Use the draw character to place an esc symbol or arrow symbols into the map) If you are editing a tile map instead of a character map, this command will be disabled.

'd': select a draw character

This command will allow you to pick a character to use as the new draw character. Either select the desired character with the mouse, or use the keyboard to type the character. When typing a character, the ALT key acts as the ATARI key to select inverse characters. If you are editing a tile map instead of a character map, this command will prompt you to input a tile number to use for drawing instead of using a pick box.

'l': load a map
This loads a map from the hard drive/.XFD image (depending on the options set in the editor). The map file format is as follows:

ANTIC mode: 1 byte
map width: 2 bytes (low byte, then high byte)
map height: 2 bytes (low byte, then high byte)
color map: 5 bytes (PF0, PF1, PF2, PF3, PF4)
...
map data: (map width*map height) bytes
...
font data: 1024 bytes
[0]: an optional zero indicating no more data is available

Note, if you are saving a tile map, the file format is extended to
look like the following:

ANTIC mode: 1 byte
map width: 2 bytes (low byte, then high byte)
map height: 2 bytes (low byte, then high byte)
color map: 5 bytes (PF0, PF1, PF2, PF3, PF4)
...
map data: (map width*map height) bytes
...
font data: 1024 bytes
...
[1]: byte indicating a tilemap type 1 block
tile width: 2 bytes (low byte, then high byte)
tile height: 2 bytes (low byte, then high byte)
number of tiles-1: 2 bytes (low byte, then high byte)
...
tile data: (tile width * tile height * number of tiles) bytes

Currently tile width and height are constrained to be between
1-8, and the maximum number of tiles is 256.

's': save a map
Saves a .map file. Note, a tile based map file cannot currently be saved onto an Atari .XFD disk image.

'w': write a RAW map
This saves only the raw data of the map. No mode/size/font info is included.

'm': select Antic mode
Enter the desired graphics mode. Valid modes are ANTIC modes (2-7) See the chart below for details.

'z': resize a map
This resizes the map. If you are making the map smaller, map data outside the new boundaries will be lost. The map dimensions can range up to 65535x65535. The smallest size map is 1x1.

'g': go to a map position
This moves the cursor to a given position, which can be very useful when the map starts getting mind-bogglingly large.

'i': retile a map
This converts the current map to and from tile mode. If the map is currently in character mode, you will be prompted for a tile size. The character map will then be converted to a tile map, if possible (there are up to 256 unique tiles allowed - if the character data and tile size would create more than 256 unique tiles, the conversion process will fail). If a map is a tile map, it will be converted to a character map.

'b': toggle block editor
This enters the block editor (see below)

'e': returns to the editor

arrow keys move the cursor/scroll the display

Again, escape will exit the program.

The Block Editor

By default, EnvisionPC starts in character mode (you can think of this as a 1x1 tile mode). However, you can also use EnvisionPC in tile mode. A tile consists of an mxn character block. For each map, the tile size is fixed, but a tile can range anywhere from 1x1 up to 8x8.

For instance, a map might be defined as a 2x2 tile map, with tile 1 defined as:
AB
CD

Now, whenever tile 1 is drawn on the map, it will display the character block AB/CD.

When you press 'b' you enter the EnvisionPC block editor. This screen is a specialized character map editor. The screen allows you to edit a 16x16 block of tiles, each tile consisting of mxn characters. When moving the cursor in this mode, the cursor will indicate the current character being edited, and the current tile will be outlined with a hollow box. In addition, a few of the editing commands behave in a different manner:

<u>'z': resize tile</u>
When you use the resize command ('z') in the tile editor, you will be asked to set the tile size rather than the map size. A tile can range anywhere from 1x1 (which revert the editor to character maps) to 8x8 tiles. Typical tile sizes are 2x2 and 3x2.

<u>'g': go to a tile definition</u>
This will take you to a specific tile number, rather than an X and Y position

<u>'i': retile</u> (disabled on the block editor map)

<u>'b': toggle block mode to return to map mode</u>

In addition, the header display will also show different information. It will indicate the current tile size, and which tile you are currently editing.

When a tile definition is changed on this screen, all instances of the tile will be updated on the map screen.

Tile mode walk-through:

1) Start-up EnvisionPC, and click on the title screen to enter Edit Mode.
2) From Edit Mode, press 'm' to enter Map Mode.
3) From Map Mode, press 'b' to enter Tile Mode ('B' for blocks, I need to fix the hot-keys!)
4) In Tile Mode, press 'z' to define the tile size. If the tile size is ever larger than 1x1, then the map mode will use tiles instead of characters. In this case, enter a width of 2 and a height of 2.
5) In Tile Mode, use the same drawing methods as in map mode - notice however that now you are actually defining the tiles. The upper left corner is tile 0. The tiles are laid out in a 16x16 grid.
6) Define tile 1 as something more interesting then blank space.
7) press 'b' again to toggle block mode. You are now back in Map Mode, but notice a few things are different. The status bar now displays the drawing tile instead of the drawing character. Also, your cursor is now the size of the tile, rather than the size of each individual character.
8) Draw in Map Mode using the current draw tile.

Disclaimer:

I provide no guarantees what-so-ever with this program. I have had some problems with the .XFD writing/reading routines--so be sure to back up your .XFD images before trying to read and write to them. You have been warned. If you find problems, or would like to see enhancements/fixes to the program contact me via the e-mail address below.

Credits:

- The .XFD disk routines are based on the xfd_tools package written by Ivo van Poorten <ipoorten@cs.vu.nl>
- The bitmap rotation routine is taken from a Graphics Gems II article by Ken Yap entitled "A Fast 90-Degree Bitmap Rotator" (pp. 84-85)
- The color header file containing RGB values for each Atari color was written by Chris Lam <lamcw@sun.aston.ac.uk>
- EnvisionPC is now based on the SDL library. libSDL is distributed under the LGPL license. The library and full source can be found at http://www.libsdl.org.

Mark Schmelzenbach
e-mail: schmelze@cs.utah.edu
08/04/97, revised 2/28/2006