

The Tale of SFLIU

It was the summer of 1992, I graduated from high school and stuck a job working in a computer store as sales/technician. My pal Brian would visit often, biking for miles each way to reach the store. (Boy we had energy back then) and I would slip some time out of work to find various ways for being unproductive an hour or two at a time, playing computer games and what not. ^_^;

One day, we were trying out this "new" SF2IBM (v1.9) game when Brian talked me into editing this game. Basically SF2IBM is a home made Street Fighter 2 clone featuring absolutely no gameplay resemblance to the real thing because of a lack of gameplay programming. The way the game's data is constructed allowed easy adjustments to gameplay and animation sequencing, plus other gameplay related attributes. A number of people on the net had then altered the game to introduce wacky moves much similar to the hacked version of the SF2 Champion Edition arcade games, though the fundamentals remained absent: jumps would last for only a fraction of a second, roundhouse kicks would have no range and has the same recovery time as a jab, basically every single aspect of what the original game was about were not there. With an immense amount of free time to spare/waste, I decided to take on the task in making this game resemble the original Street Fighter 2.

At first there wasn't the idea in mimicking the original game. I just spent a few hours making my version of wacky moves, like a special move that will automatically throw in multiple hits plus a Dragon Punch. Though the "pretty cool, huh?" aspect didn't last for long and it certainly didn't impress Brian. Then, for whatever reason, I found myself putting in 10 hour days in this crazy project of putting together a real SF2 clone.

Making a patch turned out to be a lot more fun than expected. More importantly, I learned a lot about the mechanics of the game itself. First I gone through the 80 to 100 frames that each character has, and adjusted positioning of each frame in every animation, then it was fun with sequencing, defining how long each move last and grouping them together to make them look natural, as various attributes were added for defining interruptable frames, speed, and damage. One of the most interesting attribute was the hit range. To define the fact that only a character's fist should cause damage as opposed to, say, the head when a character is throwing a punch. With the fighting game term "priority" in effect, I had to tune various moves so that more committed moves like a roundhouse kick would have more priority over a medium kick, for example. Also need to make sure that all standing attacks would miss characters that are cruched. The stack of notes I wrote during the making of the game were rather impressive as a report on how the game SF2 operates. Later on in talking to experienced programmers I had a chance to confirm all my speculations as to how the game must have been programmed, and that was pretty satisfying too.

Thanks to Brian who later joined me in making what's known today as the SFLIU version of the game, version 1.0 was released to Internet. It wasn't until 1993 that I had internet access, so Brian was kind enough to distribute the game for me, and boy it was a real phenomenum for a high school graduate. Hundreds of e-mail were written to us with nothing but just compliments on the game. This 200+ hour project made just so Brian (which didn't have a Super Nintendo back then to play SF2 with) can play SF2 at home paid off in a big way. We had fun playing SF2 on 386's, and so did thousands of other SF2 fans. Due to the numerous responses we were getting, different updates of the patch were released. Alternations and additions from gameplay to graphics made were pretty scary come to think about it. I had not only added more moves to reflect the newer SF2 versions but also added the characters that were missing in the original SF2IBM game. Loading a hundred frames of animation in Photoshop with my 386 with 8 megs of RAM (or was it 4?) was NOT fun. ^_^; But with the responses that we had gotten it was an extremely satisfying project, not to mention the experience we obtained in producing the game.

Tiny SFLIU FAQ

Q: Where can you get the game?

A: It's been more than 7 years since I stopped working on SFLIU, and finally I have my own server to place my junk into. Here is the "latest" version of SFLIU. Enjoy.

- [SFLIU 2.10 disk 1](#)
- [SFLIU 2.10 disk 2](#)
- [SFLIU 2.10 disk 3](#)
- [SFLIU 2.20 update](#)

Q: What's the newest version?

A: Version 2.2

Q: Will there be a new version?

A: Nope, we had both gotten full time jobs since then, now free time is no longer a commodity.

Q: How do you edit the game?

A: First you must know that half the data of the game is in ASCII and the other half is in Binary. The file types are listed below:

1. SEQ
2. IDE
3. KEY
4. RE
5. BK, or extensions with numbers
6. R & ID

SEQ Files

SEQ files are the main juice of every SF2IBM patch. It holds animation sequences as well as their parameters. The first part of the SEQ file consists of a maximum number of 128 lines. The first number of each line always represents the sequence number. The line is terminated by a -1, and usually the number just before that specifies the sequence number to jump to. So looking at any random .SEQ file, you'll see sequence 0 (standing animation) is specified to jump back to itself (hence, an infinite standing animation). The rest of the numbers specify each individual frame. Its properties are specified in the second part of the SEQ file. So which line number represents what? Sorry, but I never took the time of writing them down, although Stan Warman has written a pretty comprehensive guide on that and many other info for editing SF2IBM. Sorry, but for the life of me I forgot his E-mail address. But the meaning of each sequence number isn't that hard to find, just change the lines and see what goes wrong. ^ _ ^;

The second part of the SEQ are frame specifiers. They generally follow the format:

```
10 50 5 -4 A12N1Ax.
A B C D EFGHIJK
```

- A - Frame number
- B - Image number
- C - X movement
- D - Y movement
- E - Frame type (M = move, A = attack, D = Defense, F = Fall, M = Move)
- F - Hit damage (octal)
- G - Hit reaction (octal)
- H - Frame orientation (N = normal, R = reverse, F = flipped, U = upside down)
- I - Frame interruption (octal)
- J - Attack mode (A = frame hits, x = frame does not hit)
- K - Invulnerable (A = frame is invulnerable, x = frame is not invulnerable)

IDE files

IDE files also deal with individual frame properties, but are more low level and generic. IDE files are composed of three sections: Frame Size, Frame Orientation, Frame Collision.

Frame Size - Used so SF2IBM can separate each sprite from the .RE library.

Unless you're creating a new character, don't screw with the numbers. Unless you want your computer to crash.

Frame Orientation - Controls the placement of the sprite on the screen.

In the format: 10 50 -40 -1 (Defined as: A B C D)

- A - Image number
- B - X shift
- C - Y shift
- D - Delimiter

Frame Collision - Controls the hit range of the sprite.

In the format: 10 -10 20 30 (Defined as: A B C D)

- A - X hit range begin
- B - Y hit range begin
- C - X hit range end
- D - Y hit range end

Each sprite is a Cartesian co-ordinate, with (0,0) at the top left corner of a sprite, facing right. Positive X is right, positive Y is down.

KEY files

KEY files are the simplest of the three text data files. They specify the movements for special moves and define throws/grabs. The first number on the first line is the throw range, in pixels. It also defines the range where the close/far attacks are invoked. The KEY files are divided into two parts. The first specifies movements for the special moves.

The typical format is: 1fdf 45 foo.voc (Defined as: A B C)

- A - Motion, specified backwards. In this example, it means forward - down - forward - 1 (jab)
- B - Sequence number of the special move
- C - Sound file to play (_no_voice for no sound)

The second part specifies the throws.

The format goes: 41 36 46 90 1 g3.voc (A B C D E F)

- A - First number is damage, second number is slam type. (1=normal) (3=extended range)
- B - First number is direction (left=3) (right=6)
second number is the button (Fierce=6) (Roundhouse=3)
- C - Character's SEQ sequence to play
- D - Opponent's seq sequence to play
- E - 0 = character slams forward, 1 = character slams backwards
- F - Again, the sound effect.

RE File (revised 07/20/95)

RE stands for Runlength Encoded. Similar formats exist in Microsoft's RLE format with the same name. Basically, instead of defining colors pixel by pixel (no compression), it scans horizontally for repeating colors, and records in how many times the color repeats for. So if you have images with large areas filled with the same color, RE does a good job of decoding.

SF2IBM's RE are composed of three bytes in each group. An example would be "051400<20 bytes of data>". Where the first byte, 05, means that there are 5 pixels of black space (transparent color) first, then the 5 pixels are followed by hexadecimal 14 (20 pixel) of data, and the data are what's specified right after the next byte. The third byte is the "second digit", 1 byte = 255 pixels, but you'll never need it for SF2IBM, so it's always zero.

To convert between RE and other formats, you'll need Ben Cantrick's conversion utility. I'll see if I can stuff it in brawl.mindlink.net.

BK, or extensions with numbers

BK files, and any files ending with numbers as extensions are simple RAW files. Though simple, RAW files are rarely supported by graphic editing utilities because they don't contain any information at all about their sizes and their palette information. The only program I know of that can read RAW files is Adobe Photoshop. Definitely not a shareware for this one, it's retail price is about \$400 for the old version. Try to look for an old copy of Photoshop V2.5 LE (lite version). That's alot cheaper.

Horizontal pixels multiplied by vertial pixels makes the size of each RAW file. All backgrounds are 520 pixels wide, and the extensions in numbers are the horizontal pixels of that image.

The hard part is creating the palette. I'll put the palette file here upon anyone's request.

Now you can edit the background and the logo. Heh, Brian's brother was impressed when I wrote a big "Lanzer Rulz" over Ryu's background when I first figured out how to edit stuff. :)

The R format and the ID format

R and ID files are used exclusively for the character portraits. R are actually multiple RAW files, with the ID file, in text format, explains the dimensions and positions of each section.

The ID format is pretty similar to the IDE format. First part defines the dimensions, and the second part defines the positioning. All units are in pixels. Multiply the dimensions and you can get the exact sizes for each picture. Use that to split the R file up for editing.

The ending files, end.id, etc. Were included with the original SF2IBM. But too bad the EXE never called the files out. All that's there are the lame messages when you beat all the computer opponents.

^ _ ^;;

That's about it. Hope this little guide helps those who are planning to muck around with SF2IBM. Have fun!

[Back to video game page](#)

[Back to main page](#)