Encoding TASes with FCEUX

Prerequisites

FFmpeg: Installation for Windows and Linux

For Mac users, You would first have to install Homebrew, then run

```
$ brew install ffmpeg
```

in the Terminal.

FCEUX: Download Link

For Windows users, download and extract the Win32 binary, and for Linux and Mac users download the latest development build if you do not want to build it yourself.

Optional but highly recommended on Windows - Lagarith: Download Link

Download and open the installer to install.

You should also have a significant amount of storage left depending on the length of your TAS.

Dumping a raw AVI file

Open the TAS editor for the most convenient dumping process.

If you would like just the gameplay without any HUD elements, disable HUD recording in File > AVI/WAV. If you would like the HUD elements, configure the HUD via Config > Display and enable HUD recording.

Trim your movie until you are satisfied with the endpoint, pick a starting point (First frame for most cases) and go to File > AVI/WAV and press Record AVI. Save the file wherever you want and when asked for video compression, select Lagarith (or Uncompressed, if you have not installed it).

Play back the whole movie (Turbo may cause issues, such as skipped frames or pitched audio), and when its done, go back to File > AVI/WAV and press Stop AVI.

Before Encoding

Navigate to Config > Sound, and make sure all volume sliders are maxed, and quality is set to Highest. You can also change the colour palette in Config > Palette.

Locate the directory containing your AVI file(s), and

If you see one file, skip ahead to "Encoding the final video".

If you see multiple files, continue to "Merging videos".

Merging videos

If several videos have appeared in the directory you have selected, named

```
video.avi
video_part2.avi
video_part3.avi
.
.
```

First create a text file named `input.txt`, with the following contents:

```
file 'Drive:\path\to\file\video.avi'
file 'Drive:\path\to\file\video_part2.avi'
file 'Drive:\path\to\file\video_part3.avi'
.
.
```

And (IMPORTANT!) make sure the text file is encoded in UTF-8.

Then open any command line in the directory containing the text file (in this case the same directory as the video files) then run

```
ffmpeg -f concat -safe 0 -i input.txt -c:v ffv1 -c:a copy input.mkv
```

This creates a new lossless video file named `input.mkv` containing the entire movie. You may now delete the original dumped videos, as they are not necessary anymore.

Encoding the final video

If your movie is short enough for it to fit in a single dumped video file (or you are using Lagarith), rename the video to `input` and run

```
ffmpeg -i input.avi -c:v libx264 -pix_fmt yuv420p -c:a copy -vf
scale=iw*FACTOR:ih*FACTOR -sws_flags neighbor output.mkv
```

Likewise, if you have just merged several video files, run

```
ffmpeg -i input.mkv -c:v libx264 -pix_fmt yuv420p -c:a copy -vf
scale=iw*FACTOR:ih*FACTOR -sws_flags neighbor output.mkv
```

where `FACTOR` is the scaling factor in which you want to scale your video. 4 is the minimum recommended value, which corresponds to around 1080p.

If for some reason your scaling factor is smaller than 4, omit `-pix_fmt yuv420p` for the best quality (however keep it if it's destined to be uploaded to Discord).

To set the aspect ratio, you can add `-aspect X:Y` after specifying the input.

Additionally, if you would like higher quality (particularly for lower resolution encodes), add `-crf 18` after `-c:v libx264`.
Note if you are using Lagarith: FFmpeg may throw errors at the end of encoding, this should always be fine as we are using the matroska (mkv) container.

Additional content

If you would like additional content besides the gameplay (Lua input display/stats display or even another instance of gameplay if you a doing a comparison), you can horizontally or vertically stack two videos using

```
ffmpeg -i left.mkv -i right.mkv -filter_complex hstack -c:v ffv1 -c:a copy output.mkv
```

or

```
ffmpeg -i top.mkv -i bottom.mkv -filter_complex vstack -c:v ffv1 -c:a copy output.mkv
```

respectively. For more complex stacking, see this Stack Overflow thread.
Afterwards, if needed, you can crop the video before upscaling using ffmpeg's crop filter:

```
ffmpeg -i input.mkv -filter:v "crop=WIDTH:HEIGHT:XCOORD:YCOORD" -c:v libx264 -c:a copy -vf scale=iw*FACTOR:ih*FACTOR -sws_flags neighbor output.mkv
```

where `WIDTH:HEIGHT` is the width and height of the cropped video, `XCOORD:YCOORD` is the position of the top left corner of the cropped video.

More complicated encodes

For more complicated encodes such as a comparison video which includes a timer, time difference and statistics, I would recommend a free video editor such as Shotcut or DaVinci Resolve (Resolve offers a free version, however be warned that Resolve doesn't support matroska).
Configure the video to 60.098814fps (or 50.006979fps for PAL) progressive scan, and configure the audio to pcm_s16 CBR 768kbps.