FP-GAme Requirements Document - Updated 2/20/21

The main goals of our project are both representing the hardware available at the time period we're replicating and the providing a more accessible and safe (in the sense of application development) developing environment for that hardware. Thus, most of our requirements stem from one of these two goals.

CPU Requirements:

- A modern processing core which can easily be compiled to. Should operate at least at 16MHz.
    - 16MHz is the clock speed of the GBA, which seems a good metric.
- Memory protection to support a kernel.
    - If we don't have this, the user can damage the inner workings of the kernel, meaning we provide no additional safety.
- At least 512KB of ram.
    - The GBA offered 256KB + 32KB, but times have changed and more ram is better. Modern programmers may be more comfortable with something closer to ~2MB.
- Support for configurable timing interrupts.
    - Necessary to decouple game timing from the graphics card. If we don't do this, the abstraction from the graphics hardware will be far more fuzzy.
- Support for non-maskable interrupts from an external chip.
    - The PPU will likely wish to send out frame updates. The APU will likely wish to send out requests for more wave data.
- Support for some *sufficiently fast* form of communication with an external chip.
    - Must be able to meet timing requirements for both the sound cards sample updates and the graphics cards scrolling/frame/sprite updates. Graphics card will be the limiting factor of this. This is an interacting requirement with the graphics card design. Our choice in communication method must be considered in our final design, though we can likely calculate some lower bound.

Notes about the CPU:
Finding a CPU with a sufficient amount of RAM is unlikely to be difficult. Even the teensy meets this requirement (512KB of fast ram + 512KB of slow ram). However, the CPU must be considered with our software requirements as well. For example, should we include a kernel, size would be of utmost concern. Say we're using a teensy, then our ROM size is 2MB. This is half the size of the smallest GBA cart, and our users will also have to contend with our kernel; the GBA was programmed using bare metal. The tightness of space could be remedied by allowing our storage space to be expanded, say with an SD card. However, then our kernel would have to implement demand paging to deal with the slow access times. We simply do not have the time to create a kernel with those capabilities. Creating a kernel with extensive virtual memory capabilities is time consuming and well outside the scope of our project.

PPU Requirements:

- An output resolution of 640x480, which may be scaled up from a smaller resolution (at least 320x240).
    - Most modern televisions do not support resolutions lower than 480p if converting to HDMI, and so we must support at least that to ensure the best compatibility. Consoles of the time often supported resolutions around 256x240, with the actual resolutions varying quite a bit between consoles. Even consoles like the SNES, however, never hit true 480p, and so we decided it was best to require a scaled up image as our minimum. 320x240 was selected as it is the closest fit within the expected range while also obeying a 4:3 aspect ratio.
    - http://tinyvga.com/vga-timing/640x480@60Hz
- Support for generating tile based images. Tiles must be at least 8x8 in size and the PPU must support holding at least 1024 unique tiles in its memory
    - 8x8 is the standard size for most PPUs. Consoles of the time varied wildly on the number of tiles they could hold in memory, with the low end being around 512 and the high end being around 3072. 1024 was selected as our minimum, because our video output requirement specifies a larger screen size then was required by the consoles of the time, so we require more tiles.
- Each tile should support 16 different colors, with color palettes being swappable per tile.
    - Typically, consoles with this kind of PPU would vary between 4 and 256 different colors for each pixel, with those lower than 256 using palettes. Suggest 16 colors for tiles, as each tile is only 8x8 in depth, so more seems unnecessary. Having less would severely limit the color variety of what our users can display.
- Support for a background and sprite layer
    - Pixel processing engines from the time period we are simulating use this system. With sprites controlled using OAM, and backgrounds displayed from tile data.
- Ability to update OAM for every sprite every frame.
    - Game developers want to be able to change the positions/color palettes of every character on screen, for example. To be unable to do so is extremely limiting for game development.
- Vertical and horizontal hardware scrolling of the background plane.
    - Games like platformers cannot function without this functionality. Only the most static games, e.g. arcade games like galaga, pacman, and donkey kong, can function without scrolling. Our customers will most likely want to develop more complex software than this.
- Video output color depth of at least 12bpp
    - Visually, 12bpp 16bpp and 24bpp aren't too different (see link below). We decided to set 12bpp as our minimum as it is just under the color depth offered by the SNES and GBA (15bpp) while still being great than the depth offered by the NES (which is simply a 64-color palette).
    - https://www.cambridgeincolour.com/tutorials/bit-depth.htm

- A sprite layer, supporting at least 64 sprites with the ability to draw at least 16 sprites per scanline and layer sprites and in an arbitrary order, with sprites either behind or in front of the background layer.
    - 64 sprites is the number seen most often among older consoles. Typically, that number isn't far exceeded as it's unnecessary to do so. Consoles like the NES could only display 8 sprites per scanline, and suffered for it. We want to beat that number, as our scan lines will be larger in pixels. The genesis is a console which supports the same resolution as we require, and it supported about 20 sprites per scanline. Therefore, we decided on 16 as a nice inbetween. Layering sprites is between each other and between background tile layers is another function most PPUs support, and so we require it as well.

Audio Requirements:

- Mono output to stereo headphone jack.
    - Offered by basically all consoles, it allows the user to connect a speaker alongside VGA, which makes for a nice and easy to setup development environment.
- 8kHz, 8-bit PCM channel
    - Simple to implement and was the main sound advancement offered by the GBA over the original Game Boy. Note that the GBA offered 16KHz and 32KHz as well as stereo, though these often went unused and stereo output could not work on the console's speaker.

Software Requirements:

- A simple kernel to launch a user mode process
    - If our goal is to provide a nice dev environment, it's best we abstract away kernel mode hardware access.
- Support for the standard C libraries
    - Yeah. Hard for the user to do much without C libraries.
- System calls for interfacing with the graphics and audio hardware in an efficient way.
    - The user should be able to write code which is mostly abstracted away from communications protocols (SPI, I2C, etc.). We want them to be able to write games without writing drivers.
- System calls to support some kind of timing information (access to a timer interrupt, or perhaps WFI/HALT).

Input Requirements:

- Minimum of 6 buttons + 4 directional buttons (D-PAD)
    - GBA offers this.

Other considered ideas. Currently out of scope, but left in for reference.

APU (sound chip):

- User should be able to set the frequency of 2 square wave channels.
    - Common Audio requirement for retro consoles such as the Game Boy
- User should be able to turn on/off a noise channel
    - Similar audio to the Game Body and GBA.
- User should be able to load in an arbitrary 32-sample 4-bit waveform and play (arbitrary wave channel)
    - Similar audio to the Game Body and GBA.
- Digitally synthesized audio should have bit-depth of at least 4-bits (8 bit preferred).
    - Similar audio to the Game Body and GBA.