LIKO-12

★ Stage one: Documenting the current API:
- Extend the generator script to generate events and DiskOS APIS
- Write tutorials for LIKO-12
- Finish the introduction
- Make the documentation available in LIKO-12

★ Stage two: Cleanup and rewrite some peripherals.
★ The GPU:
- Make it possible to load and write binary encoded images.
- Make it possible to set the default palette using the BIOS.
- The cursors should be stored in objects instead of a global list.
- Create fonts support.
- Clean up and optamize.

★ The HDD:
- Make it possible to read captured GIFs and Screenshots, by mounting them into a special drive.
- Add virtual drives support.
- Add support for file objects

* The FDD
    - Make sure it's possible to read pico8 cartiges.
    - Add support for writing data into GIFs.

* The touch controls.
    - They have to be complitely rewritten.
    - The joystick has to be made cleaner.
    - Add multiple styles support.
    - Make the default position closer to the device corners.
    - Give much customization support for both the user and the OS/games.
    - Add support for adding extra buttons.
    - Add custom on-screen keyboard to solve the keyboard events problem and make it possible to reach iOS.
    - It shouldn't feel limited but also not give love apis out :P

* The RAM.
    - Look for optimisation possabilities
    - Clean it up.

* The keyboard.
  - Add touch keyboard.
  - Maybe add gpio key support (RPi)
  - Keys simulation api.

* The GPIO:
  - Available only on the raspberry pi.
  - Depends on Lua-Periphery.

* The BIOS:
  - Add apis for reading the current Bios settings
  - Give the user the ability to hide the Bios settings from the OS.
  - Add bootloader support
  - Add .zip, .png, .lk12, and web hosted OS (like github) installation support in the OS Installer.
  - Add the ability to tweak the peripherals settings and invent your own fantasy console.
  - Add palettes editor.
  - Add the option to enable a special debug peripheral.
  - Make it possible to change the boot drive.
  - Add the option to update LIKO-12 on the fly.

★ Stage three: Rebuild the operating system
this stage will have some non-connected ideas
and won't be listed like the 2 previous stage.

- ★ Support all the resolutions.
- ★ Add the ability to use less colors
  (add support for 1-bit displays).
- ★ Add OS updates system.
- ★ Make the system modular.

$$\begin{tikzcd}
\text{DiskOS} \arrow[r] \arrow[ddr] & \text{boot.lua} \\
 & \text{builds the global environment for the OS} \\
 & \text{terminal}
\end{tikzcd}$$

- ★ The terminal should apply to some of the ASCII standard
- ★ The Editors shouldn't be connected with the terminal
  but instead there would be a command for binding
  keys into specific functions.
- ★ The runtime should be made very easy to port
  and update into other operating systems.
- ★ The proses of rebuilding LIKO-12's OS will go through
  PoorOS $\longrightarrow$ DiskOS-Light $\longrightarrow$ DiskOS $\longrightarrow$ GlassOS

PoorOS: This will be an operating system made out from a single file, that makes it easy to bootup without installation.

- It will simply provide a Lua interpreter. That allows it to be the minimum tool required to rebuild everything using LIKO-12 itself.

DiskOS-Lite: This will be the base of the new DiskOS. It will define an important structure that's good enough to be used for building other operating systems, like "distros"

[illegible]

~~- LIKO Peripherals APIs / globalizer~~
  - ~~Add an option to block some peripherals from being a global.~~
  - ~~Add an option to make a periphal functions to be available in the root of the globals table (Like the CPU).~~

~~- Re-build most of the Lua standard library.~~
~~- Run the sorted scripts in Boot/~~

```
C:/ ──> boot.lua
  └──> Boot/ ──> 01_io.lua => rebuild the io library
              └──> 02_package.lua => rebuild the package library
```

* It would be more wise to use a 3 digits prefex * instead of 2, and count to by 10, because that will allow others to easily add scripts between them

boot.lua ⇒ List the scripts in the C:/Boot/ directory
Sort them, exeute them inorder passing
them the arguments passed by the BIOS.

Boot/01_io.lua ⇒ Recreates the standard lua io library
Starting from the HDD api.
This allows standard lua scripts to work
in LIKO-12.

Boot/02_package.lua ⇒ Recreate the standard lua package library
restoring the require, loadstring, loadfile
and module functions.

Boot/03_lfs.lua ⇒ Recreate the luafilesystem library using HDDapi.
Since most standard lua scripts that work with
filesystems usually need it.

Boot/04_os.lua ⇒ Restore most of the standard os library functions
that actually work in the LIKO-12's os
like os.execute, but insteads of executing binary
programs, it would execute lua or lua_asm programs.

Boot/05_apis.lua ⇒ This would load the special apis provided
by DiskOS in the C:/APIS/ directories.

Boot/06_shell.lua ⇒ This is the final script, and it won't ever
reach the end, otherwise the boot.lua would
finish executing the scripts and shutdown.
It will run the shell program that the user
interacts with.

Boot/00_peripherals.lua ⇒ It's responsible for globalizing the
peripherals functions.

GamediskOS only:

Boot/06-Shell.lua ⇒ Boot/06-game.lua

This script is responsible for loading the fused game and playing it.

DiskOS:

Boot/051-sheet.lua ⇒ loads the system sheet and expose it as a global.

Boot/052-Cursors.lua ⇒ loads the system cursors.

Boot/054-editors.lua ⇒ loads the editors.

Boot/055-binds.lua ⇒ bind tasks into hotkeys, like the editors to the escape key

* note: the terminal api is loaded earlier, maybe in Boot/041-shellapi.lua

Boot/001-preferences.lua ⇒ it would load some os an user preferences and expose them as globals so the can alter some of the bootup functions.

Boot/053-fonts.lua ⇒ Load the fonts of the operating system.

* maybe there should be a palettes api.