Link : https://oblerion.itch.io/gba-engine ver a1.6 2024

Engine GBA

The gameboy advance like engine
by Magnus Oblerion

Link : https://oblerion.itch.io/gba-engine ver a1.6 2024

Describe :

2d game engine made with C, raylib.h and lua.h .
egba = Tic80 api + 32bits upgrade + modern GPU optimization.

It's the synthesis of 7 years of game and framework creation,
which is why (exceptionally) the sources are closed.

Features:

- UI Projet browser (add/delete/load project)
- UI Project (edit/debug/run/export/import)
- UI Sprite viewer (view/save/load-delete palette/load spritesheet)
- UI Script viewer (view)
- locked cartbridge
- Color Theme (white/black/blue/red/green/yellow)
- Run egba and lua in CLI
- Save-state 100 bools, 50 string, 50 number

Spec:

- 100 projets max in browser
- 5 palette color : 32 colors + alpha
- 256 sprite : 16x16
- lua script : 64000 char

Fast start :

Setup new project

- Download palette image .png <= 32 color lospec.com
- on browser touch « + » , it create new.lua and new.png
- click on « new » project, click on « spr » button
- drag and drop palette image to window
- save, quit and rename new .lua/.png with your projet name

Debug (GUI)

- load projet with browser
- press space for start/stop debug

Link : https://oblerion.itch.io/gba-engine ver a1.6 2024

Create EGBA from script/sprite

- After load projet , « -> » button or ctrl + e
- ctrl + l for locked egba (can’t import lua/spr from egba)

Import sprite/script from EGBA

After load projet , « <- » button or ctrl + i

Run EGBA (GUI)

After load projet, press enter for start/stop running EGBA

Debug (CLI)

- open cmd, tape it

```
egba.exe project.lua
```

it use .png for load color and sprite

Run (CLI)

- open cmd, tape it

```
egba.exe project.egba
```

Load spritesheet

- drag and drop 256x256 img, it to browser→project→spr
- drag and drop 256x257 img, it to browser→project→spr

Link : https://oblerion.itch.io/gba-engine ver a1.6 2024

Lua api :

Main

loop function

```lua
function EGBA()
-- your code run 60/seconds
end
```

log

```lua
trace("hi")
-- print hi in console (only string)

local v=2 print("value :",v)
-- lua print function can be used to view variable value in console
```

palette swap

```lua
pal(2)
-- id 0 -> 4, change curant palette
-- for all function after with id color
```

Graphics

clear screen

```lua
cls(0)
-- clear screen with id color 0
```

print text

```lua
text("hello world",23,23,0,16)
-- x 23, y 23, id color 0,font size 16
```

draw rectangle

```lua
rect(23,23,50,50,2)
-- draw fill rectangle in x 23, y 23, width 50, height 50, id color 2
rectb(23,23,50,50,2)
-- draw line rectangle in x 23, y 23, width 50, height 50, id color 2
```

draw pixel

```lua
pix(10,10,2)
-- draw pixel in x 10,y 10, id color 2
```

draw sprite

```lua
spr(1,23,23,2)
-- draw sprite id 1,x 23,y 23,scale 2
```

Input

keyboard

```
if btn(0) then
-- if key down is down
elseif btn(1) then
-- if key up is down
end
if btnp(2) then
-- if key left is pressed
elseif btnp(3) then
-- if key right is pressed
end
-- id 0 : down, 1 : up, 2 : left, 3 : right, 4 : x, 5 : c
-- btn -> btn is down
-- btnp -> btn is pressed
```

mouse

```
local x,y,btnl,btnm,btnr = mouse()
-- x = x mouse
-- y = y mouse
-- btnl = true/false mouse left button
-- btnm = true/false mouse mid button
-- btnr = true/false mouse right button
```

Save-state (.sav file)

This is static data, it keep value when quit.

Write save

```
wsave_numb(id,100)
-- id 0 → 50
wsave_string(id,"name")
-- id 0 → 50, string 35 char
wsave_bool(id,true)
-- id 0 → 100
```

Read save

```
local i_num = rsave_numb(id)
-- id 0 → 50
local i_string = rsave_string(id)
-- id 0 → 50
local i_bool = rsave_bool(id)
-- id 0 → 100
```

Table index