Tutorial

VLF Setup

The VLF Mainloop

Custom Waves

Text with shadow

The Central menu

The Scroll bar

The Titlebar

Tasks and VLF.FadeObjects()

VLF Mp3 Player

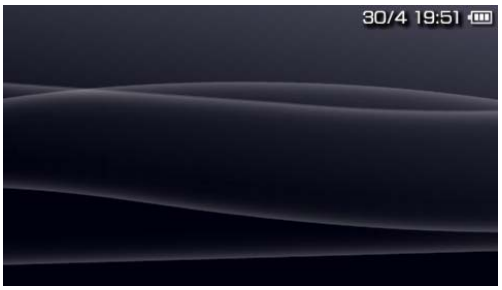
* means optional (don't put it into your code!)

Part 1 VLF Setup

if you want to use the vlf lib in one of your projects,
just copy the 3 files VLF.lua, theme.vlf and main.vlf that are contained in the download
to the folder of your project.

Part 2 The VLF mainloop

Screen:



first we use the following commands

```
require("VLF")
VLF.init()
```

the first one loads the vlf library
and the 2. initialises the vlf lib
now you can use all the commands the library provides

next we are going to load a background,
which is done with the following function

```
VLF.GetBG(BG*,file*,set_img_color*)
```

none of the options is mandatorly, but we'll define them anyway [*]

to choose the background that is meant to be loaded you can define backgrounds from 1-30 (1-12 fat | 13-30 slim)
is there no background defined, that means the clips are empty,
then the background will be loaded depending on the current month.
here we choose background 28.

all backgrounds are actually stored in the file "theme.vlf" which should be located in the folder of your application.
if you placed the file on another directory you have to submit the respective path.
in our case the path is "theme.vlf"

by using set_img_color you can decide whether the pictures used by the library should be adapted to the background or not. (battery, checkbox and textbox-shadow)
true ja | false nein wir entscheiden uns für true

so the command looks like this:

```
VLF.GetBG(28,"theme.vlf",true)
```

and the code (including the mainloop) should now look like this: code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

while pge.running() do
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()

pge.gfx.enddrawing()
pge.gfx.swapbuffers()
end
```

to show the background, the xmb-wave, the battery-symbol and the clock
we will put the following function

VLF.DrawFrame()

into the mainloop

however, if you want higher flexibility of the functions (eg. wave speed setup) you have to call each function separately
which ones is written in the documentation.

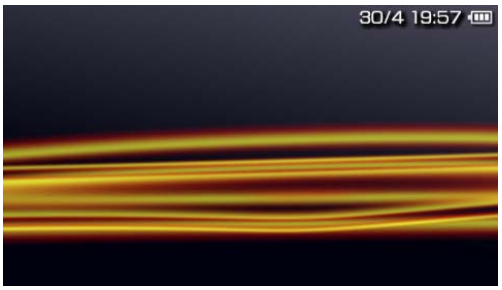
code:
```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

while pge.running() do
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
end
```

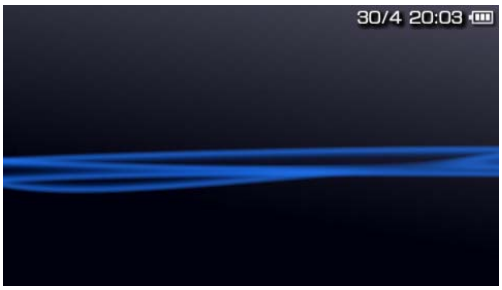
now you could already execute the code.

Part 3 Custom Waves

If you want to use custom waves instead of the original ones you can load them with this function
VLF.SetCustomWaves(path,key) --only use key if the wave has been packed with a changed key
(the contained ones have been packed with standard keys).
Call the function before the mainloop.
E.G.
code:
```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)
VLF.SetCustomWave("blue wave.wv")—loads the blue custom wave
```

screens:





Part 4 Using text with shadow

As you propably want to show more than only the wave
we will put some text on the screen

screen:

the function looks like this:
VLF.DrawText(x,y,font,text,distance*,radius*,color*,shadowcolor*,selected*,fade*)

to make it easier we will abandon the last to parameters (description in the documentation)

x: to place the text in the middle of the screen we will write VLF_CENTER instead of the x coordinate

y: we defined this value as 130 (which should also be the middle of the screen)

font: we are using the standarf font of the vlf lib VLF.Font (you might also use VLF.Font_small or VLF.Font_huge)
btw: you don't need to activate the font

text: now the text that is meant to be shown is following: "VLF Tutorial"

distance, radius : to show a shadow we have to define these two parameters
the first one sets the distance of the shadow to text and the second one sets the size of the shadow (radius)
in our case the distance is 2 pixels
and the radius is 3 (if there are speed bumps you should decrease the value of radius)

color: here the color of our text will be the standard color VLF.Text_Color

shadowcolor: and the shadow color will also be standard one VLF.Shadow_Color (the color will be adapted to the background)

now the function looks the following:

VLF.DrawText(VLF_CENTER,130,VLF.Font,"VLF Tutorial",2,3,VLF.Text_Color,VLF.Shadow_Color)

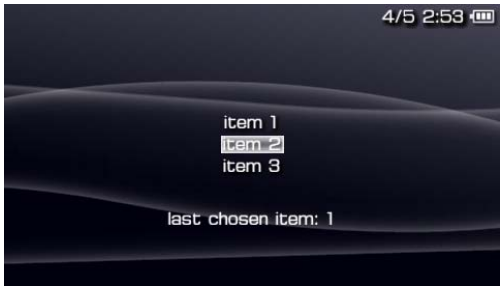
...this we are going to put after VLF.DrawFrame() into the mainloop and the application is done.

code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

while pge.running() do
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
VLF.DrawText(VLF_CENTER,130,VLF.Font,"VLF Tutorial",2,3,VLF.Text_Color,VLF.Shadow_Color)
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
end
```

Part 5 Using the Central Menu

Screen:



we start once again with the mainloop...

code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

while pge.running() do
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
end
```

first we have to define a table in order to use the central menu,
that contains all the items to be shown...

ant it could look like this:

```
table = {
"item 1",
"item 2",
"item 3"
}
```

because we want to print a text that shows which item is selected,
we need to define another variable (last_item)

so we add:

```
last_item = ""
```

this we'll put before the mainloop

code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

menu = {
"item 1",
"item 2",
"item 3"
}

last_item = ""

while pge.running() do
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
pge.gfx.enddrawing()
```

```
pge.gfx.swapbuffers()
end
```

now we'll care about the actual function
```
VLF.DrawCentralMenu(table,selected*,font*,align*,spacing*,x*,y*,object_limit*)
```

for simplicity reasons we will only use 4 parameters
all the other ones are explained in the documentation.

```
VLF.DrawCentralMenu(table,VLF_USE_BUTTONS,VLF.Font,1)
```

table: the table we set up before

selected: as we want the function to control the item selection we'll choose VLF_USE_BUTTONS (now all the button inputs will be observed by that function)

font: we once again use the standard font VLF.Font

align: there are 3 different align settings
- 1:middle
- 2:left
- 3:right

we'll use number 1, thus middle

to check for the selected item VLF.DrawCentralMenu() returns a value,
so we have to store that value into a variable (I am using selected_item)

```
selected_item = VLF.DrawCentralMenu(menu,VLF_USE_BUTTONS,VLF.Font,1)
```

this we'll add after VLF.DrawFrame to the mainloop

in order to show text by choosing an item we need the following codeblock

code:
```
if selected_items then -- an item has been chosen
   if selected_items == 1 then --item 1 has been chosen
   last_item = 1
   elseif selected_items == 2 then --item 2 has been chosen
   last_item = 2
   elseif selected_items == 3 then --item 3 has been chosen
   last_item = 3
   end
end

VLF.DrawText(VLF_CENTER,230,VLF.Font,"last chosen item: "..last_item)
```

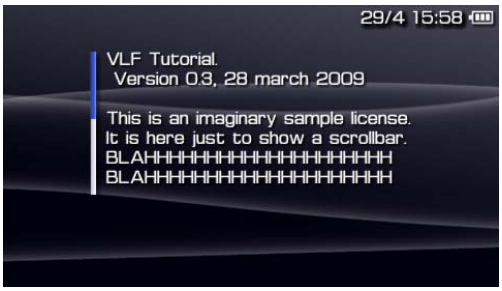
again we'll add this to the mainloop
and the application is done.

code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

table = {
"item 1",
"item 2",
"item 3"
}

last_item = ""

while pge.running() do
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()

selected_item = VLF.DrawCentralMenu(table,VLF_USE_BUTTONS,VLF.Font,1)

if selected_item then --an item has been chosen
   if selected_item == 1 then --item 1 has been chosen
   last_item = 1
   elseif selected_item == 2 then -- item 2 has been chosen
   last_item = 2
   elseif selected_item == 3 then -- item 3 has been chosen
   last_item = 3
   end
end

VLF.DrawText(VLF_CENTER,200,VLF.Font,"last chosen item: "..last_item)

pge.gfx.enddrawing()
pge.gfx.swapbuffers()
end
```

Part 6 Using the scrollbar

Screen:



exactly as with the central menu we have to define a table,
that contains the text that is meant to be shown

```
table = {
	"VLF Tutorial.",
	" Version 0.3, 28 march 2009",
	"",
	"This is an imaginary sample license.",
	"It is here just to show a scrollbar.",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH"
	}
```

now we'll put this before the mainloop

code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

table = {
	"VLF Tutorial.",
	" Version 0.3, 28 march 2009",
	"",
	"This is an imaginary sample license.",
	"It is here just to show a scrollbar.",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH",
	"BLAHHHHHHHHHHHHHHHHHHHH"
	}

while pge.running() do
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
end
```

lets move on to the main function

```
VLF.DrawScrollBar(text,act,x*,y*,height*)
```

as we don't need to set the x, y position and the height here
we abandon these parameters

text = the table we defined

act: there are 3 available values
  1: to scroll down
  2: to scroll up
  or VLF_USE_BUTTONS to let the function control it.

and the function looks like this:

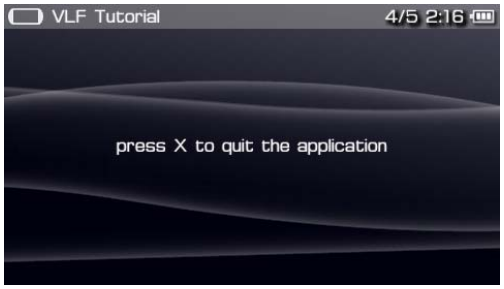
code:
```
VLF.DrawScrollBar(table,VLF_USE_BUTTONS)
```

to complete our application we only need to put this after VLF.DrawFrame() into the mainloop.

the complete code:
```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

table = {
      "VLF Tutorial.",
      " Version 0.3, 28 march 2009",
      "",
      "This is an imaginary sample license.",
      "It is here just to show a scrollbar.",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH",
      "BLAHHHHHHHHHHHHHHHHHHHH"
      }

while pge.running() do
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()

VLF.DrawScrollBar(table,VLF_USE_BUTTONS)

pge.gfx.enddrawing()
pge.gfx.swapbuffers()
end
```

Part 7 Using the title bar

The title bar can be used to show the name or state of an application
(eg.in xmb usb mode or dialogs)
Screen:



VLF.DrawTitleBar(text,anim*,image*,Color*)

text: here we're gonna enter ther text we want to show, indeed it is "VLF Tutorial"

anim: if you enter a 1 here then the titel bar is visible/it will be shown
if you enter a 2 here, there will be a fadeout/it will be invisible
we choose 1.

image: do you additionally want to show an icon you have to enter it here.
the VLF lib already contains the standard psp icons
TOP.APP shows a psp icon
TOP.UPD shows an update symbol
TOP.EGG shows an egg (is acutally shown within the installation)
TOP.INF shows a ? (is shown within info menus)
we'll decide for the psp symbol

Color: If you want the titlebar to have a special color you have to enter it here.
We won't choose one, so there will be used a color that is adapted to the background.

the function now looks like this

VLF.DrawTitleBar("VLF Tutorial",1,TOP.APP)

As it still looks pretty boring we will add some more text which says, that the application can be quit by pressing X
You should already know how it works...

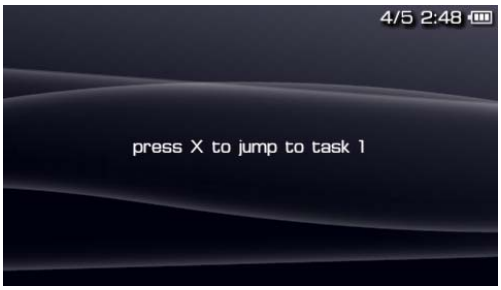
in order to execute it we need the following codeblock

code:
```
if pge.controls.pressed(PGE_CTRL_CROSS) then
pge.exit()
end
```

this codeblock we have to put into the mainloop
and our tiny application is done...

code:
```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

while pge.running() do
pge.controls.update()
if pge.controls.pressed(PGE_CTRL_CROSS) then
pge.exit()
end
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
VLF.DrawTitleBar("VLF Tutorial",1,TOP.APP)
VLF.DrawText(VLF_CENTER,130,VLF.Font,"press X to quit the application")
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
end
```

Part 8 Using tasks and VLF.FadeObjects()



Most of the programs consist of more than one "site", so you have to use tasks to show more than one site.

the basic principle is pretty easy:
a variable stands for the current task

code:
```
current_task = 1
```

the state of the variables is checked within the mainloop
and the respective task is called

code:
```
while pge.running() do

if current_task == 1 then
--task1
elseif  current_task == 2 then
--task2
end

end
```

in order to switch the task only the value of this variable has to change.
if you want to jump from value 1 to 5 by pressing x you only have to insert the following code

code:
```
if current_task == 1 then

if pge.controls.pressed(PGE_CTRL_CROSS) then
current_task = 5
end

end
```

But the transition does not look very professional and nice,
so we're gonna use the function VLF.FadeObjects()

Using this function without making any errors is pretty hard
To make it more easy I wrote the following function.

code:
```
function Fade_Control()
old_Next_task = old_Next_task or 0
Next_task = Next_task or 0
   if Fade_mode == 1 then
      if Fade_finished then
         current_task = Next_task
         Fade_mode = 2
         SET_HIGHLIGHT_BUTTON = 1
         VLF_DISABLE_FOCUS_ANIMATION = true
      end
   elseif Fade_mode == 2 then
      if Fade_finished then
         VLF_DISABLE_FOCUS_ANIMATION = false
      end
   end
   if old_Next_task ~= Next_task then
      Fade_mode = 1
   end
   old_Next_task = Next_task
end
```

Note: there has not been a great change concerning the usage.
to switch the task you only have to define Next_task

E.G.
code:
```
if current_task == 1 then

if pge.controls.pressed(PGE_CTRL_CROSS) then
Next_task = 5
end

end
```

However the function needs some preparation
first we have to put it before the mainloop
and call it within the mainloop.

code:
```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

function Fade_Control()
old_Next_task = old_Next_task or 0
Next_task = Next_task or 0
   if Fade_mode == 1 then
      if Fade_finished then
         current_task = Next_task
         Fade_mode = 2
         SET_HIGHLIGHT_BUTTON = 1
         VLF_DISABLE_FOCUS_ANIMATION = true
      end
   elseif Fade_mode == 2 then
      if Fade_finished then
         VLF_DISABLE_FOCUS_ANIMATION = false
      end
   end
   if old_Next_task ~= Next_task then
      Fade_mode = 1
   end
   old_Next_task = Next_task
end


current_task = 1

while pge.running() do

Fade_Control()

pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
end
```

now let's care about VLF.FadeObjects()

VLF.FadeObjects(in_out,speed)

in_out: As we are using a function we've written several lines before, we have to enter Fade_mode her
if we didn't use it we would have to write 1 to fadeout and 2 to fade in

speed: here we've got 5 different modes
VLF_FADE_SPEED_SLOW
VLF_FADE_SPEED_STANDARD
VLF_FADE_SPEED_FAST
VLF_FADE_SPEED_VERY_FAST
VLF_FADE_SPEED_SUPER_FAST
I am using VLF_FADE_SPEED_FAST, but you can choose whatever you like

Now our function is ready...

```
VLF.FadeObjects(Fade_mode,VLF_FADE_SPEED_FAST)
```

to detect when the fade has ended the function returns a value (fade is done -> true, otherwise false)
we'll define it as Fade_finished (likewise caused through Fade_Control())

code:
```
Fade_finished = VLF.FadeObjects(Fade_mode,VLF_FADE_SPEED_FAST)
```

The function has now to be inserted at the end of the while loop

code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)


function Fade_Control()
old_Next_task = old_Next_task or 0
Next_task = Next_task or 0

   if Fade_mode == 1 then
      if Fade_finished then
         current_task = Next_task
         Fade_mode = 2
         SET_HIGHLIGHT_BUTTON = 1
         VLF_DISABLE_FOCUS_ANIMATION = true
      end
   elseif Fade_mode == 2 then
      if Fade_finished then
         VLF_DISABLE_FOCUS_ANIMATION = false
      end
   end
   if old_Next_task ~= Next_task then
      Fade_mode = 1
   end
   old_Next_task = Next_task
end


current_task = 1

while pge.running() do

Fade_Control()

pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
Fade_finished = VLF.FadeObjects(Fade_mode,VLF_FADE_SPEED_FAST)
end
```

so far the basic steps are completed

now we can take care of the tasks, we're gonna use 3 tasks that can be switched by pressing X
additionally, in order to show in wich task we're in we'll print some text.

the codeblock looks like this:

code:

```
if current_task == 1 then

if pge.controls.pressed(PGE_CTRL_CROSS) then
Next_task = 2
end
--task 1

VLF.DrawText(VLF_CENTER,130,VLF.Font,"press X to jump to task 2")

elseif current_task == 2 then

if pge.controls.pressed(PGE_CTRL_CROSS) then
Next_task = 3
end
--task 2

VLF.DrawText(VLF_CENTER,130,VLF.Font, "press X to jump to task 3")

elseif current_task == 3 then

if pge.controls.pressed(PGE_CTRL_CROSS) then
Next_task = 1
end
--task 3

VLF.DrawText(VLF_CENTER,130,VLF.Font, "press X to jump to task 1")

end
```

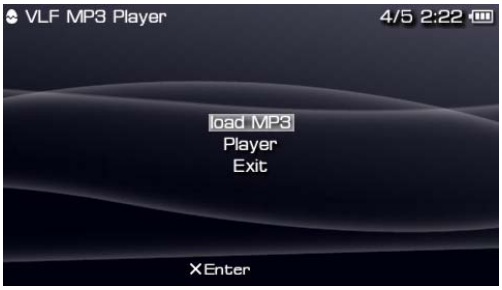
We'll write it after VLF.DrawFrame() into the mainloop and the application is complete.

code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

function Fade_Control()
old_Next_task = old_Next_task or 0
Next_task = Next_task or 0

   if Fade_mode == 1 then
      if Fade_finished then
         current_task = Next_task
         Fade_mode = 2
         SET_HIGHLIGHT_BUTTON = 1
         VLF_DISABLE_FOCUS_ANIMATION = true
      end
   elseif Fade_mode == 2 then
      if Fade_finished then
         VLF_DISABLE_FOCUS_ANIMATION = false
      end
   end
   if old_Next_task ~= Next_task then
      Fade_mode = 1
   end
   old_Next_task = Next_task
end


current_task = 1

while pge.running() do

Fade_Control()
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()

if current_task == 1 then

if pge.controls.pressed(PGE_CTRL_CROSS) then
Next_task = 2
end
--task 1

VLF.DrawText(VLF_CENTER,130,VLF.Font,"press X to jump to task 2")

elseif current_task == 2 then

if pge.controls.pressed(PGE_CTRL_CROSS) then
Next_task = 3
end
--task 2

VLF.DrawText(VLF_CENTER,130,VLF.Font, "press X to jump to task 3")

elseif current_task == 3 then

if pge.controls.pressed(PGE_CTRL_CROSS) then
Next_task = 1
end
--task 3

VLF.DrawText(VLF_CENTER,130,VLF.Font, "press X to jump to task 1")

end

pge.gfx.enddrawing()
pge.gfx.swapbuffers()
Fade_finished = VLF.FadeObjects(Fade_mode,VLF_FADE_SPEED_FAST)
end
```

Extra: VLF MP3 Player

Screen:



As I have explained all basic commands,
it is time to introduce you into a greater project.
I decided to create a simple mp3 player...

We're gonna use the following functions besides the mainloop

- VLF.DrawTitleBar()
- VLF.DrawCentralMenu()
- VLF.FadeObjects()
- VLF.DrawOptions()
- VLF.DrawText()

We will start with the extended Mainloop.(take a look at part 8)

code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

function Fade_Control()
old_Next_task = old_Next_task or 0
Next_task = Next_task or 0
   if Fade_mode == 1 then
      if Fade_finished then
         current_task = Next_task
         Fade_mode = 2
         SET_HIGHLIGHT_BUTTON = 1
         VLF_DISABLE_FOCUS_ANIMATION = true
      end
   elseif Fade_mode == 2 then
      if Fade_finished then
         VLF_DISABLE_FOCUS_ANIMATION = false
      end
   end
   if old_Next_task ~= Next_task then
      Fade_mode = 1
   end
   old_Next_task = Next_task
end


current_task = 1

while pge.running() do
Fade_Control()
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
Fade_finished = VLF.FadeObjects(Fade_mode,VLF_FADE_SPEED_FAST)
end
```

Now we'll add a title bar to our project.
You should already know how to add one...

in this example I choose the egg icon and the name is "VLF MP3 Player"
VLF.DrawTitleBar("VLF MP3 Player",1,TOP.EGG)

To make the controls within the menus easier to understand we're gonna use the function VLF.DrawOptions(),
which is meant to show the button allocation at the bottom of the screen, eg. X Enter / O exit and so on.
As we want to change the allocation allerdings depending on the task we have to use a variable instead of a static definiton (I am using the variable options)
Does it have the value 1, the function will show "X Enter", at value 2 there will be "O Back" and at value 3 there will be both.
In order to use the function we have to put it at the end of the mainloop, but before pge.gfx.enddrawing().

additionally we'll define some mp3 variables

your code should now look like this:

code:
```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

function Fade_Control()
old_Next_task = old_Next_task or 0
Next_task = Next_task or 0
   if Fade_mode == 1 then
      if Fade_finished then
         current_task = Next_task
         Fade_mode = 2
         SET_HIGHLIGHT_BUTTON = 1
         VLF_DISABLE_FOCUS_ANIMATION = true
      end
   elseif Fade_mode == 2 then
      if Fade_finished then
         VLF_DISABLE_FOCUS_ANIMATION = false
      end
   end
   if old_Next_task ~= Next_task then
      Fade_mode = 1
   end
   old_Next_task = Next_task
end


current_task = 1

current_mp3 = 1 --the current mp3
mp3 = {} --the table that contains all paths to the mp3 files

while pge.running() do
Fade_Control()
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
VLF.DrawTitleBar("VLF MP3 Player",1,TOP.EGG)

--here the menu code has to be inserted

VLF.DrawOptions(options)
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
Fade_finished = VLF.FadeObjects(Fade_mode,VLF_FADE_SPEED_FAST)
end
```

Next we will define the tables for each menu

first the Main menu
```
main = {"load MP3","Player","Exit"}
```

next is the Player menu
```
player = {"Pause/Play","Stop","forward","back"}
```

These ones we put again as definitions before the mainloop

well, next we'll create all tasks that we need

- Main menu | task 1
- Player Menu | task 2
- MP3 loading | task 3
- Exit | task 4

```
if current_task == 1 then --MAIN MENU
--to show "X Enter" with VLF.DrawOptions() we'll set options (take a look at line 915) = 1
options = 1

selected_item = VLF.DrawCentralMenu(main,VLF_USE_BUTTONS,VLF.Font,1) --shows the mainmenu and stores the selectem item in selected_item

--to check which option has been chosen in the menu we're gonna use this if block

if selected_item == 1 then --item 1 "choose MP3" has been chosen

--to switch to the MP3 menu we set Next_task = 3  (take a look at the task allocation at line 988)
Next_task = 3

elseif selected_item == 2 then --item 2 (player menu) has been chosen

--to switch to the player menu we set Next_task = 2
--but before we have to check if there are mp3 files available in order to play them

if #mp3 ~= 0 then --checks if mp3s have been loaded // if none have been loaded the MP3 task will be called (due to the mp3 files are loaded within the MP3 task)
Next_task = 2
else
Next_task = 3
end

elseif selected_item == 3 then --exit has been chosen

--here we also have to change Next_task
Next_task = 4

end


elseif current_task == 2 then --PLAYER MENU


--to show "X Enter" and "O Back" with VLF.DrawOptions() we have to set options (take a look at line 915) = 3
options = 3

--to return to the main menu by pressing O we're gonna add the following codeblock
if pge.controls.pressed(PGE_CTRL_CIRCLE) then --checks if O is pressed
Next_task = 1 --changes the current task to 1 (main menu)
pge.wav.play(SND_CURSOR) --the button sound will be played
end

selected_item = VLF.DrawCentralMenu(player,VLF_USE_BUTTONS,VLF.Font,1) --shows the player menu and stores the selected item in selected_item

if pge.mp3.isplaying() then --checks if a mp3 file is being played
VLF.DrawText(VLF_CENTER,50,VLF.Font,pge.mp3.title(pge.mp3.getinfo())..”  "..current_mp3.."/"..#mp3)--shows the title of the current track and the mp3-position within the table
end
```

```
if selected_item == 1 then --item 1 ("Pause / Play") has been chosen
--to execute the related function we're gonna use the  pge mp3 pause function
pge.mp3.pause() --if the function is called one time the mp3 will be paused, if the function is called 2 times the mp3 playback will continue

elseif selected_item == 2 then --item 2 ("stop") has been chosen

pge.mp3.stop()

elseif selected_item == 3 then --"continue" has been chosen
--to call the next MP3 function we're gonna use the following codeblock

if mp3[current_mp3+1] ~= nil then --checks for antoher mp3 file being available

current_mp3 = current_mp3 + 1 --increases the value of the current_mp3 variable by 1
pge.mp3.play(mp3[current_mp3]) --plays the next mp3 out of the mp3 table

else --if no more files are available within the table the first track will be played

current_mp3 = 1 --the value of the current_mp3 variable is set to 1
pge.mp3.play(mp3[current_mp3]) --plays the first mp3 out of the mp3 table

end

elseif selected_item == 4 then --"back" has been chosen

if current_mp3 ~= 1 then --checks for current_mp3 not being 1 (checks for the last mp3 being available)

current_mp3 = current_mp3 - 1 --decreases the value of the current_mp3 variable by 1
pge.mp3.play(mp3[current_mp3]) --plays the last played track out of the mp3 table

else --if current_mp3 = 1 (no more mp3s available) the last track that is contained in the table will be played

current_mp3 = #mp3 --the current_mp3 variable is set to the last field of the mp3 table (the last mp3 is chosen)
pge.mp3.play(mp3[current_mp3]) --plays the last mp3 of the mp3 table

end

end

elseif current_task == 3 then --MP3 LOADING

VLF.DrawText(VLF_CENTER,130,VLF.Font,"Searching MP3 files... ",2,3,VLF.Text_Color,VLF.Shadow_Color) --shows the state message
pge.gfx.swapbuffers()--in order to show the text without delay the buffers have to be swapped
--to insert all mp3s located in "ms0:/MUSIC/" into the mp3 table we're gonna use the following codeblock
 mp3 = {} --the mp3 table is emptied
 dir = pge.dir.open("ms0:/MUSIC/") --opens the directory "ms0:/MUSIC/"
 music_files = dir:read() --reads the directory and returns it to music_files
 dir:close() --closes the directory
 for i = 1, #music_files do --starts the for loop that checks all fields of the table music_files
    if not music_files[i].dir then --checks if the current field variable is a folder
       if string.sub(music_files[i].name,-3) == "mp3" then --checks if the current field variable is a mp3
       mp3[#mp3+1] = "ms0:/MUSIC/"..music_files[i].name --adds its path to the mp3 table
       end
    end
 end

if #mp3 ~= 0 then --checks if mp3 files have been found
pge.mp3.play(mp3[1]) --plays the first mp3 of the table
current_mp3 = 1
Next_task = 2 --switches to the player menu
else --there are no mp3 files available
Next_task = 4 --since there are no mp3 files available the application quits (the exit task is called)
end

elseif current_task == 4 then --EXIT
pge.gfx.swapbuffers() --in order to finish the fade properly the buffers are swapped a bit earlier
pge.exit() --the application quits

end
```

if you got that all you can copy the whole code into the mainloop and run the application.
but make sure that there are some MP3 files at ms0:/MUSIC/, otherwise the application gets pretty boring.

The complete code

code:

```
require("VLF")
VLF.init()
VLF.GetBG(28,"theme.vlf",true)

function Fade_Control()
old_Next_task = old_Next_task or 0
Next_task = Next_task or 0
   if Fade_mode == 1 then
      if Fade_finished then
         current_task = Next_task
         Fade_mode = 2
         SET_HIGHLIGHT_BUTTON = 1
         VLF_DISABLE_FOCUS_ANIMATION = true
      end
   elseif Fade_mode == 2 then
      if Fade_finished then
         VLF_DISABLE_FOCUS_ANIMATION = false
      end
   end
   if old_Next_task ~= Next_task then
      Fade_mode = 1
   end
   old_Next_task = Next_task
end


current_task = 1

current_mp3 = 1 --die aktuelle mp3
mp3 = {} --der table in dem alle mp3 pfade abgespeichert werden

--menu tables
main = {"load MP3","Player","Exit"}
player = {"Pause/Play","Stop","forward","back"}
while pge.running() do
Fade_Control()
pge.controls.update()
pge.gfx.startdrawing()
pge.gfx.clearscreen()
VLF.DrawFrame()
VLF.DrawTitleBar("VLF MP3 Player",1,TOP.EGG,pge.gfx.createcolor(0,0,0,0))

if current_task == 1 then --MAIN MENU
--to show "X Enter" with VLF.DrawOptions() we'll set options (take a look at line 915) = 1
options = 1

selected_item = VLF.DrawCentralMenu(main,VLF_USE_BUTTONS,VLF.Font,1) --shows the mainmenu and stores the selectem item in
selected_item

--to check which option has been chosen in the menu we're gonna use this if block

if selected_item == 1 then --item 1 "choose MP3" has been chosen

--to switch to the MP3 menu we set Next_task = 3  (take a look at the task allocation at line 988)
Next_task = 3

elseif selected_item == 2 then --item 2 (player menu) has been chosen

--to switch to the player menu we set Next_task = 2
--but before we have to check if there are mp3 files available in order to play them

if #mp3 ~= 0 then --checks if mp3s have been loaded // if none have been loaded the MP3 task will be called (due to the mp3 files are
loaded within the MP3 task)
Next_task = 2
else
Next_task = 3
end
```

```
elseif selected_item == 3 then --exit has been chosen

--here we also have to change Next_task
Next_task = 4

end


elseif current_task == 2 then --PLAYER MENU


--to show "X Enter" and "O Back" with VLF.DrawOptions() we have to set options (take a look at line 915) = 3
options = 3

--to return to the main menu by pressing O we're gonna add the following codeblock
if pge.controls.pressed(PGE_CTRL_CIRCLE) then --checks if O is pressed
Next_task = 1 --changes the current task to 1 (main menu)
pge.wav.play(SND_CURSOR) --the button sound will be played
end

selected_item = VLF.DrawCentralMenu(player,VLF_USE_BUTTONS,VLF.Font,1) --shows the player menu and stores the selected item in
selected_item

if pge.mp3.isplaying() then --checks if a mp3 file is being played
VLF.DrawText(VLF_CENTER,50,VLF.Font,pge.mp3.title(pge.mp3.getinfo())..."  "..current_mp3.."/"..#mp3)--shows the title of the current
track and the mp3-position within the table
end

if selected_item == 1 then --item 1 ("Pause / Play") has been chosen
--to execute the related function we're gonna use the  pge mp3 pause function
pge.mp3.pause() --if the function is called one time the mp3 will be paused, if the function is called 2 times the mp3 playback will continue

elseif selected_item == 2 then --item 2 ("stop") has been chosen

pge.mp3.stop()

elseif selected_item == 3 then --"continue" has been chosen
--to call the next MP3 function we're gonna use the following codeblock

if mp3[current_mp3+1] ~= nil then --checks for antoher mp3 file being available

current_mp3 = current_mp3 + 1 --increases the value of the current_mp3 variable by 1
pge.mp3.play(mp3[current_mp3]) --plays the next mp3 out of the mp3 table

else --if no more files are available within the table the first track will be played

current_mp3 = 1 --the value of the current_mp3 variable is set to 1
pge.mp3.play(mp3[current_mp3]) --plays the first mp3 out of the mp3 table

end

elseif selected_item == 4 then --"back" has been chosen

if current_mp3 ~= 1 then --checks for current_mp3 not being 1 (checks for the last mp3 being available)

current_mp3 = current_mp3 - 1 --decreases the value of the current_mp3 variable by 1
pge.mp3.play(mp3[current_mp3]) --plays the last played track out of the mp3 table

else --if current_mp3 = 1 (no more mp3s available) the last track that is contained in the table will be played

current_mp3 = #mp3 --the current_mp3 variable is set to the last field of the mp3 table (the last mp3 is chosen)
pge.mp3.play(mp3[current_mp3]) --plays the last mp3 of the mp3 table

end

end

elseif current_task == 3 then --MP3 LOADING

VLF.DrawText(VLF_CENTER,130,VLF.Font,"Searching MP3 files... ",2,3,VLF.Text_Color,VLF.Shadow_Color) --shows the state message
pge.gfx.swapbuffers()--in order to show the text without delay the buffers have to be swapped
--to insert all mp3s located in "ms0:/MUSIC/" into the mp3 table we're gonna use the following codeblock
```

```
 mp3 = {} --the mp3 table is emptied
 dir = pge.dir.open("ms0:/MUSIC/") --opens the directory "ms0:/MUSIC/"
 music_files = dir:read() --reads the directory and returns it to music_files
 dir:close() --closes the directory
 for i = 1, #music_files do --starts the for loop that checks all fields of the table music_files
    if not music_files[i].dir then --checks if the current field variable is a folder
       if string.sub(music_files[i].name,-3) == "mp3" then --checks if the current field variable is a mp3
       mp3[#mp3+1] = "ms0:/MUSIC/"..music_files[i].name --adds its path to the mp3 table
       end
    end
 end

if #mp3 ~= 0 then --checks if mp3 files have been found
pge.mp3.play(mp3[1]) --plays the first mp3 of the table
current_mp3 = 1
Next_task = 2 --switches to the player menu
else --there are no mp3 files available
Next_task = 4 --since there are no mp3 files available the application quits (the exit task is called)
end

elseif current_task == 4 then --EXIT
pge.gfx.swapbuffers() --in order to finish the fade properly the buffers are swapped a bit earlier
pge.exit() --the application quits

end

VLF.DrawOptions(options)
pge.gfx.enddrawing()
pge.gfx.swapbuffers()
Fade_finished = VLF.FadeObjects(Fade_mode,VLF_FADE_SPEED_FAST)
end
```

That's it so far for the VLF tutorial
if there you have got any problems concerning the tutorial or the library or you just want to give some feedback, then feel free to contact me.

ICQ: 596120749
email: lxd6000@gmail.com

Regards
LXD

Special THX to Dark Syntaxx, Bumuckl, stefan1381, Anadox und Betta

Translated by Bumuckl (bumuckl.com)