TNT ENGiNE “Virtual Pad”

v1.40 for Gideros SDK

coded By Gianluca D'Angelo

Special thanks to SinisterSoft for Help/Support/BugFixes

www.tntengine.com
(dedicated with love to my Wife “Cinzia” and my son “Mattia”)

TNT Virtual PAD it's free and you can use it in free and commercial games but (if you use it) PLEASE support this project making a donation to the author.

If it's free why i ask to help me with a donation?

I spent many hours to make TNT Virtual PAD the best that i can… with your donation i can continue to keep this project alive adding new features, bugfix and make it even better! Plus if you make a donation i will send you the **FULL SOURCE CODE of TNT Virtual PAD**; so if I saved you some hours of work why not donate? ;)

Thanks for your help and support.
Gianluca D'Angelo.
(gregbug@gmail.com)

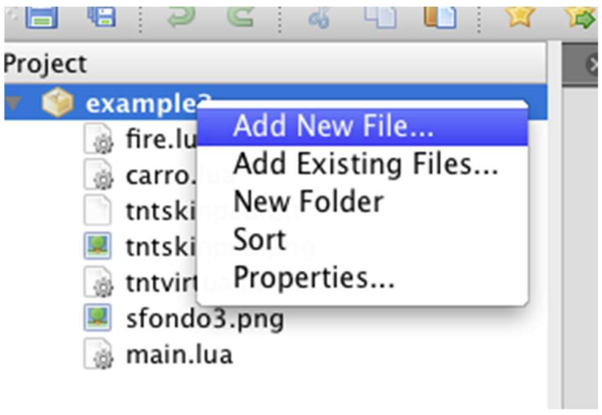
LET'S *START!*

Ok, let's start!
So what is TNT Virtual PAD?
Simple... it's a "virtual" joystick for your multi-touch screen mobile device (iOS or Android) it can manage 2 virtual joystick (analogical or digital) and up of 4 buttons!

Every component (like buttons, joysticks) can be TOTALLY customizable like textures, colours, size, position and behaviours.

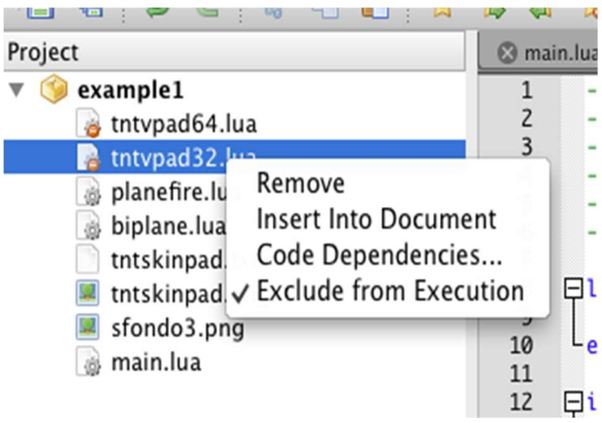
Using TNT Virtual PAD is very simple and his event management it's very easy and fast to manage!

First of all you need to add TNT Virtual Pad (VPAD from now) to your existing project:



if you have full sources of VPAD just right click on the project name and select "Add New File..." then select tntvirtualpad.lua

Now you are ready to use VPAD!



If you don't have sources add the two files "tntvpad32.lua" and "tntvpad64.lua" to your project files (use the same method used above) and then right click on the two files and select "Exclude from Execution" then in your main.lua file you need to add this simple code (thanks to Atilim) :

```
local function is32bit()
        return
```

```
string.dump(is32bit):byte(9) == 4
                                        end

                                        if is32bit() then
                                                        require("tntvpad32")
                                                else
                                                        require("tntvpad64")
                                                end
```

then you need to add to your project the texture file that contain graphics for Vpad.

for more info on setup VPAD try the examples.

VPAD have one main public Class defined and it's called **CTNTVirtualPAD** and with this class you can manage all your needs.

After you have added VPAD library to you project you need to init VPAD Class and you can do with this simple code:

```
local vPad = CTNTVirtualPad.new(stage, "tntskinpad",  PAD.STICK_SINGLE, PAD.BUTTONS_TWO, 20, 1)
```

parameters for CTNTVirtualPad constructor are:

```
CTNTVirtualPad:init(parent, textureFileName, padSticks, padButtons, borderSpace, layerIndex, vBorderSpace, restrict)
```

parent:
define parent sprite (like stage)

textureFileName:
filename of texturepack that contains all virtual pad needed graphics it MUST contain at least tntvpad_analogpad.png, tntvpad_base.png, tntvpad_buttondown.png, tntvpad_buttonup.png you can customize default texture file name using function “setDefaultJoySprites(spriteA, spriteB)” for joysticks and setDefaultButtonSprites(spriteA, spriteB) for buttons.

padSticks:
define the numbers of virtual pads (joysticks) and can be:
PAD.STICK_NONE no joystick used.
PAD.STICK_SINGLE use only one joystick (left by default)
PAD.STICK_DOUBLE use two joystick (left and right by default)

padButtons:
define the numbers of buttons used and can be:
PAD.BUTTONS_NONE no buttons used
PAD.BUTTONS_ONE one button used
PAD.BUTTONS_TWO two buttons used
PAD.BUTTONS_THREE three buttons used
PAD.BUTTONS_FOUR four buttons used
PAD.BUTTONS_FIVE five buttons used
PAD.BUTTONS_SIX six buttons used

borderSpace:
define minimum space (in pixel) from screen borders and VPAD components.

layerIndex:
define the layer of the virtual pad (if not defined use layer 0). With this value you can manage the position (layer) of virtual pad.

vBorderSpace:
optional vertical border space also activate flipped mode if negative else same as horizontal border space.

restrict:
screen restrict for setting screen height from base, default to height

to start VPAD simply use:

```
vPad:start()
```

for more detailed information please try examples.

After you have used VPAD remember to release memory used by VPAD using

example:

```
vPad = vPad:free()
```

TNT Virtual Pad Class reference.

function CTNTVirtualPad:init(parent, textureFileName, padSticks, padButtons, borderSpace, layerIndex, vBorderSpace)

Virtual Pad class constructor

usage Example:

```
local vPad = CTNTVirtualPad.new(stage, "tntskinpad",  PAD.STICK_SINGLE, PAD.BUTTONS_TWO, 20, 1)
```

parameters for CTNTVirtualPad constructor are:

parent:
define parent sprite (like stage)

textureFileName:
filename of texturepack that contains all virtual pad needed graphics it MUST contain at least tntvpad_analogpad.png, tntvpad_base.png, tntvpad_buttondown.png, tntvpad_buttonup.png you can customize default texture file name using function “setDefaultJoySprites(spriteA, spriteB)” for joysticks and setDefaultButtonSprites(spriteA, spriteB) for buttons.

padSticks:
define the numbers of virtual pads (joysticks) and can be:
PAD.STICK_NONE no joystick used.
PAD.STICK_SINGLE use only one joystick (left by default)
PAD.STICK_DOUBLE use two joystick (left and right by default)

padButtons:
define the numbers of buttons used and can be:
PAD.BUTTONS_NONE no buttons used
PAD.BUTTONS_ONE one button used
PAD.BUTTONS_TWO two buttons used
PAD.BUTTONS_THREE three buttons used
PAD.BUTTONS_FOUR four buttons used
PAD.BUTTONS_FIVE five buttons used
PAD.BUTTONS_SIX six buttons used

borderSpace:
define minimum space (in pixel) from screen borders and VPAD components.

layerIndex:
define the layer of the virtual pad (if not defined use layer 0). With this value you can manage the position (layer) of virtual pad.

vborderSpace:
optional vertical border space;
also activated flipped mode if negative, else same as horizontal border space

restrict:
screen restrict for setting screen height from base, default to height

function CTNTVirtualPad:free()

Virtual Pad class destructor. Free all memory allocated by VPAD and release nil.

usage Example:

vPad = vPad:free()

function CTNTVirtualPad:setAlpha(alphaOn, alphaOff)

set alpha channel value (from 0 to 1) for all VPAD components;

alphaOn:
define max alpha value when pad is visible on screen
alphaOff:
define min alpha value when pad is in mode “ghost”

usage Example:
vPad:setAlpha(1, 0.5)

function CTNTVirtualPad:setColor(padComponent, r, g, b)

set colour of virtual pad component.

padComponent:
PAD.COMPO_BUTTON1 for button 1
PAD.COMPO_BUTTON2 for button 2
PAD.COMPO_BUTTON3 for button 3
PAD.COMPO_BUTTON4 for button 4
PAD.COMPO_BUTTON5 for button 5
PAD.COMPO_BUTTON6 for button 6
PAD.COMPO_LEFTPAD for left joystick
PAD.COMPO_RIGHTPAD for right joystick

r:
red value (0 to 255)
g:
green value (0 to 255)
b:
blue value (0 to 255)

usage Example:
vPad = vPad:setColor(PAD.COMPO_BUTTON1, 255, 100, 100)

function CTNTVirtualPad:setHideDelay(msDelay)

set delay before hide virtual pad (defined in millisecs)

msDelay: milliseconds before hide the pad.

usage Example:
vPad = vPad:setHideDelay(1500)

function CTNTVirtualPad:setHideMode(hideMode)

set hide modality of virtual pad.

hideMode:
PAD.MODE_NOHIDE set VPAD always visible on screen. (setHideDelay is ignored)
PAD.MODE_GHOST set VPAD in ghost mode VPAD is first rendered with alphaOn value (defined with setAlpha) then (after delay setHideDelay) is rendered with alphaOff value.
PAD.MODE_HIDDEN set VPAD first with alphaOn value then after hideDelay Vpad is totally hidden.

usage Example:
vPad = vPad:setHideMode(PAD.MODE_GHOST)

*function CTNTVirtualPad:setICade(onOff) *** IN NEXT VERSION ****

activate the support for Hardware iCade (plugin coded by Magnusviri) set hide modality of virtual pad.

onOff:
if true iCade support is enabled and VPAD is disabled.
if false (default) iCade is disabled and Virtual pad is enabled

usage Example:
vPad = vPad:setICade(true)

*function CTNTVirtualPad:getICade() *** IN NEXT VERSION ****

release true if iCade mode is active else release false

usage Example:
iCadePad = vPad:getICade()

function CTNTVirtualPad:setJoyAsAnalog(padComponent, isAnalogic)

set joystick analog mode on/off.

padComponent:
PAD.COMPO_LEFTPAD for left pad
PAD.COMPO_RIGHTPAD for right pad
isAnalogic:
if true (default) joystick is in analog mode so it return exactly the angle (in radiants) of joystick and it's power; if false the angle returned is a value form 0 to 7 (eight direction mode) 0 is right and so on in clockwise order steps are every 45 degree.

usage Example:
```
vPad = vPad:setJoyAsAnalog(PAD.COMPO_LEFTPAD, false)
```

function CTNTVirtualPad:setJoyStyle(padComponent, style)

set joystick movement style.

padComponent:
PAD.COMPO_LEFTPAD for left pad
PAD.COMPO_RIGHTPAD for right pad
style:
PAD.STYLE_CLASSIC (default) Joystick are fixed
PAD.STYLE_MOVABLE joystick move where user last touch the screen then is fixed to this position
PAD.STYLE_FOLLOW joystick follow user finger movement

usage Example:
```
vPad = vPad:setJoyStyle(PAD.COMPO_LEFTPAD, PAD.STYLE_FOLLOW)
```

function CTNTVirtualPad:setPosition(padComponent, x, y)

set VPAD components x, y position

padComponent:
PAD.COMPO_LEFTPAD for left pad
PAD.COMPO_RIGHTPAD for right pad
PAD.COMPO_BUTTON1 for button 1
PAD.COMPO_BUTTON2 for button 2
PAD.COMPO_BUTTON3 for button 3
PAD.COMPO_BUTTON4 for button 4
PAD.COMPO_BUTTON5 for button 5
PAD.COMPO_BUTTON6 for button 6

x,y:
x, y position of pad componen

usage Example:
```
vPad = vPad:setPosition(PAD.COMPO_LEFTPAD, 100,100)
```

function CTNTVirtualPad:setScale(padComponent, scale)

set VPAD components scale factor (1 is original size, 2 is double size, 0.5 is half size and so on...)

padComponent:
PAD.COMPO_LEFTPAD for left pad
PAD.COMPO_RIGHTPAD for right pad
PAD.COMPO_BUTTON1 for button 1
PAD.COMPO_BUTTON2 for button 2
PAD.COMPO_BUTTON3 for button 3
PAD.COMPO_BUTTON4 for button 4
PAD.COMPO_BUTTON5 for button 5
PAD.COMPO_BUTTON4 for button 6

scale:
scale factor (1 is original size, 2 is double size, 0.5 is half size and so on...)

usage Example:
```
vPad = vPad:setScale(PAD.COMPO_LEFTPAD, 0.6)
```

function CTNTVirtualPad:setTextures(padComponent, textureA, textureB)

set VPAD component sprites (and replace default sprites graphics for selected component)

padComponent:

PAD.COMPO_LEFTPAD for left pad
PAD.COMPO_RIGHTPAD for right pad
PAD.COMPO_BUTTON1 for button 1
PAD.COMPO_BUTTON2 for button 2
PAD.COMPO_BUTTON3 for button 3
PAD.COMPO_BUTTON4 for button 4
PAD.COMPO_BUTTON5 for button 5
PAD.COMPO_BUTTON4 for button 6

textureA:

sprite used for render “unpressed” button or the joystick's “stick”

textureB:

sprite used for render “pressed” button or the joystick “basement”

usage Example:

```
vPad = vPad:setScale(PAD.COMPO_LEFTPAD, 0.6)
```

function CTNTVirtualPad:setMaxRadius(padComponent, maxRadius)

set touch radius size detections for left and right virtual pads.
If maxRadius<1 then touch detection for leftPad is left half Screen and for rightPad is right half screen.

padComponent:

PAD.COMPO_LEFTPAD for left pad
PAD.COMPO_RIGHTPAD for right pad

maxRadius:

Touch radius size detection. (if < 1 half screen is used)

usage Example:

```
vPad:setMaxRadius(PAD.COMPO_LEFTPAD, 110)
```

function CTNTVirtualPad:start()

start Virtual Pad and it's events.

function CTNTVirtualPad:stop()

stop Virtual Pad and it's events.

function setDefaultButtonSprites(spriteA, spriteB)

set default sprites used for render buttons. SpriteA is “unpressed” button, SpriteB is “pressed” button sprite.

spriteA:

sprite used for render “unpressed” button

spriteB:

sprite used for render “pressed” button

function setDefaultJoySprites(spriteA, spriteB)

set default sprites used for render joysticks. SpriteA is joystick's “stick”, SpriteB is joystick “base” sprite.

spriteA:

sprite used for render joystick's “stick”

spriteB:

sprite used for render joystick “base”

TNT Virtual Pad Class summary.

function CTNTVirtualPad:init(parent, textureFileName, padSticks, padButtons, borderSpace, layerIndex)

function CTNTVirtualPad:free()

function CTNTVirtualPad:setAlpha(alphaOn, alphaOff)
function CTNTVirtualPad:setColor(padComponent, r, g, b)

function CTNTVirtualPad:setHideDelay(msDelay)
function CTNTVirtualPad:setHideMode(hideMode)

function CTNTVirtualPad:setICade(onOff)
function CTNTVirtualPad:getICade()

function CTNTVirtualPad:setJoyAsAnalog(padComponent, isAnalogic)
function CTNTVirtualPad:setJoyStyle(padComponent, style)

function CTNTVirtualPad:setPosition(padComponent, x, y)
function CTNTVirtualPad:setScale(padComponent, scale)

function CTNTVirtualPad:setTextures(padComponent, textureA, textureB)
function CTNTVirtualPad:setMaxRadius(padComponent, maxRadius)

function CTNTVirtualPad:start()
function CTNTVirtualPad:stop()

function setDefaultButtonSprites(spriteA, spriteB)
function setDefaultJoySprites(spriteA, spriteB)

TNT Virtual Pad Class Event System.

TNT Virtual Pad “communicates” with your code through very efficient Gideros event system so events are the central mechanism to handle user action code replies and they allow to create interactive applications (like Gideros built-in events).

This is a very basic skeleton on how manage VPAD events (left joystick movement and fire button) they are just simple standard Gideros events.

```
local function leftJoy(Event)
--handle left joy code here...
--this function is activated ONLY when user interact with left joystick
end
local function fire(Event)
--handle fire button code (button1) here ...
--this function is activated ONLY when user press button 1 (fire)
end

--add event VPAD event listeners to your game...
vPad:addEventListener(PAD.LEFTPAD_EVENT, leftJoy, self)
vPad:addEventListener(PAD.BUTTON1_EVENT, fire, self)
```

TNT Virtual Pad Events.

for joysticks elements events are

PAD.LEFTPAD_EVENT
dispatched when user interact with left pad.
PAD.RIGHTPAD_EVENT
dispatched when user interact right pad.

Joysticks event has data associated that contains in “data” field useful information that permit to you to manage user movements:

data.selected
can be true or false (true when element is interacting with user)
data.power
value (float) can be from 0 to 1 (1 when stick is on the base border)
data.angle
angle of joystick in radiants.
If joystick is setted as digital values are from 0 to 7 (0 is right and so on in clockwise order step are 45 degree)
data.state
is the state of joystick
PAD.STATE_NONE -- nothing happen
PAD.STATE_BEGIN -- joystick is touched
PAD.STATE_DOWN -- joystick is touched and pressed
PAD.STATE_END -- joystick is released

for buttons elements events are

PAD.BUTTON1_EVENT
dispatched when user interact with button1.
PAD.BUTTON2_EVENT
dispatched when user interact with button2.
PAD.BUTTON3_EVENT
dispatched when user interact with button3.
PAD.BUTTON4_EVENT

dispatched when user interact with button4.

PAD.BUTTON5_EVENT

dispatched when user interact with button4.

PAD.BUTTON6_EVENT

dispatched when user interact with button4.

Also buttons events has data associated that contains in “data” field useful information that permit to you to manage user actions:

data.selected

can be true of false (true when element is interacting with user)

data.state

is the state of joystick
PAD.STATE_NONE -- nothing happen
PAD.STATE_BEGIN -- button is touched
PAD.STATE_DOWN -- button is touched and pressed
PAD.STATE_END -- button is released

as always it's recommended to try VPAD examples for better explanation.
Special thanks to all Gideros forum, Gideros Team (Atilim and Gorkem) for making Gideros every day the best Mobile SDK out there!.

Please visit: www.giderosmobile.com

Special Thanks to
SinisterSoft for Support/Help/BugFix.

Thanks to all my beta tester (in alphabetical order):

Ar2rsawseen (please visit his very useful site for tutorials www.appcodingeasy.com or www.builtwithgideros.com)
Atilim,
Gorkem,
Magnusviri,
Scouser.

HISTORY:

13/05/2012
v1.00 First Version out!
20/07/2012
v1.10
function CTNTPadBase:setTextures(spriteA, spriteB)
- FIXED - >> BUG report by Gleen Bacon (thanks)
- Some Internal Optimizations.

18/11/2012
v1.20
Fixed vPad Event System (PAD.STATE_END not raised) reported by Tom2012 thx!
Examples Updates (compatibility with Gideros 2012.9.x)
SetMaxRadius() function added. (see example 2 and uncommend line 28 to test)
thx to Tom2012 for suggestion.

10/01/2013
v.1.30 ***THANKS TO SinisterSoft.***
FIXED Clippin/Scale Bug on Vpad. (added by/and THANKS TO SinisterSoft).
Added Button5 and Button6 Feature. (added by/and THANKS TO SinisterSoft).

11/01/2013

v.1.31 ***THANKS TO SinisterSoft.***

Added optional parameter in Vpad Init method for vertical space and flip mode.

22/01/2013

v1.36 Several Bug Fix for new Buttons system (by SinisterSoft)

Fix a possible Crash if layerInde† is nil or < 1

Added screen Restrict (By Sinistersoft)

New Button “default”position (By Sinistersoft)

14/03/2013

v.1.40 setMaxRadius half screen Bug Fixed.

Example 3 (2 Players) added

some code optimizations.

Ciao Ciao,
Gianluca D'Angelo

L'Aquila (...you will back to fly!) (ITALY)
14 Marzo 2013