Archived:Como usar eventos de toque em PySymbian

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

The article is believed to be still valid for the original topic scope.

Introdução

Esta aplicação exemplo em Archived:PySymbian demonstra como usar o módulo `appuifw` que tem extensão para Touch UI desde a versão 1.9.3 do PySymbian.

Este artigo contem exemplo de código demonstrando:

- Como definir áreas ativas na tela
- Como capturar eventos de toque, interpretar os dados recebidos e tomar ações do usuário baseados neles.

Há também um exemplo de código sobre como definir e manipular "teclas virtuais" para botões do menu de opções e sair.

Tutorial para iniciantes

Primeiramente, nós temos um pequeno, e espero, simples exemplo para iniciantes. Quando você tiver entendido como isso funciona, podemos passar para um exemplo mais complexo.

```
import appuifw, audio, e32, key_codes

def draw_state():
    pass
    canvas.clear()
    canvas.text((0,12),u"event",0x008000)
    #audio.say("cleared")

def left(arg):
    # Crash, if called before previous audio.say() has completed
    audio.say("left")

def right(arg):
    # Crash, if called before previous audio.say() has completed
    audio.say("right")

def quit(arg):
    audio.say("quit")
    e32.ao_sleep(1)
    lock.signal()

def callback(event):
    #print event
    #canvas.text((0,12),event,0x008000)
    pass

canvas=appuifw.Canvas(redraw_callback=lambda rect:draw_state())

appuifw.app.screen = 'full'
```


(C) Copyright Nokia 2014. All rights reserved.

```
# Note: directional_pad only with PySymbian 1.9.7 and later
appuifw.app.directional_pad = False
appuifw.app.body=canvas

canvas.bind(key_codes.EButton1Down, left, ((0,0),(360,200)))
canvas.bind(key_codes.EButton1Down, right, ((0,440),(360,640)))
canvas.bind(key_codes.EButton1Down, quit, ((160,300),(200,340)))

canvas.rectangle(((0,0),(360,200)), fill = (255,0,0))
canvas.rectangle(((0,440),(360,640)), fill = (0,255,0))
canvas.rectangle(((160,300),(200,340)), fill = (0,0,255))

lock = e32.Ao_lock()
appuifw.app.exit_key_handler=lock.signal
lock.wait()
```

Trecho de código

Um exemplo um pouco mais complexo com capturas de tela da aplicação a seguir:

```
'''
Watch Me - Light Touch
Fun color changing touch application for Nokia 5800
'''

VERSION = '1.20'

import sys
import e32
import appuifw
import graphics
import key_codes
import random

# Check if this can run at all
(a, b, c, d, e) = e32.pys60_version_info
if (a > 1) or (a == 1 and b >= 9 and c >= 3):
    if not appuifw.touch_enabled():
        appuifw.note(u"Touch is not enabled!")
        appuifw.app.set_exit()
else:
    appuifw.note(u"Touch is not enabled!")
    appuifw.app.set_exit()

# BUG: should exit, if cannot run
# TODO: How to do it gracefully
# appuifw.app.set_exit()

# RGB color model
# http://en.wikipedia.org/wiki/Rgb
RGB_MIN = 0
RGB_MAX = 255
RGB_RED = (255, 0, 0)
RGB_GREEN = (0, 255, 0)
RGB_BLUE = (0, 0, 255)
RGB_WHITE = (255, 255, 255)
```


(C) Copyright Nokia 2014. All rights reserved.

```
RGB_GRAY = (120, 120, 120)
RGB_BLACK = (0, 0, 0)
(rgb_red, rgb_green, rgb_blue) = RGB_BLUE
dir_red = dir_green = dir_blue = -1

# Global variables, UI controls
canvas = img = None
g_maxx = g_maxy = 0
lb = None

# Global variables, save last touch point coordinates
g_rx = g_ry = 0     # Red touch
g_gx = g_gy = 0     # Green touch
g_bx = g_by = 0     # Blue touch

# Control automatic on-screen color rotation
g_rotate_time = 0.01
g_rotate = False

# Control screensaver on/off status
my_timer = e32.Ao_timer()
rot_timer = e32.Ao_timer()
g_screensaver_on = True

def key_lesscolor(a_color, a_value):
    ''' Make RGB value of color smaller '''
    global rgb_red, rgb_green, rgb_blue
    a_value = max(a_value, RGB_MIN)
    if a_color == "red":
        rgb_red = a_value
    elif a_color == "green":
        rgb_green = a_value
    elif a_color == "blue":
        rgb_blue = a_value
    cb_handle_redraw()

def key_morecolor(a_color, a_value):
    ''' Make RGB value of color bigger '''
    global rgb_red, rgb_green, rgb_blue
    a_value = min(a_value, RGB_MAX)
    if a_color == "red":
        rgb_red = a_value
    elif a_color == "green":
        rgb_green = a_value
    elif a_color == "blue":
        rgb_blue = a_value
    cb_handle_redraw()

def set_fullcolor(a_color):
    ''' Set screen to given RGB color '''
    global rgb_red, rgb_green, rgb_blue
    (rgb_red, rgb_green, rgb_blue) = a_color
    cb_handle_redraw()

def validate_rgb(a_value):
    ''' Fit given value inside RGB limits '''
```


(C) Copyright Nokia 2014. All rights reserved.

```
    if a_value < RGB_MIN:
        a_value = RGB_MIN
    elif a_value > RGB_MAX:
        a_value = RGB_MAX
    return a_value

def validate_color(r, g, b):
    ''' Make sure given values can be used as RGB color '''
    r = validate_rgb(r)
    g = validate_rgb(g)
    b = validate_rgb(b)
    return (r, g, b)

def cb_red_down(pos=(0, 0)):
    ''' Event handler for Red area '''
    global g_rx, g_ry
    g_rx, g_ry = pos

def cb_red_up(pos=(0, 0)):
    ''' Event handler for Red area '''
    pass

def cb_red_drag(pos=(0, 0)):
    ''' Event handler for Red area '''
    global g_rx, g_ry
    if pos[0] < g_rx:
        key_lesscolor("red", rgb_red-(g_rx-pos[0]))
    else:
        key_morecolor("red", rgb_red+(pos[0]-g_rx))
    g_rx, g_ry = pos

def cb_green_down(pos=(0, 0)):
    ''' Event handler for Green area '''
    global g_gx, g_gy
    g_gx, g_gy = pos

def cb_green_up(pos=(0, 0)):
    ''' Event handler for Green area '''
    pass

def cb_green_drag(pos=(0, 0)):
    ''' Event handler for Green area '''
    global g_gx, g_gy
    if pos[0] < g_gx:
        key_lesscolor("green", rgb_green-(g_gx-pos[0]))
    else:
        key_morecolor("green", rgb_green+(pos[0]-g_gx))
    g_gx, g_gy = pos

def cb_blue_down(pos=(0, 0)):
    ''' Event handler for Blue area '''
    global g_bx, g_by
    g_bx, g_by = pos

def cb_blue_up(pos=(0, 0)):
    ''' Event handler for Blue area '''
    # Options softkey box
```


(C) Copyright Nokia 2014. All rights reserved.

```
    if pos[0] < 100 and pos[1] > g_maxy-100:
        cb_options_menu()
    # Exit softkey box
    elif pos[0] > g_maxx-100 and pos[1] > g_maxy-100:
        cb_quit()

def cb_blue_drag(pos=(0, 0)):
    ''' Event handler for Blue area '''
    global g_bx, g_by
    if pos[0] < g_bx:
        key_lesscolor("blue", rgb_blue-(g_bx-pos[0]))
    else:
        key_morecolor("blue", rgb_blue+(pos[0]-g_bx))
    g_bx, g_by = pos

def cb_options_menu(dummy=(0, 0)):
    ''' Look-a-like Options menu handler '''
    # Change Options menu item text dynamically
    if g_rotate:
        s = u"Stop Color Rotation"
    else:
        s = u"Start Color Rotation"

    # Show look-a-like Options menu
    i = appuifw.popup_menu(\
        [s, u"Set RGB Color", u"About", u"Exit"],
        u"Options")

    # Handle selection
    if i == 0:
        menu_rgb_rotate()
    elif i == 1:
        menu_rgb_query()
    elif i == 2:
        menu_about()
    elif i == 3:
        cb_quit()

def rgb_rotate():
    ''' Do on-screen color rotation '''
    rot_timer.cancel()

    global rgb_red, rgb_green, rgb_blue
    global dir_red, dir_green, dir_blue

    # Red
    value = random.choice([0, 1, 2])
    if dir_red < 0:
        value = -value
    rgb_red = validate_rgb(rgb_red + value)
    if rgb_red <= RGB_MIN or rgb_red >= RGB_MAX:
        dir_red = -dir_red
    # Green
    value = random.choice([0, 1, 2, 3])
    if dir_green < 0:
        value = -value
```


(C) Copyright Nokia 2014. All rights reserved.

```
    rgb_green = validate_rgb(rgb_green + value)
    if rgb_green <= RGB_MIN or rgb_green >= RGB_MAX:
        dir_green = -dir_green
    # Blue
    value = random.choice([0, 1, 2, 3, 4])
    if dir_blue < 0:
        value = -value
    rgb_blue = validate_rgb(rgb_blue + value)
    if rgb_blue <= RGB_MIN or rgb_blue >= RGB_MAX:
        dir_blue = -dir_blue
    # Update screen, make it visible
    draw_screen()
    rot_timer.after(g_rotate_time, rgb_rotate)

def cb_listbox():
    ''' Callback for RGB query listbox '''
    global lb
    global rgb_red, rgb_green, rgb_blue
    i = lb.current()

    # Red edit
    if i == 0:
        a = appuifw.query(u"New \'Red\' value (0-255):", "number", int(rgb_red))
        rgb_red = validate_rgb(a)
    # Green edit
    elif i == 1:
        a = appuifw.query(u"New \'Green\' value (0-255):", "number", int(rgb_green))
        rgb_green = validate_rgb(a)
    # Blue edit
    elif i == 2:
        a = appuifw.query(u"New \'Blue\' value (0-255):", "number", int(rgb_blue))
        rgb_blue = validate_rgb(a)

    # Refresh listbox with new value
    entries = [
        (u"Red", unicode(int(rgb_red))),
        (u"Green", unicode(int(rgb_green))),
        (u"Blue", unicode(int(rgb_blue))),
        ]
    lb = appuifw.Listbox(entries, cb_listbox)
    appuifw.app.body = lb

def menu_rgb_query():
    ''' Define listbox for RGB query '''
    # Force screen size 'normal', otherwise looks weird
    appuifw.app.screen = "normal"

    # Create a new Listbox with current color RGB values
    entries = [
        (u"Red", unicode(int(rgb_red))),
        (u"Green", unicode(int(rgb_green))),
        (u"Blue", unicode(int(rgb_blue))),
        ]
    global lb
    lb = appuifw.Listbox(entries, cb_listbox)
    appuifw.app.exit_key_handler = cb_rgb_close
    appuifw.app.body = lb
```


(C) Copyright Nokia 2014. All rights reserved.

```
    # Use Listbox specific Options menu
    appuifw.app.menu = [
        (u"Select", cb_listbox),
        (u"Close", cb_rgb_close)]

def cb_rgb_close():
    ''' Callback for RGB query listbox Exit '''
    # Restore initial application setup
    appuifw.app.screen = "full"
    appuifw.app.body = canvas
    appuifw.app.exit_key_handler = cb_quit
    # Make it visible
    draw_screen()

def menu_rgb_rotate():
    ''' Toggle on-screen color rotation status '''
    cb_handle_redraw()

    global g_rotate
    if g_rotate:
        g_rotate = False
        rot_timer.cancel()
    else:
        g_rotate = True
        rgb_rotate()

def menu_about():
    ''' Callback for menu item About '''
    appuifw.note(u'Watch Me - Light Touch v' + VERSION + u'\n'+\
        u'jouni.miettunen.googlepages.com\n\u00a9 2009 Jouni Miettunen')

def cb_handle_redraw(dummy=(0, 0, 0, 0)):
    ''' Overwrite default screen redraw event handler '''
    global img
    if img == None:
        img = graphics.Image.new(canvas.size)
    draw_screen()

def draw_screen():
    ''' Prepare off-screen and show it '''
    if img:
        img.clear((rgb_red,rgb_green,rgb_blue))
        canvas.blit(img)

def handle_screensaver():
    ''' Callback to handle screensaver activation '''
    global g_screensaver_on
    if g_screensaver_on:
        # Reset inactivity timer to keep lights on
        e32.reset_inactivity()
        my_timer.cancel()
        # N82 Settings UI has minimum value 5 seconds
        # Guess: set timeout as 4 seconds
        my_timer.after(4, handle_screensaver)
    else:
```

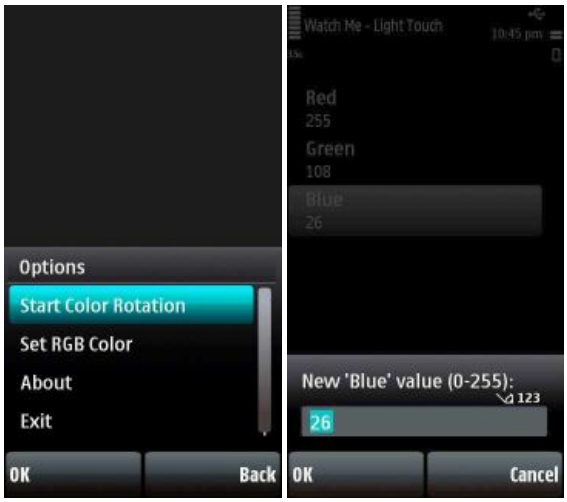

(C) Copyright Nokia 2014. All rights reserved.

```
        my_timer.cancel()

def cb_focus(fg):
    ''' System callback to tell when focus is lost/regained '''
    global g_screensaver_on
    if fg:
        # Got focus
        g_screensaver_on = True
    else:
        # Lost focus
        g_screensaver_on = False
    handle_screensaver()

def cb_quit():
    ''' Prepare for application exit, do clean-up '''
    my_timer.cancel()
    rot_timer.cancel()
    app_lock.signal()

# Initialize application
appuifw.app.screen = 'full'
canvas = appuifw.Canvas(redraw_callback = cb_handle_redraw)
appuifw.app.body = canvas
appuifw.app.exit_key_handler = cb_quit
appuifw.app.focus = cb_focus
appuifw.app.title = u"Watch Me - Light Touch";

# Setup global variables with screen max resolution
g_maxx, g_maxy = canvas.size
y1 = g_maxy/3
y2 = 2 * y1

# Define touchable areas
# HOX: additional code left in comments to help further expiriments

# Blue vertical box
canvas.bind(key_codes.EButton1Down, cb_blue_down, ((0,y2+1), (g_maxx,g_maxy)))
canvas.bind(key_codes.EButton1Up, cb_blue_up, ((0,y2+1), (g_maxx,g_maxy)))
canvas.bind(key_codes.EDrag, cb_blue_drag, ((0,y2+1), (g_maxx,g_maxy)))
#canvas.bind(key_codes.ESwitchOn, lambda:set_fullcolor(RGB_BLUE), ((0,y2+1),
(g_maxx,g_maxy)))
#canvas.rectangle(((0,y2+1), (g_maxx,g_maxy)), fill=RGB_BLUE, width=5)

# Green vertical box
canvas.bind(key_codes.EButton1Down, cb_green_down, ((0,y1+1), (g_maxx,y2)))
#canvas.bind(key_codes.EButton1Up, cb_green_up, ((0,y1+1), (g_maxx,y2)))
canvas.bind(key_codes.EDrag, cb_green_drag, ((0,y1+1), (g_maxx,y2)))
#canvas.bind(key_codes.ESwitchOn, lambda:set_fullcolor(RGB_GREEN), ((0,y1+1),
(g_maxx,y2)))
#canvas.rectangle(((0,y1+1), (g_maxx,y2)), fill=RGB_GREEN, width=5)

# Red vertical box
canvas.bind(key_codes.EButton1Down, cb_red_down, ((0,0), (g_maxx,y1)))
#canvas.bind(key_codes.EButton1Up, cb_red_up, ((0,0), (g_maxx,y1)))
canvas.bind(key_codes.EDrag, cb_red_drag, ((0,0), (g_maxx,y1)))
#canvas.bind(key_codes.ESwitchOn, lambda:set_fullcolor(RGB_RED), ((0,0), (g_maxx,y1)))
#canvas.rectangle(((0,0), (g_maxx,y1)), fill=RGB_RED, width=5)
```


(C) Copyright Nokia 2014. All rights reserved.

```
handle_screensaver()

# Wait for user to do anything
app_lock = e32.Ao_lock()
app_lock.wait()
```

Capturas de tela



Links Relacionados

PySymbian for 5th Edition + Sample applications

How to check for touch support in PySymbian

(C) Copyright Nokia 2014. All rights reserved.