



ID		Creation date	June 13, 2009
Platform	S60 3rd Edition	Tested on devices	Nokia N82
Category	Python	Subcategory	ui, graphics

Keywords (APIs, classes, methods, functions): TopWindow, graphics, Image, progressbar

Introduction

Here's an implementation of a progressbar using topwindow in Python.

Usage

```
pb = ProgressBar()
pb.set_text(u"Please wait...")
i = 0
for i in range(100):
    pb.set_value(i)
pb.close()
del pb
```

Example Screenshots





The Code

Obs.: As you can notice, I'm also using the canvas toolbar made by Marcelo Barros (Link: [Toolbar on canvas for touch and non touch S60 devices](#))

```
from graphics import *
from e32 import *
from TopWindow import *
from sysinfo import display_pixels

class ProgressBar(object):
    """ Implements a ProgressBar on TopWindow
    """
    def __init__(self, start=0, end=100, color=(0,0,77),
                 fill=(255,255,200), outline=(0,0,0)):
        screen_size = display_pixels()
        #sizes & positions
        self.height = 60
        self.width = int(screen_size[0] * 0.8)
        self.top = screen_size[1] - self.height - 5
        self.left = int((screen_size[0] - self.width) / 2)
        #ProgressBar size
        self.progress_margin = 5
        self.progress_w = self.width - (2 * self.progress_margin)
        self.progress_h = 18 #height of progressbar
        self.progress_l = self.progress_margin
        self.progress_t = self.height - self.progress_h - \
            self.progress_margin
        #internal progressbar expects that external has 1px border
        self.internal_w_max = self.progress_w - 2
        self.internal_h = self.progress_h - 2
        self.internal_l = self.progress_l + 1
        self.internal_t = self.progress_t + 1
        self.internal_w = 0
        self.glow_h = int(self.internal_h / 2)
        #colors & values
        self.start = start
```

A_simple_class_to_implement_a_progressbar_in_Python

```
self.end = end
self.value = start
self.color = color
self.glow_color = self.color_combine(color, (255,255,255), 0.5)
self.outline = outline
self.fill = fill
#text attributes
self.caption = u""
self.font = (u"dense", 12, FONT_ANTI_ALIAS)
self.text_top = self.progress_t - \
                self.progress_margin - \
                self.font[1]

#create topwindow
self.window = TopWindow()
self.window.corner_type = 'square'
self.window.position = (self.left, self.top)
self.window.size = (self.width, self.height)
self.canvas = Image.new(self.window.size)
self.window.add_image(self.canvas, (0,0,self.width,self.height))
#shows initial progressbar
self.redraw()
self.window.show()

def close(self):
    #Closes the window and frees the image buffers memory
    del self.canvas
    self.window.hide()
    del self.window

def set_text(self, text):
    self.caption = text
    self.redraw()

def set_value(self, value):
    if value > self.end:
        value = self.end
    elif value < self.start:
        value = self.start
    self.value = value
    self.internal_w = int(((1.0 * self.value - self.start) / \
                          (1.0 * self.end - self.start)) \
                          * self.internal_w_max)

    self.redraw()

def redraw(self):
    #You don't need call redraw on application. Just use set_value to redraw the progressbar
    #external window
    self.canvas.rectangle((0, 0, self.width, self.height),
                          outline=self.outline,
                          fill=self.fill)

    #progressbar external border
    self.canvas.rectangle((self.progress_l,
                          self.progress_t,
                          self.progress_l + self.progress_w,
                          self.progress_t + self.progress_h),
                          outline=self.outline,
                          fill=self.fill)

    #progressbar core with glow
    self.canvas.rectangle((self.internal_l,
                          self.internal_t,
                          self.internal_l + self.internal_w,
                          self.internal_t + self.internal_h),
```

A_simple_class_to_implement_a_progressbar_in_Python

```
        outline=None,
        fill=self.color)
self.canvas.rectangle((self.internal_l,
                      self.internal_t,
                      self.internal_l + self.internal_w,
                      self.internal_t + self.glow_h),
                    outline=None,
                    fill=self.glow_color)

#window caption
self.canvas.text((self.progress_margin, self.text_top),
                self.caption, fill = self.outline,
                font = self.font)

#exchange images
self.window.remove_image(self.window.images[0][0])
self.window.add_image(self.canvas, (0,0,self.width,self.height))
ao_sleep(0.001)

def color_combine(self, c1, c2, perc):
    return tuple(int(round(a*(1-perc))) + int(round(b*perc)) for a,b in zip(c1, c2))
```