

<b>ID</b>		<b>Creation date</b>	April 27, 2009
<b>Platform</b>	S60 3rd Edition	<b>Tested on devices</b>	Nokia N82
<b>Category</b>	Python	<b>Subcategory</b>	graphics

**Keywords (APIs, classes, methods, functions):** canvas,graphics,images

## Introdução

Apesar do PyS60 não ser capaz de tratar imagens com informação de alfa (32 bits RGBA) existe uma maneira de exibir imagens GIF e PNG transparentes, usando o parâmetro mask no método blit do Image.

Suponha que você tenha feito um relógio no Inkscape com um fundo transparente. Depois o exportou para PNG e gostaria de colocá-lo sobreposto a um fundo com um degradê do verde para o amarelo.

A primeira idéia que viria à cabeça de qualquer um seria escrever o código abaixo (Supondo que você tem as imagens no diretório e:\python\lib):

```
self.gradient = Image.open(u'e:\\python\\lib\\gradient.png')
self.clock_img = Image.open(u'e:\\python\\lib\\clock.png')
self.canvas.clear()
self.canvas.blit(self.gradient)
self.canvas.blit(self.clock_img)
```

Apesar de tudo parecer estar certo, o resultado pode não ser o que você estava esperando:



A solução para esse problema do fundo ficar preto é o parâmetro mask no método blit, para ajudar à biblioteca graphics a saber aonde estão os pixels transparentes.

## Como\_exibir\_imagens\_PNG\_no\_canvas

A máscara (mask) pode ser uma imagem em preto e branco (1-bit de profundidade) ou em tons de cinza (8-bits de profundidade). O método **não** aceita imagens com 12, 16 e 24 bits de profundidade.

Ao procurar aqui no Wiki do Forum Nokia por um exemplo de máscara automática, você vai encontrar a função abaixo, que cria uma máscara usando como cor transparente o primeiro pixel da imagem.

A função pode ser encontrada nessa página "[How to display transparent image on canvas](#)".

```
def automask(im):
    width, height = im.size          # tamanho da imagem
    mask = Image.new(im.size, '1')  # cria uma máscara em preto e branco
    tran = im.getpixel((0,0))[0]    # pega a cor do primeiro pixel
    for y in range(height):
        line = im.getpixel([(x, y) for x in range(width)])
        for x in range(width):
            if line[x] == tran:
                mask.point((x,y), 0) # cria a máscara
    return mask
```

Um exemplo de uso dessa função com o parâmetro mask pode ser o que está logo a seguir:

```
self.gradient = Image.open(u'e:\\python\\lib\\gradient.png')
self.clock_img = Image.open(u'e:\\python\\lib\\clock.png')
self.mask = automask(self.clock_img)
self.canvas.clear()
self.canvas.blit(self.gradient)
self.canvas.blit(self.clock_img, mask=self.mask)
```

Mesmo sendo mil vezes melhor que o fundo totalmente preto, o efeito não é ainda o esperado, já que a imagem fica serrilhada (sem anti-alias) e não tem a sombra original, como podemos ver abaixo.

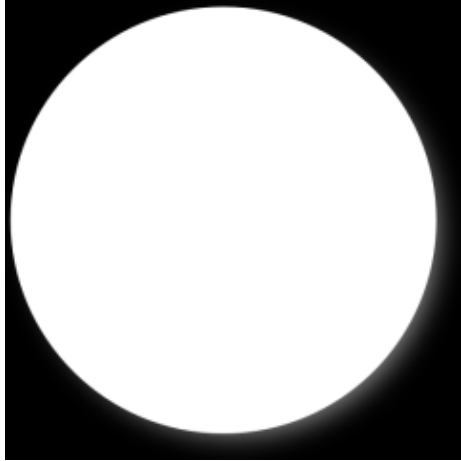


A solução definitiva para acabar com o serrilhado e poder exibir pixels com transparência parcial é criar uma máscara em tons de cinza ao invés de preto e branco. Os pixels pretos serão 100% transparentes e os brancos

## Como\_exibir\_imagens\_PNG\_no\_canvas

0%. Os cinza serão usados para transparência parcial, de acordo com a proximidade do preto.

Usando o seu programa de processamento de imagens (no meu caso era o Gimp), você pode extrair só o canal alfa da imagem e exportá-lo para tons de cinza. Depois basta inverter a imagem para o branco virar preto e vice-versa, como a imagem abaixo:



Com essa máscara, você pode usar o código abaixo para aplicar a transparência no PNG:

```
self.gradient = Image.open(u'e:\\python\\lib\\gradient.png')
self.clock_img = Image.open(u'e:\\python\\lib\\clock.png')
mask = Image.open(u'e:\\python\\lib\\clock-mask.png')
self.mask = Image.new(mask.size, 'L') #tons de cinza 8-bits
self.mask.blit(mask) #converte a imagem para 8-bits
self.canvas.clear()
self.canvas.blit(self.gradient)
self.canvas.blit(self.clock_img, mask=self.mask)
```

O resultado é mostrado abaixo:



## Como\_exibir\_imagens\_PNG\_no\_canvas

Repare que a imagem do relógio ficou perfeitamente combinada com o fundo, e a sombra no canto inferior direito passou a aparecer.

### **Artigo em Inglês**

[How to display transparent PNG on canvas with masks](#)