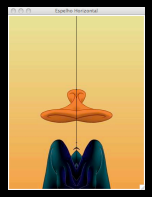
Imagens

preto e branco

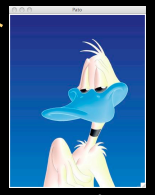
roda 90

sepia



espelho

estica



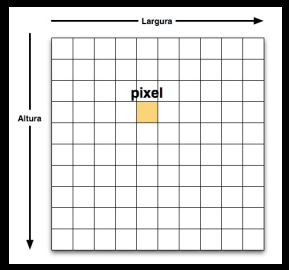
negativo

Imagens

Digitais

Pixel

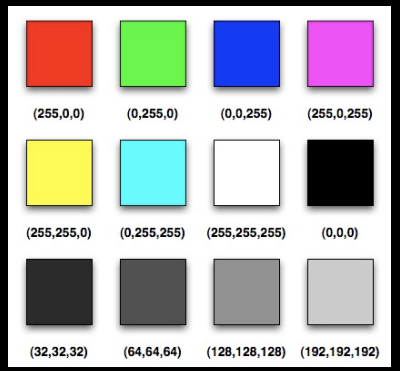
Resolução



Imagens

Cor

Modelo RGB



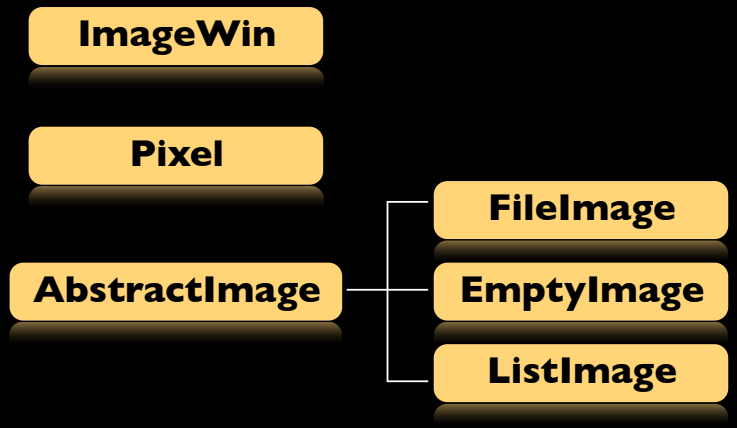
$256^3 = 16\ 777\ 216$

Escala **cinzentos**

cImage

Um módulo Python

Tipos de Objectos



ImageWin

Métodos

Nome	Exemplo	Explicação
ImageWin(titulo,largura,altura)	ImageWin('Teste',800,600)	Cria uma janela (**construtor**)
exitOnClick()	janela.exitOnClick()	Fecha a janela e termina quando clico rato na janela
getMouse()	pos=janela.getMouse()	Devolve o par (x,y) de coordenadas do rato

Pixel

Métodos

Nome	Exemplo
Pixel(r,g,b)	p=Pixel(25,100,0)
getRed()	r = p.getRed()
getGreen()	g = p.getGreen()
getBlue()	b = p.getBlue()
setRed(r)	p.setRed(25)
setGreen(g)	p.setGreen(100)
setBlue(b)	p.setBlue(0)

construtor

FileImage EmptyImage

Métodos

Nome	Exemplo
FileImage(nome_ficheiro) construtor	img=FileImage('toto.jpg')
EmptyImage(largura,altura) construtor	img=EmptyImage(largura,altura)
draw(janela)	img.draw(janela)
save(nome_ficheiro)	img.save('toto.jpg')

FileImage EmptyImage

Métodos

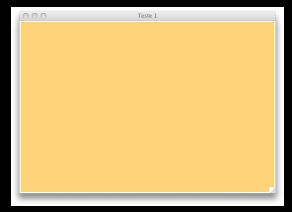
Nome	Exemplo
getWidth()	larg=img.getWidth()
getHeight()	alt=img.getHeight()
setPixel(col,lin,pix)	img.setPixel(10,100,Pixel(1,10,5))
setPosition(col,lin)	img.setPosition(100,50)

Mostra Imagens

```
import cImage

def mostra_imagem(img_fich):
    imagem = cImage.FileImage(img_fich)
    largura = imagem.getWidth()
    altura = imagem.getHeight()

    janela = cImage.ImageWin('Pato', largura,altura)
    imagem.draw(janela)
    janela.exitOnClick()

if __name__ =='__main__':
    mostra_imagem('/tempo/imagens/duck3.jpg')
```

Cria Imagem

```
def cria_imagem_cor_1(largura,altura):
    janela = cImage.ImageWin('Teste 1',largura,altura)
    imagem = cImage.EmptyImage(largura,altura)
    pixel_cor = cImage.Pixel(255,204,102)
    for coluna in range(largura):
        for linha in range(altura):
            imagem.setPixel(coluna,linha,pixel_cor)
    imagem.draw(janela)
    janela.exitOnClick()
```

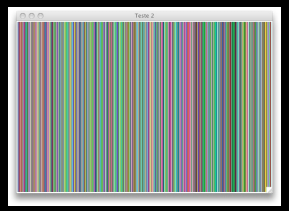


Ciclos Imbricados

Cria Imagem

Ciclos Imbricados

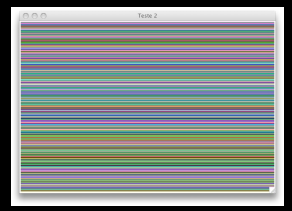
```
def cria_imagem_cor_2(largura,altura):
    janela = cImage.ImageWin('Teste 2',largura,altura)
    imagem = cImage.EmptyImage(largura,altura)

    for coluna in range(largura):
        r = random.randint(0,255)
        g = random.randint(0,255)
        b = random.randint(0,255)
        pixel = cImage.Pixel(r,g,b)
        for linha in range(altura):
            imagem.setPixel(coluna,linha,pixel)
    imagem.draw(janela)
    janela.exitOnClick()
```



Cria Imagem

Ciclos Imbricados

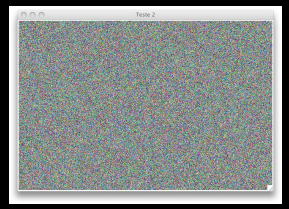
```
def cria_imagem_cor_4(largura,altura):
    janela = cImage.ImageWin('Teste 2',largura,altura)
    imagem = cImage.EmptyImage(largura,altura)

    for linha in range(altura):
        r = random.randint(0,255)
        g = random.randint(0,255)
        b = random.randint(0,255)
        pixel = cImage.Pixel(r,g,b)
        for coluna in range(largura):
            imagem.setPixel(coluna,linha,pixel)
    imagem.draw(janela)
    janela.exitOnClick()
```



Cria Imagem

Ciclos Imbricados

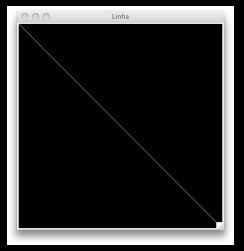
```
def cria_imagem_cor_3(largura,altura):
    janela = cImage.ImageWin('Teste 2',largura,altura)
    imagem = cImage.EmptyImage(largura,altura)

    for coluna in range(largura):
        for linha in range(altura):
            r = random.randint(0,255)
            g = random.randint(0,255)
            b = random.randint(0,255)
            pixel = cImage.Pixel(r,g,b)
            imagem.setPixel(coluna,linha,pixel)
    imagem.draw(janela)
    janela.exitOnClick()
```



Cria Imagem

Linhas

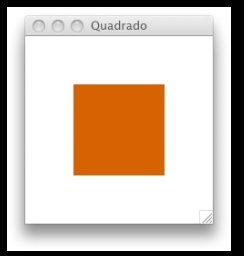
```
def desenha_linha(largura,altura,pixel):
    """
    Cria uma imagem com as dimensões largura x altura
    e desenha um diagonal com a cor do pixel.
    """
    janela = cImage.ImageWin('Linha', largura, altura)
    imagem = cImage.EmptyImage(largura,altura)
    for coluna in range(largura):
        for linha in range(altura):
            if coluna == linha:
                imagem.setPixel(coluna,linha,pixel)
    imagem.draw(janela)
    imagem.save("/tempo/imagens/linha.jpg")
    janela.exitOnClick()
```



Cria Imagem

Figuras

```
def desenha_quadrado_cheio(posx,posy, lado):
    """
    Desenha um quadrado cheio na janela de lado com o canto superior
    esquerdo em (posx,posy).
    """
    janela=cImage.ImageWin('Quadrado',2*lado,2*lado)
    imagem = cImage.EmptyImage(lado,lado)
    imagem.setPosition(posx,posy)
    p = cria_random_pixel()
    for linha in range(lado):
        for coluna in range(lado):
            imagem.setPixel(coluna,linha,p)
    imagem.draw(janela)
    janela.exitOnClick()
```

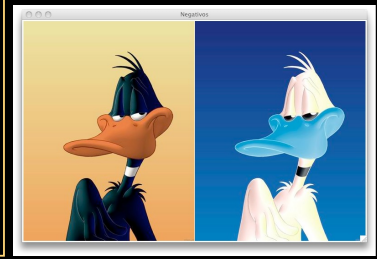


Manipula Imagem

Negativo

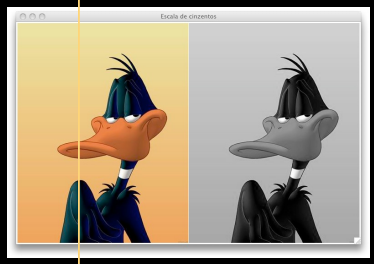
```
def negativo_imagem(imagem_ficheiro):

    imagem_velha = cImage.FileImage(imagem_ficheiro)
    largura = imagem_velha.getWidth()
    altura = imagem_velha.getHeight()
    janela = cImage.ImageWin('Negativos',2*largura,altura)
    imagem_velha.draw(janela)

    imagem_nova = cImage.EmptyImage(largura,altura)

    for coluna in range(largura):
        for  linha in range(altura):
            pixel_original = imagem_velha.getPixel(coluna,linha)
            novo_pixel = negativo_pixel(pixel_original)
            imagem_nova.setPixel(coluna,linha,novo_pixel)
    imagem_nova.setPosition(largura+1,0)
    imagem_nova.draw(janela)
    janela.exitOnClick()
```

```
def negativo_pixel(pixel):
    red = 255 - pixel.getRed()
    green = 255 - pixel.getGreen()
    blue = 255 - pixel.getBlue()
    novo_pixel = cImage.Pixel(red,green,blue)
    return novo_pixel
```



Manipula Imagem
Escala Cinzentos

```
def imagem_cinzentos(imagem_fich):
    """ Transforma para escala de cinzentos a imagem."""

    imagem = cImage.FileImage(imagem_fich)
    largura = imagem.getWidth()
    altura = imagem.getHeight()

    janela = cImage.ImageWin('Escala de cinzentos', 2* largura, altura)
    imagem.draw(janela)
    nova_imagem=cImage.EmptyImage(largura,altura)

    for coluna in range(largura):
        for linha in range(altura):
            pixel = imagem.getPixel(coluna,linha)
            novo_pixel = pixel_cinzento(pixel)
            nova_imagem.setPixel(coluna,linha,novo_pixel)
    nova_imagem.setPosition(largura+1,0)
    nova_imagem.draw(janela)
    janela.exitOnClick()
```



Manipula Imagem
Escala Cinzentos

```
def pixel_cinzento(pixel):
    """ Converte um pixel para escala de cinzentos."""
    vermelho = pixel.getRed()
    verde = pixel.getGreen()
    azul = pixel.getBlue()

    int_media = (vermelho + verde + azul) / 3
    novo_pixel = cImage.Pixel(int_media,int_media, int_media)
    return novo_pixel
```

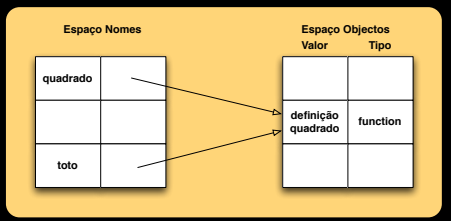

Abstração
Que diferenças?

```
...
for coluna in range(largura):
    for  linha in range(altura):
        pixel_original = imagem_velha.getPixel(coluna,linha)
        novo_pixel = negativo_pixel(pixel_original)
        imagem_nova.setPixel(coluna,linha,novo_pixel)
...
```

```
...
for coluna in range(largura):
    for linha in range(altura):
        pixel = imagem.getPixel(coluna,linha)
        novo_pixel = pixel_cinzento(pixel)
        nova_imagem.setPixel(coluna,linha,novo_pixel)
...
```


Abstração
Funções como argumento

```
>>> def quadrado(n):
...     return n * n
...
>>> quadrado(4)
16
>>> quadrado(quadrado(4))
256
>>> id(quadrado)
13432880
>>> quadrado
<function quadrado at 0xccf830>
>>> type(quadrado)
<type 'function'>
>>> toto = quadrado
>>> toto(8)
64
>>> id(toto)
13432880
>>> toto
<function quadrado at 0xccf830>
>>> type(toto)
<type 'function'>
>>>
```



Abstração

Funções como argumento

```
def transforma(imagem_fich, funcao_cor):
    """ Transforma uma imagem de acordo com a função de cor."""
    imagem = cImage.FileImage(imagem_fich)
    largura = imagem.getWidth()
    altura = imagem.getHeight()
    janela = cImage.ImageWin('Transformação de Imagem', 2*largura,altura)
    imagem.draw(janela)

    nova_imagem = manipula_imagem(imagem,funcao_cor)
    nova_imagem.setPosition(largura + 1, 0)
    nova_imagem.draw(janela)
    janela.exitOnClick()
```


Abstração

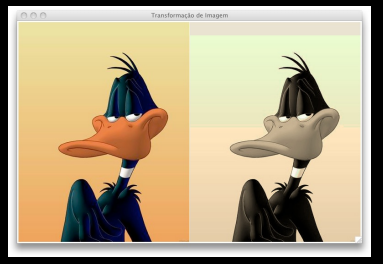
Funções como argumento

```
def manipula_imagem(imagem, funcao_cor):
    """ Manipula uma imagem de acordo com uma
função."""
    largura = imagem.getWidth()
    altura = imagem.getHeight()
    nova_imagem = cImage.EmptyImage(largura,altura)

    for coluna in range(largura):
        for linha in range(altura):
            pixel = imagem.getPixel(coluna,linha)
            novo_pixel = funcao_cor(pixel)
            nova_imagem.setPixel(coluna,linha, novo_pixel)
    return nova_imagem
```


Manipula Imagem

Sepia

```
def pixel_sepia(pixel):
    """ Tempo do passado."""
    r = pixel.getRed()
    g = pixel.getGreen()
    b = pixel.getBlue()

    novo_r = (r * 0.393 + g * 0.769 + b * 0.189)
    novo_g = (r * 0.349 + g * 0.686 + b * 0.168)
    novo_b = (r * 0.272 + g * 0.534 + b * 0.131)
    if novo_r > 255: novo_r = r
    if novo_g > 255: novo_g = g
    if novo_b > 255: novo_b = b

    novo_pixel = cImage.Pixel(novo_r,novo_g,novo_b)
    return novo_pixel
```



Manipula Imagem

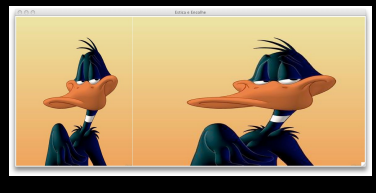
Distorcer

```
def altera(imagem,factor_x, factor_y):
    """Altera a imagem de acordo com os factores.
    Estes devem ser inteiros.
    """
    largura = imagem.getWidth()
    altura = imagem.getHeight()
    nova_imagem = cImage.EmptyImage(factor_x * largura, factor_y * altura)

    for coluna in range(largura):
        for linha in range(altura):
            pixel = imagem.getPixel(coluna, linha)

            for i_x in range(factor_x):
                for i_y in range(factor_y):
                    nova_imagem.setPixel(factor_x * coluna + i_x, factor_y * linha + i_y, pixel)
    return nova_imagem
```

Manipula Imagem

Distorcer

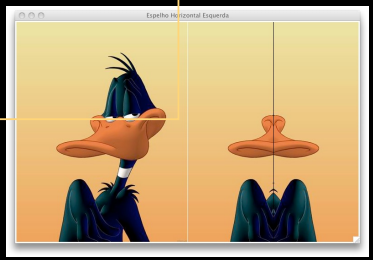
```
janela = cImage.ImageWin('Estica e Encolhe', 3 *375,480)
imagem = cImage.FileImage('/tempo/imagens/duck3.jpg')
imagem.draw(janela)

nova_img = altera(imagem, 2,1)
nova_img.setPosition(375 + 1,0)
nova_img.draw(janela)
janela.exitOnClick()
```



Manipula Imagem

Espelho

```
def espelho_h_e(imagem_fich):
    """Faz o espelho orizontal de uma imagem. Usa a parte esquerda."""
    imagem = cImage.FileImage(imagem_fich)
    largura = imagem.getWidth()
    altura = imagem.getHeight()
    janela = cImage.ImageWin('Espelho Horizontal Esquerda', 2*largura,altura)
    imagem.draw(janela)
    nova_imagem = cImage.EmptyImage(largura,altura)
    for coluna in range(largura/2):
        for linha in range(altura):
            pixel = imagem.getPixel(coluna, linha)
            nova_imagem.setPixel(coluna,linha,pixel)
            nova_imagem.setPixel(largura - coluna - 1,linha,pixel)
    nova_imagem.setPosition(largura + 1, 0)
    nova_imagem.draw(janela)
    janela.exitOnClick()
```



Extracção Características

Detecção de Lados

Zonas que separam intensidades

Kernel ou filtros

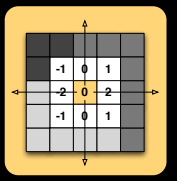
Convolução

Kernel_x		
-1	0	1
-2	0	2
-1	0	1

Kernel_y		
1	2	1
0	0	0
-1	-2	-1

Extracção Características

Detecção de Lados



Algoritmo

1. Converter a imagem para excala de cinzentos
2. Criar uma imagem vazia de igual dimensão
3. Processar cada pixel
   (a) Convolução com Kernel_x
   (b) Convolução com Kernel_y
   (c) Acha raíz quadrada da soma

Extracção Características

Detecção de Lados

```
def detecta_lados(imagem, limiar):
    """Fabrica uma imagem em que os lados foram detectados. Usa
operadores de Sobel."""
    imagem_cinza = manipula_imagem(imagem,pixel_cinzento)
    largura = imagem_cinza.getWidth()
    altura = imagem_cinza.getHeight()
    nova_imagem = cImage.EmptyImage(largura, altura)
    preto = cImage.Pixel(0,0,0)
    branco = cImage.Pixel(255,255,255)
    kernel_x = [[-1,-2,-1],[0,0,0],[1,2,1]]
    kernel_y = [[-1,0,1],[-2,0,2],[-1,0,1]]

    for linha in range(1, altura - 1):
        for coluna in range(1, largura - 1):
            cx = convolve(imagem_cinza, linha, coluna, kernel_x)
            cy = convolve(imagem_cinza, linha, coluna, kernel_y)
            c = math.sqrt(cx ** 2 + cy ** 2)

            if c > limiar:
                nova_imagem.setPixel(coluna, linha, preto)
            else:
                nova_imagem.setPixel(coluna, linha, branco)
    return nova_imagem
```


Extracção Características

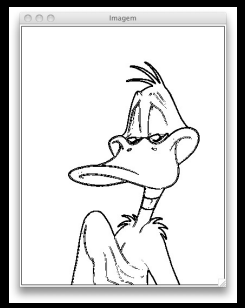
Detecção de Lados

```
def convolve(imagem, pix_linha, pix_coluna, kernel):
    """ Calcula a convolução de um pixel."""
    kernel_coluna_base = pix_coluna - 1
    kernel_linha_base = pix_linha - 1

    soma = 0
    for linha in range(kernel_linha_base, kernel_linha_base + 3):
        for coluna in range(kernel_coluna_base,kernel_coluna_base + 3):
            k_coluna_indice = coluna - kernel_coluna_base
            k_linha_indice = linha - kernel_linha_base

            pixel = imagem.getPixel(coluna, linha)
            intensidade = pixel.getRed()

            soma = soma + intensidade * kernel[k_linha_indice][k_coluna_indice]
    return soma
```

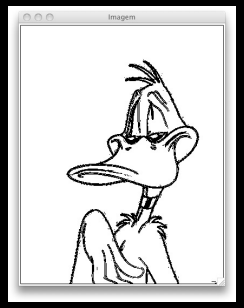

Extracção Características

Detecção de Lados

Limiar = 175



Limiar = 100



Limiar = 50