Representação de imagens

Aplicação C++/Assembly para manipulação de imagens

João Canas Ferreira

Março 2014

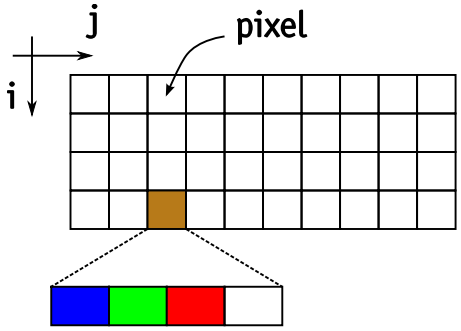
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Imagens

- Como são representadas imagens de cor?

Matriz de pixels

- Uma imagem é uma matriz bidimensional de pontos
  - cada ponto é um *pixel* (picture element)



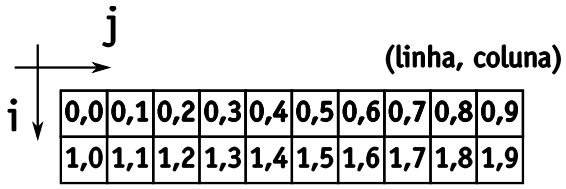
Representação de pixel

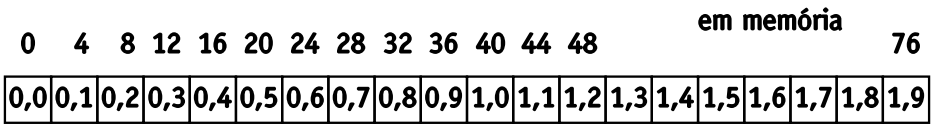
- Componente R (vermelho): 1 byte (0–255)
- Componente G (verde): 1 byte (0–255)
- Componente B (azul): 1 byte (0–255)
- Opacidade (alfa): 1 byte (0: transparente, 255: opaco)

➠ Cada pixel ocupa 1 DWORD (4 bytes)

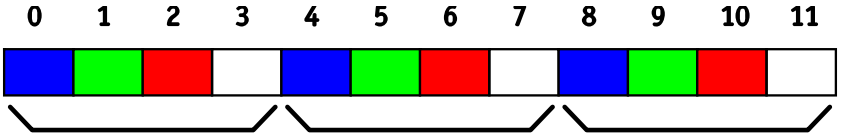
Organização da imagem em memória

Linhas sucessivas ficam seguidas em memória





Ordem das componentes



Cálculo de posições em memória

Caraterísticas da imagem

- b: posição (endereço) do início da imagem em memória
- p: nº de bytes de um pixel (=4, no formato usado)
- L: largura da imagem (número de pixels na horizontal)
- A: altura da imagem (número de pixels na vertical)
- pixel na linha i, coluna j com $0 \leq i < A$ e $0 \leq j < L$

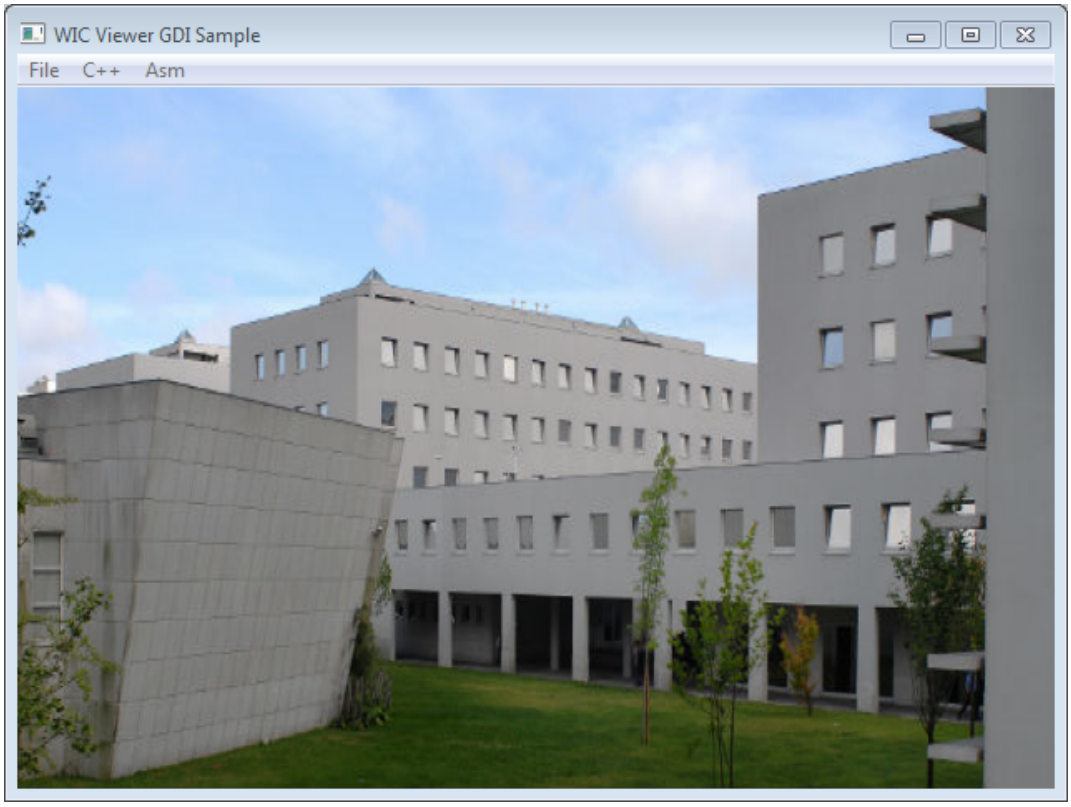
Fórmulas para o cálculo de posições em memória

- início da linha i: $b + i \times L \times p$
- pixel na linha i, coluna j: $b + (i \times L + j) \times p$

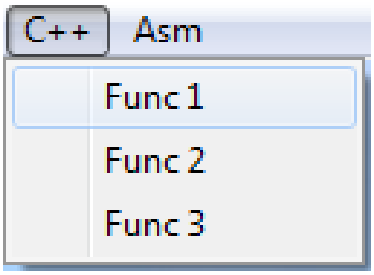
➠ Endereço do vizinho superior do pixel P: subtrair $L \times p$ ao endereço de P

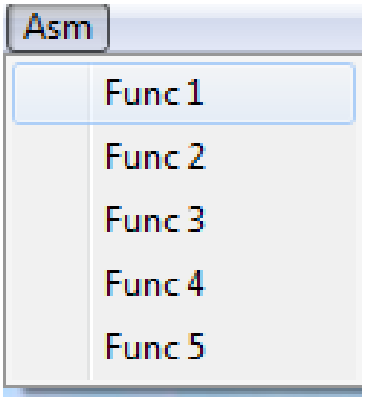
➠ Endereço do vizinho inferior de P: somar $L \times p$ ao endereço de P

Programa Viewer



Funções





Declaração em C++:

```
void cfunc1(unsigned char *pixels, long largura, long altura);
```

Declaração em *assembly language:*

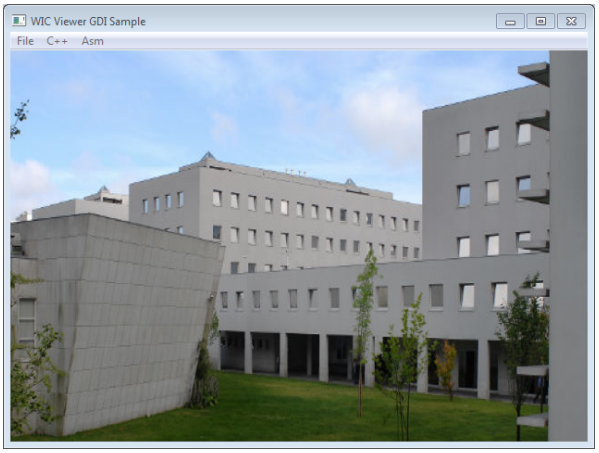
```
afunc1 PROTO pixels:ptr byte, largura:dword, altura:dword
```


Exemplo: Conversão de cor para níveis de cinzento

Conversão de cor para cinzento

- Substituir os componentes R, G e B pela sua média.

```
para todos os pixels da imagem:
   media = (R + G + B) / 3
   R = media
   G = media
   B = media
```



Exemplo em C++

```
typedef unsigned int media_t;
void cfunc1(unsigned char *pixels, long largura, long altura)
{
  unsigned char *linha;
  for (int j = 0; j < altura; j++) {
     linha = pixels + (j*largura*BYTES_PER_PIXEL);
     for (int i = 0; i < largura*BYTES_PER_PIXEL;
                       i += BYTES_PER_PIXEL) {
       media_t media;
       media = (media_t) linha[i] + (media_t) linha[i+1]
                                  + (media_t) linha[i+2];
       media = media / 3;
       linha[i] = (unsigned char) media;        // Azul (B)
       linha[i+1] = (unsigned char) media;      // Verde (G)
       linha[i+2] = (unsigned char) media;      // Vermelho (R)
       linha[i+3] = 255;   // irrelevante neste caso
    }
  }
  return;
}
```


Exemplo em assembly

```
afunc1 PROC USES edi ebx pixels: ptr byte, largura: dword, altura:dword
   mov  eax,largura
   mul  altura          ; EAX = número de pixels
   mov  ecx, eax
   mov  edi, pixels
   .WHILE (ecx > 0)
        movzx   eax, byte ptr [edi]     ;; componente B
        movzx   ebx, byte ptr [edi+1]   ;; componente G
        add     eax, ebx
        movzx   ebx, byte ptr [edi+2]   ;; componente R
        add     eax, ebx
        xor     edx, edx
        div     val3                    ;; divisão por 3
        mov     [edi], al
        mov     [edi+1], al
        mov     [edi+2], al
        add     edi, 4              ;;  BYTES per PIXEL
        dec     ecx
   .ENDW
   ret
afunc1 ENDP
```