Introdução à Biblioteca Allegro

Tiago da Conceição Mota
tiagomt@gmail.com

Instalação

- http://alleg.sourceforge.net
- http://www.allegro.cc, Files
- Opções para download
  - Fonte
    - Comando **`make`** pode demorar
    - Compilado com otimizações para a máquina do usuário
  - Pré-compilado
    - É necessário apenas descompactar os arquivos para o diretório do compilador
    - Sem otimizações específicas

Primeiros Passos

- Incluir o *header* da biblioteca
  - **`#include <allegro.h>`**
- Iniciar a biblioteca
  - **`allegro_init();`**
- Finalizar a biblioteca
  - **`allegro_exit();`**
- Marcar o final da função **`main`**
  - **`END_OF_MAIN();`**
- Compilar o programa, linkando a biblioteca
  - **`gcc -o prog prog.c –Wall –lalleg`**
- Exemplo 1

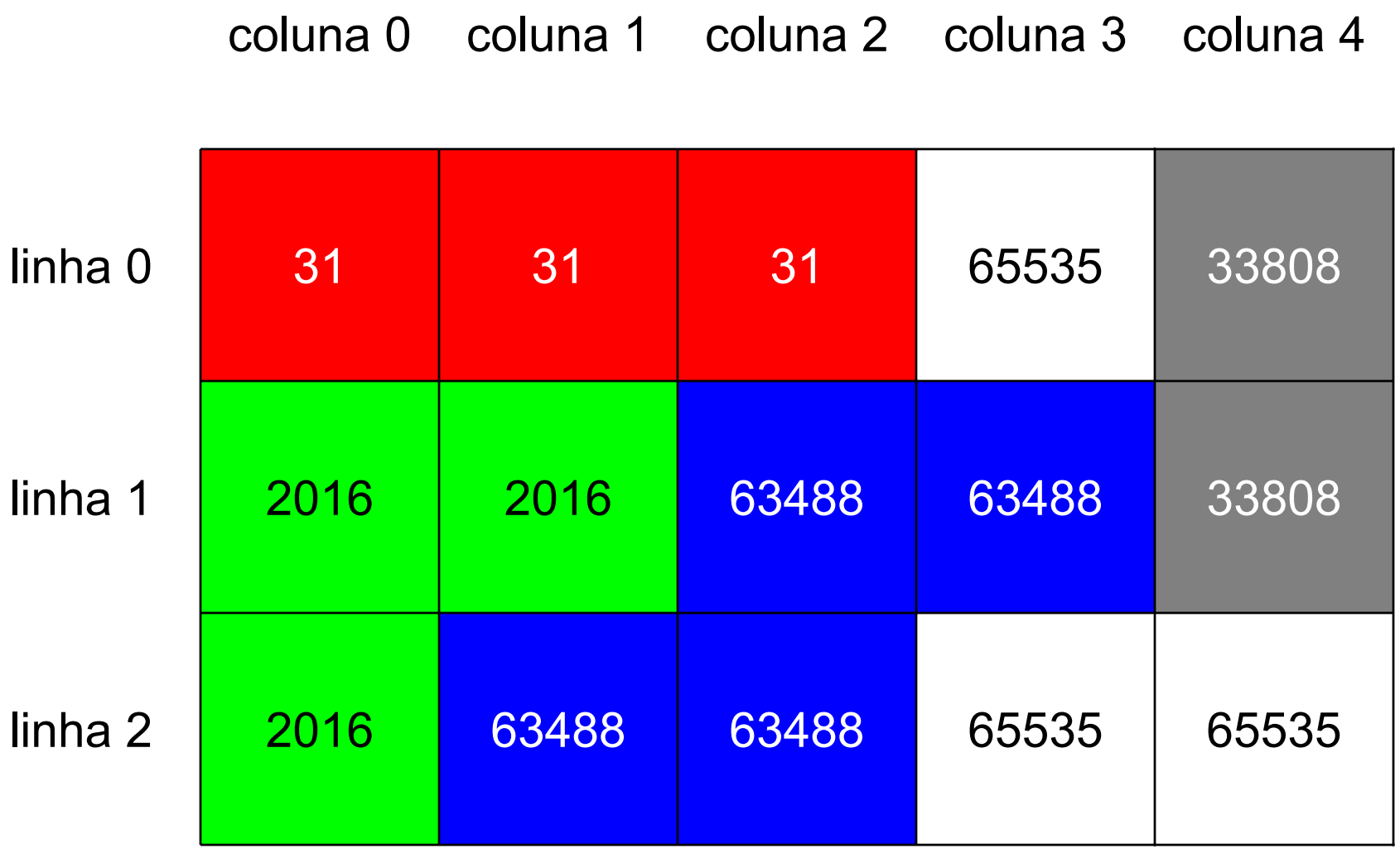
Modo Gráfico

- **`set_color_depth(depth);`**
  - Determina o número de cores do modo gráfico
  - **`depth`** pode ser 8, 15, 16, 24 ou 32
- **`set_gfx_mode(GFX_AUTODETECT, RESOL_X, RESOL_Y, 0, 0);`**
  - Inicia o modo gráfico
  - **`RESOL_X`** e **`RESOL_Y`** correspondem, respectivamente, à largura e à altura da tela
  - É importante verificar o valor de retorno da função (negativo em caso de erro)
- Exemplo 2

Bitmaps

- *Bitmap* é uma matriz de pontos, cada um com um valor indicando a cor do ponto
- No Allegro, todo desenho é feito em *bitmaps*
- A tela é representada por um *bitmap* já declarado pelo Allegro, de nome **`screen`**
- A coordenada (**`x`**, **`y`**) corresponde ao ponto localizado na coluna **`x`**, na linha **`y`**.
- A coordenada (**`0`**, **`0`**) corresponde ao ponto do canto superior esquerdo do *bitmap*

Bitmaps


coluna 0
coluna 1
coluna 2
coluna 3
coluna 4
linha 0
linha 1
linha 2
31
31
31
65535
33808
2016
2016
63488
63488
33808
2016
63488
63488
65535
65535

Primitivas de Desenho

- **`makecol(r, g, b);`**
  - Retorna o código para a cor a partir dos valores de **`r`** (vermelho), **`g`** (verde) e **`b`** (azul)
  - Estes valores devem estar entre 0 e 255 (inclusive)
- **`clear_to_color(bitmap, color);`**
  - Limpa **`bitmap`** para a cor **`color`**
- **`putpixel(bitmap, x, y, color);`**
  - Desenha um ponto na coordenada (**`x`**, **`y`**) de **`bitmap`** com a cor **`color`**

Primitivas de Desenho

- **`hline(bitmap, x1, y, x2, color);`**
  - Desenha uma reta horizontal da coordenada (**`x1`**, **`y`**) à coordenada (**`x2`**, **`y`**), com cor **`color`**, em **`bitmap`**
- **`vline(bitmap, x, y1, y2, color);`**
  - Desenha uma reta vertical da coordenada (**`x`**, **`y1`**) à coordenada (**`x`**, **`y2`**)
- **`line(bitmap, x1, y1, x2, y2, color);`**
  - Desenha uma reta da coordenada (**`x1`**, **`y1`**) à coordenada (**`x2`**, **`y2`**)
- Exemplo 3

Primitivas de Desenho

- **`rect(bitmap, x1, y1, x2, y2, color);`**
  - Desenha um retângulo cujos vértices opostos têm coordenadas (**`x1`**, **`y1`**) e (**`x2`**, **`y2`**), com cor **`color`**, em **`bitmap`**
  - A função **`rectfill`** faz o mesmo que **`rect`**, mas preenche a região do interior do retângulo
- **`circle(bitmap, x, y, r, color);`**
  - Desenha um círculo com centro na coordenada (**`x`**, **`y`**), raio **`r`** e cor **`color`**, em **`bitmap`**
  - A função **`circlefill`** faz o mesmo que **`circle`**, mas preenche a região do interior do círculo
- Exemplo 4

Primitivas de Desenho

- Outras primitivas de desenho
  - **`triangle`**
  - **`polygon`**
  - **`ellipse`**, **`ellipsefill`**
  - **`arc`**
  - **`spline`**
  - **`floodfill`**

Manipulação de *bitmaps*

- Declaração, criação e destruição de *bitmaps*

```
BITMAP *bitmap;
bitmap = create_bitmap(width, height);
destroy_bitmap(bitmap);
```

- **`blit(source, dest, sx, sy, dx, dy, width, height);`**
  - Copia a área de **`source`** que começa na coordenada (**`sx`**, **`sy`**) e tem largura **`width`** e altura **`height`**, para **`dest`**, na coordenada (**`dx`**, **`dy`**)
- Exemplo 5

Imagens

- **`bitmap = load_bitmap(filename, NULL);`**
  - Carrega o *bitmap* do arquivo **`filename`** e o guarda em **`bitmap`**
  - A extensão do arquivo (BMP, LBM, PCX ou TGA) define o tipo de arquivo
  - Retorna **`NULL`** caso não consiga carregar o arquivo
- Exemplo 6
- **`draw_sprite(dest, source, x, y);`**
  - Desenha **`source`** em **`dest`**, na coordenada (**`x`**, **`y`**)
- Exemplo 7

Animações

- Método inicial

```
loop
    limpa screen
    desenha em screen
```

- Exemplo 8

Animações

- *Double buffering*

```
cria buffer com mesmo tamanho de screen
loop
   limpa buffer
   desenha em buffer
   copia buffer para screen
```

- Exemplo 9

Animações

- Outros métodos
  - *Page flipping*
  - *Triple buffering*
  - *Dirty rectangles*

Texto

- **`text_mode(color);`**
  - Define a cor a ser usada no fundo dos textos
  - Se **`color`** for negativo, o fundo será transparente
- **`textout(bitmap, font, s, x, y, color);`**
  - Escreve a *string* **`s`** na coordenada (**`x`**, **`y`**) de **`bitmap`**, com cor **`color`** e fonte **`font`**
- **`textprintf(bitmap, font, x, y, color, fmt, ...);`**
  - O mesmo que **`textout`**, mas com saída formatada
- Há, também, as funções **`textout_centre`**, **`textout_right`** e **`textout_justify`**
- Exemplo 10

Teclado

- **`install_keyboard();`**
- **`readkey();`**
  - Retorna a próxima tecla do *buffer* do teclado
  - 2 bytes
    - o mais baixo (do bit 0 ao bit 7) contém o código ASCII
    - o mais alto (do bit 8 ao bit 15) contém o *scancode*

```
val = readkey();
if ((val & 0xFF) == 'b') /* tecla 'b'. */ ;
if ((val >> 8) == KEY_B) /* tecla B. */ ;
if ((val & 0xFF) == 2) /* tecla CTRL+B. */ ;
if (val == (KEY_B << 8)) /* tecla ALT+B. */ ;
```

Teclado

- **`clear_keybuf();`**
  - Limpa o *buffer* do teclado
- **`keypressed();`**
  - Retorna **`TRUE`** se alguma tecla tiver sido pressionada
- **`key[];`**
  - Vetor de **`KEY_MAX`** posições, cada uma contendo o estado de uma tecla (pressionada ou não pressionada)
  - Cada tecla possui uma constante para o acesso ao vetor (por exemplo, **`KEY_SPACE`**, **`KEY_ENTER`**, **`KEY_ESC`**, **`KEY_LEFT`**, **`KEY_Q`**)
- Exemplo 11

Mouse

- **`install_mouse();`**
- **`show_mouse(bitmap);`**
  - Mostra o mouse em **`bitmap`**
  - Antes de desenhar no *bitmap*, deve-se esconder o mouse, com **`show_mouse(NULL)`**
- **`mouse_x; mouse_y;`**
  - Contêm a posição do mouse no *bitmap*
- **`mouse_b;`**
  - Contém o estado dos botões do mouse. O bit 0 corresponde ao botão esquerdo; o bit 1, ao botão direito; e o bit 2, ao botão do meio
- Exemplo 12

Temporizadores

- **`install_timer();`**
- **`install_int(proc, speed);`**
  - Instrui o Allegro a chamar a função **`proc`** a cada **`speed`** milisegundos
- **`remove_int(proc);`**
  - Remove o temporizador da função **`proc`**
- **`END_OF_FUNCTION(proc);`**
  - Marca o final da função **`proc`**
- **`LOCK_FUNCTION(proc);`**
  - Macro que deve ser chamada antes de **`install_int`**

Temporizadores

- **`LOCK_VARIABLE(var);`**
  - Macro que também deve ser chamada antes de **`install_int`**
  - Deve-se utilizar esta macro em todas as variáveis usadas dentro das funções de temporizadores
- Além disso, tais variáveis devem ser declaradas como **`volatile`**, para evitar que o compilador faça otimizações erradas
- Exemplo 13

Velocidade de Execução

- Dependendo da velocidade da máquina, cada iteração do *loop* principal de um programa pode ter um tempo de execução diferente
- Pode-se utilizar temporizadores para controlar a velocidade de execução de um programa, de modo que esta seja independente da velocidade da máquina
- Utiliza-se uma variável para controlar a taxa de atualização do estado do programa (normalmente, 60 vezes por segundo)

Velocidade de Execução

```
função inc_ctrl_vel
  incrementa ctrl_vel
função principal
  inclui temporizador para inc_ctrl_vel
  zera ctrl_vel
  loop
    desenha
    se ctrl_vel > 0 então
      atualiza estado do programa
      decrementa ctrl_vel
```

- Exemplo 14

Som e Música

- **`install_sound(DIGI_AUTODETECT, MIDI_AUTODETECT, NULL);`**
  - Retorna negativo em caso de erro
- Assim como *bitmaps* são acessados através de variáveis do tipo **`BITMAP *`**, sons são acessados através de variáveis do tipo **`SAMPLE *`** e músicas são acessadas através de variáveis do tipo **`MIDI *`**
- **`sample = load_sample(filename);`**
  - Carrega o arquivo de som de nome **`filename`**
  - O arquivo pode ser WAV ou VOC

Som e Música

- **`play_sample(sample, vol, pan, freq, loop);`**
  - Toca o som **`sample`**, com volume **`vol`** e balanço **`pan`**
  - **`freq`** determina em que freqüência o som será tocado, levando-se em conta valores relativos (1000 é a freqüência normal, 2000 é o dobro etc)
  - Se **`loop`** for verdadeiro, o som é tocado em *loop*
- **`stop_sample();`**
  - Interrompe a execução de um som em *loop*
- **`destroy_sample(sample);`**

Som e Música

- **`music = load_midi(filename);`**
  - Carrega o arquivo de múisca de nome **`filename`**
  - O arquivo deve ser do tipo MIDI
- **`play_midi(music, loop);`**
  - Toca o arquivo de música **`music`**
  - Se **`loop`** for verdadeiro, a música será tocada em *loop*
- **`stop_midi();`**
  - Interrompe a execução de uma música em *loop*
- **`destroy_midi(music);`**
- Exemplo 15

Outras funções

- Outras funções dos tópicos anteriores
- *Datafiles*
- Arquivos de configuração
- *Joystick*
- Fontes
- Matemática de ponto fixo
- Gráficos em 3D

Ajuda

- http://alleg.sourceforge.net
- http://www.alleg.cc
  - Manual do Allegro
  - Programas que utilizam Allegro, normalmente com código fonte disponível
  - Fóruns
- http://equipe.nce.ufrj.br/adriano/c/apostila/allegro
- Arquivos que acompanham o Allegro
  - Demo
  - Exemplos