

A UTILIZAÇÃO DA BIBLIOTECA ALLEGRO PARA O DESENVOLVIMENTO DE APLICAÇÕES MULTIMÍDIA

Luana Pereira de Lima
luanaelima@gmail.com

Túlio Vitor Machado Faria
tuliofaria@gmail.com

Faculdade de Administração e Informática

Resumo - O artigo apresenta a biblioteca Allegro, que pode ser usada com a linguagem C/C++ para o desenvolvimento de aplicações multimídia e de jogos. Com a Allegro, podem-se adicionar aos aplicativos, recursos como imagens, sons, temporizadores e entrada de dados via mouse e teclado. Para demonstrar essas funcionalidades utiliza-se o ambiente Dev C++.

Abstract - This work introduces Allegro Library, which can be used with C/C++ language, in the development of multimedia and game applications. Using this library, one can add resources such as images, sounds, timers, mouse and keyboard data entry. Dev C++ Environment is used to demonstrate some of these functionalities.

Palavras-chave - Linguagem C/C++, Allegro, multimídia, jogos.

Keywords - C/C++ Language, Allegro, multimedia, games.

1. INTRODUÇÃO

O objetivo deste artigo é demonstrar as funcionalidades da Allegro, biblioteca criada especialmente para o desenvolvimento de jogos e aplicações multimídia. Essa biblioteca de funções permite adicionar sons, imagens, tratamento de entrada de dados via mouse e teclado, temporizadores, entre outras funcionalidades, nas aplicações desenvolvidas em C/C++.

Neste artigo é mostrado como instalar a biblioteca e como utilizar as funções que permitem inserir imagens e sons, empregar a

técnica de *Double Buffering*, tratar eventos de teclado e de mouse e controlar o tempo por meio de temporizadores.

Espera-se propiciar ao leitor, condições de desenvolver uma aplicação simples com recursos multimídia, utilizando para isso a biblioteca Allegro e a Linguagem C.

2. A BIBLIOTECA ALLEGRO

A Allegro é uma biblioteca multi-plataforma de código aberto utilizada para a programação de jogos e de aplicações multimídia em geral. Foi desenvolvida, inicialmente, para ser utilizada com a plataforma Atari ST, mas depois do desaparecimento dessa, a biblioteca passou pelo Borland C, até ser adotada pelo DJGPP (ambiente com o compilador C/C++), onde a mesma cresceu com sua mesclagem entre código C e Assembly.

Ela é muito conhecida pela sua facilidade em adicionar entrada de dados via teclado, mouse e joystick, áudio, temporizadores e imagens, em programas C e C++, aliada à pluralidade de sistemas operacionais suportados, como DOS, Windows, BeOS, Mac OS X e vários sistemas Unix (ALLEGRO, 2005)

3. INSTALAÇÃO DA ALLEGRO NO AMBIENTE DEV C++

A instalação da biblioteca Allegro no ambiente Dev C++ tem a vantagem de poder ser realizada através do Gerenciador de Pacotes do próprio compilador. Para isso, é preciso obter o pacote de instalação que fica

disponível em <http://www.devpaks.org> e abri-lo no Dev C++, sendo a mesma automaticamente instalada.

4. CRIAÇÃO DE UM PROGRAMA COM A ALLEGRO

A criação de um programa que utiliza a Allegro no Dev C++ é simplificada com a abertura de um novo projeto. Para isso, devem-se escolher as opções Arquivo, Novo e Projeto; depois selecionar *Multimedia, Allegro Application (static)* e a linguagem que será utilizada, no exemplo, é a linguagem C. Será necessário também selecionar uma pasta para salvar o projeto e dar um nome para o arquivo de projeto. Logo após esses procedimentos, será criado automaticamente um arquivo contendo todo código para iniciar um aplicativo em C com a Allegro.

Escolhendo as opções Executar, Compilar & Executar, do menu principal, poderá ser visto o aplicativo em execução. Note que o mesmo foi aberto em uma janela, conforme mostra a Figura 1.

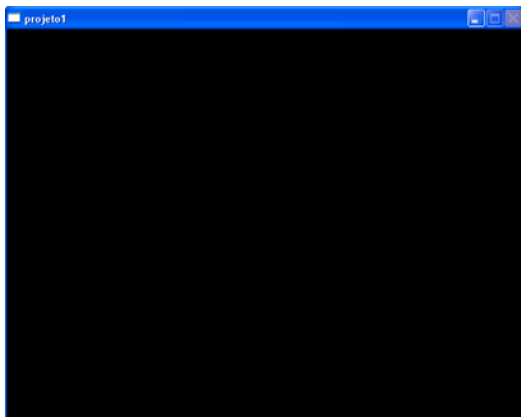


Figura 1. Janela do aplicativo em C com a Allegro

O código fonte, mostrado na Figura 2, foi criado automaticamente. Os comentários foram colocados para melhor entendimento das linhas de código.

5. USO DE IMAGENS

A carga e a utilização de arquivos de imagem na Allegro são feitas através de um ponteiro do tipo BITMAP. Para carregar a imagem do disco é utilizada a função *load_bitmap*. Depois de carregada a imagem para a memória é necessário exibi-la na tela. Para

isso, pode-se utilizar a função *draw_sprite(bitmap1, bitmap2, x, y)* que apenas desenha a imagem *bitmap2* em *bitmap1* na posição *x,y*.

```
#include <allegro.h>
/* inclui a biblioteca allegro.h */
void init();
void deinit();

int main() {
    init();

    while (!key[KEY_ESC]) {
        /* todo o jogo acontece em um laço
infinito, até que a tecla ESC seja pressionada. */
    }
    deinit();
    return 0;
}
END_OF_MAIN();

void init() {
    int depth, res;
    /* indica que será usada a allegro na aplicação */
    allegro_init();

    /* obtém a profundidade de cores usadas no PC de 8
a 32 bits */
    depth = desktop_color_depth();

    /* se depth contiver 0, define a mesma como 32 */
    if (depth == 0) depth = 32;

    /* define a quantidade de cores usadas na aplicação */
    set_color_depth(depth);

    /* seta o modo gráfico que vai ser usado */
    /* no caso,
    GFX_AUTODETECT_WINDOWED - detecta a
placa de vídeo automaticamente no modo windows
640, 480 - largura e altura da janela */
    res=
set_gfx_mode(GFX_AUTODETECT_WINDOWED,
640, 480, 0, 0);

    /* caso não consiga reconhecer a placa de vídeo
exibe uma mensagem de erro e encerra a aplicação */
    if (res != 0) {
        allegro_message(allegro_error);
        exit(-1);
    }

    install_timer();
    install_keyboard();
    install_mouse();
}
void deinit() {
    clear_keybuf();
}
```

Figura 2. Código fonte criado pelo Dev C++ com comentários

Pode parecer estranho desenhar uma imagem em outra, mas a Allegro interpreta a tela como uma imagem, que é definida como *screen*. Assim sendo, quando houver a necessidade de mostrar uma imagem na tela, basta desenhá-la na mesma.

Um detalhe interessante é que a Allegro permite que a imagem tenha fundo transparente. Para isso basta que ela possua a cor RGB (255,0,255), onde se deseja ter as áreas com transparência.

Um exemplo de como declarar, carregar e exibir uma imagem na tela é mostrado na Figura 3.

```
BITMAP *imagem;
imagem=load_bitmap("Fai.bmp",NULL);
draw_sprite(screen, imagem, 0, 0);
```

Figura 3. Exemplo de declaração e carga de um *Bitmap* com a Allegro

6. A TÉCNICA *DOUBLE BUFFERING*

A técnica de *Double Buffering* consiste em armazenar toda a tela em um ponteiro do tipo *BITMAP*, para depois escrevê-lo. Assim, caso haja a necessidade de escrever algo na tela, limpa-se o buffer com a função *clear* e inicia-se o processo de escrita novamente, sobrescrevendo a tela com o conteúdo do buffer. Essa técnica é utilizada em animações ou em trocas de desenhos na tela, para evitar que a mesma tenha variação na apresentação (WIKIPEDIA, 2006).

A Figura 4 mostra um exemplo de uma imagem que troca de posição na tela sem o uso de *Double Buffering*. Percebe-se que a tela, no segundo instante, fica sem nada desenhado, o que provoca um efeito de “pisque” da tela. A Figura 5 mostra o mesmo exemplo, porém já com o uso da técnica *Double Buffering*. Percebe-se que a tela, em nenhum instante, fica vazia, ou seja, em nenhum momento a mesma sofre o efeito de “pisca”.

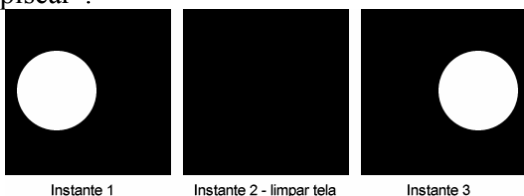


Figura 4. Exemplo de uma imagem trocando de posição, sem o uso de *Double Buffering*

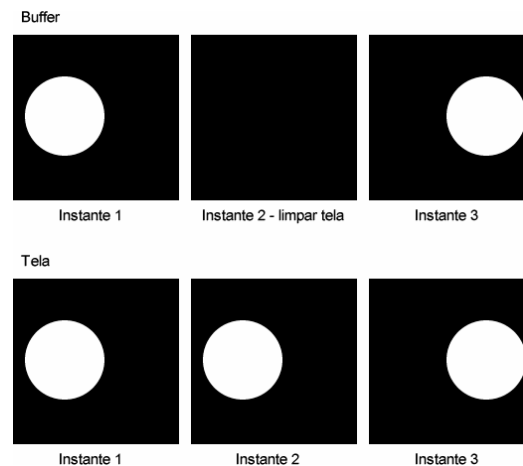


Figura 5. Exemplo de uma imagem trocando de posição, com o uso de *Double Buffering*

A Figura 6 mostra o código que realiza a movimentação de uma figura na tela, por meio das funções que implementam a técnica de *Double Buffering*.

```
BITMAP *buffer;
BITMAP *imagem;
buffer = create_bitmap(640,480);
/* usa a função create_bitmap para criar uma nova
imagem bitmap de 640 por 480 px */
imagem = load_bitmap("imagem.bmp",NULL);
int x = 0;
while(!key[KEY_ESC]){
    if (x==400){ x = 0; } /* evitar que a imagem saia
da tela */
    clear(buffer); /* limpa o buffer */
    draw_sprite(buffer, imagem, x, 0);
    draw_sprite(screen, buffer, 0,0);
    x ++;
}
```

Figura 6. Exemplo de código de uma animação com *Double Buffering*

7. USO DE FUNÇÕES DE ÁUDIO

A biblioteca Allegro permite o uso de vários formatos de sons, dentre eles, o MIDI e WAV. Para isso, deve-se inicialmente utilizar a função *install_sound* para obter a mesma configuração de som utilizada pelo computador onde o aplicativo for executado. Também, deve-se declarar um ponteiro do tipo *SAMPLE*, para o caso de um arquivo WAV ou do tipo *MIDI* para um arquivo MIDI.

Depois de declarados os ponteiros, os arquivos precisam ser carregados na memória,

com o uso das funções *load_midi* ou *load_wav*, de acordo com o tipo de arquivo.

A execução do som é feita através da função *play_midi* (*musica, loop*), que toca um arquivo MIDI uma única vez, caso o parâmetro *loop* seja *FALSE*, ou infinitas vezes caso seja *TRUE*. A função *play_sample*(*musica, vol, pan, freq, loop*) executa um arquivo WAV, com o volume determinado em *vol*, no canal *pan* ($0 \leq \text{pan} < 255$, sendo 0 canal esquerdo e 255 direito), na velocidade *freq* (sendo 1000 o valor original, caso o valor seja 2000, a música será executada com o dobro da velocidade). O parâmetro *loop* tem a mesma função já citada.

Na Figura 7 é mostrado o código fonte para carga e execução de uma música MIDI e uma música WAV.

```
MIDI *midi;
midi = load_midi("Greensleeves.mid");
play_midi(midi, TRUE);

SAMPLE *wav;
wav = load_wav("Beep.wav");
play_sample(wav, 100, 128, 1000, FALSE);
```

Figura 7. Código para execução de um MIDI e WAV

8. ENTRADA DE DADOS VIA TECLADO

A biblioteca Allegro facilitou o uso do teclado nas aplicações multimídia, sendo esse um dos meios de entrada mais utilizados em jogos.

Quando o projeto é criado, o teclado é automaticamente iniciado com a função *install_keyboard*. Depois de iniciado, para se saber se uma tecla foi pressionada, basta testar se o seu código está dentro do *array key*. A Figura 8 mostra um exemplo.

```
if (key[KEY_SPACE]){
    /* ação que será realizada caso o espaço seja
    pressionado */
}
```

Figura 8. Código para verificar se a barra de espaço foi pressionada

9. ENTRADA DE DADOS VIA MOUSE

A manipulação do mouse em Allegro pode ser feita após a iniciação com a função *install_mouse*.

Depois disso, pode-se capturar a posição do mouse na tela através das variáveis *mouse_x* e *mouse_y*. O cursor do mouse, por padrão, não fica visível na tela. Para torná-lo visível têm-se três maneiras. A primeira é através da função *show_mouse*, que exibe o cursor padrão na tela; a segunda é através da função *set_mouse_sprite*, que define um ponteiro do tipo BITMAP como sendo o cursor do mouse e a terceira e última maneira, desenhando uma imagem na posição *mouse_x* e *mouse_y*.

Para saber se os botões do mouse foram pressionados existem duas maneiras. A primeira, usando *mouse_b & 1*, para o caso do botão esquerdo pressionado, e *mouse_b & 2* para o direito.

Na Figura 9 são apresentados os dois primeiros métodos para mostrar o cursor e como capturar o clique do mouse.

```
/* primeira maneira */
show_mouse(screen);

/* segunda maneira */
BITMAP *cursor;
cursor = load_bitmap("cursor.bmp", NULL);
set_mouse_sprite(cursor);

/* verificando se o mouse foi pressionado */
if (mouse_b & 1){
    /* ação para quando o cursor esquerdo foi acionado */
}
```

Figura 9. Exemplos de uso de funções do mouse

10. USO DE TEMPORIZADORES

Um recurso bastante interessante oferecido pela biblioteca Allegro é a possibilidade de adicionar temporizadores (*timers*) nas aplicações, de forma que uma função possa ser executada em intervalos de tempo definidos. Esse recurso permite criar vários itens em um jogo, como marcadores de tempo e animações, de forma independente da velocidade do computador. Para utilizar um temporizador, é necessário iniciá-lo com a função *install_timer*.

A Figura 10 exemplifica o uso de temporizadores e suas particularidades em um código comentado.

```
volatile int tempo = 0; /* declaração de uma variável
inteira tempo */
LOCK_VARIABLE(tempo); /* toda variável global
usada em uma função manipulada pelo Trimer deve ser
travada */
void relógio(){
    tempo++;
    /* toda função manipulada por Timers, não deve
retornar nenhum valor e não pode manipular a tela.
Além de ter que executar instruções simples para não
sobrecarregar o processamento da máquina e porque
timers também são usados em funções de música, por
exemplo. */
}
END_OF_RELOGIO(); /* indica o fim da função */

/* quando quiser que a função comece a ser executada
de tempos em tempos */
install_int(relógio,1000); /* executará a função relógio
de 1000 em 1000 milisegundos. A cada intervalo de
tempo, a variável tempo será incrementada em 1 */
```

Figura 10. Exemplo de uso de temporizadores

11. CONCLUSÃO

Procurou-se, neste artigo, mostrar as funções básicas da biblioteca Allegro que permitem a utilização de recursos de multimídia.

O desenvolvimento de jogos e aplicações multimídia é um trabalho complexo em linguagens como C e C++. Levando isso em consideração, a biblioteca Allegro facilita o processo de adicionar recursos como imagens, sons e temporizadores, que são de suma importância para esses aplicativos, permitindo que se diminua o seu tempo de desenvolvimento.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLEGRO. **Official Site**. 2005. Disponível em: <http://www.allegro.cc>. Acesso em: 5 nov. 2005

WIKIPEDIA. **Double Buffering**. 2006. Disponível em: http://en.wikipedia.org/wiki/Double_buffering. Acesso em: 5 out. 2006.