A GAME PROGRAMMING LIBRARY
"Allegro Low Level Game Routines"

> *Allegro 4.1* // *WinAllegro* (DirectX) // *XWinAllegro*
> **Compilador: DJGPP** - GNU C/C++ for PC's
> **Sistema Operacional: DOS**, Windows e Linux

WEB: http://www.talula.demon.uk/allegro/

ALLEGRO - Conceitos Básicos:

- *Conjunto de rotinas para acesso ao dispositivos periféricos:* mouse, timer, teclado, joystick, placas gráficas (drivers, palette, funções gráficas bit-map/2D/3D/animação), sons, músicas, compressão de armazenamento de dados, funções matemáticas e GUI (Graphical User Interface).

- *Programação em C / C++ usando o compilador GNU para PC*
- *Library em C e Assembly* a ser linkada ao programa do usuário
- *Orientada ao desenvolvimento de Jogos*

- *Vantagens:*
  * Facilidade de uso e desenvolvimento de aplicações
  * Portabilidade: ao contrário do DirectX, não está restrita apenas a plataforma PC/Windows (roda em DOS, Windows e Linux)

- *Concorrentes diretos:* DirectX, OpenGL e Java (Java3D / JMF)

ALLEGRO - Referências Básicas:

WEB:

Site Oficial	http://www.talula.demon.uk/allegro
Allegro Vivace	http://www.canvaslink.com/gfoot/vivace
	Tutorial: criação de jogos usando C (DJGPP + Allegro) para DOS/Win95
WinAllegro	http://www.casema.net/~berryfra/winallg.htm
	http://www.canvaslink.com/gfoot/winallegro/
XWinAllegro	http://www.canvaslink.com/allegro/xwinallegro/
Allegro Games	http://agd.emuware.com/
	http://www.dtek.chalmers.se/~d98peitz/fld/games.html

How to Program Games - http://www.canvaslink.com/gfoot/htpg/

Allegro na Unisinos:

Drive G: (Aplica) => G:\Djgpp\Allegro [Inclui Exemplos]

ALLEGRO - Usando a library:

=> Incluir no programa:
include <allegro.h> { Adicionar no path dos includes }

=> Compilar com um comando do tipo:
gcc foo.c -o foo.exe -lalleg
- Arquivo da library: liballeg.a
- Adicionar o path para o diretório onde se encontra a lib.

=> É possivel selecionar somente alguns dos módulos do Allegro.
Exemplo - #define `alleg_mouse_unused 1`

```
=> Possibilidade de usar o ambiente integrado
   de desenvolvimento de programas "RHIDE"
   (Interface idêntica a do Turbo Pascal)
```

ALLEGRO - Funções da library:

=> Grupos de rotinas:

- Init
- Configure
- Mouse
- Timer
- Keyboard
- Joystick
- Graphics Modes
- Bitmap Objects
- Load Image Files (BMP,PCX,TGA)
- Truecolor Pixel
- Blitting and Sprites
- Palette Routines
- File Compression (LZSS)
- Datafile Routines (Grab:*.dat)
- Math Routines
- 3D Math Routines
- Drawing Primitives
- RLE Sprites
- Compiles Sprites
- Text Output
- Polygon Rendering
- Transparency and Patterns
- Color Conversion
- Video Memory Access
- FLI/FLC Routines
- Sound Init
- Digital Sounds (Wav, Voc)
- Music Sounds (Midi)
- Audio Stream (big files)
- Recording Sounds
- GUI Routines
- Debugging

```
/*
 *    Example program for the Allegro library, by Shawn Hargreaves.
 *
 *    This is a very simple program showing how to get into graphics
 *    mode and draw text onto the screen.
 */


#include <stdlib.h>
#include <stdio.h>

#include "allegro.h"


int main()
{
  /* you should always do this at the start of Allegro programs */
  allegro_init();

  /* set up the keyboard handler */
  install_keyboard();

  /* set VGA graphics mode 13h (sized 320x200) */
  set_gfx_mode(GFX_VGA, 320, 200, 0, 0);

  /* set the color pallete */
  set_pallete(desktop_pallete);

  /* write some text to the screen */
  textout_centre(screen, font, "Hello, world!", SCREEN_W/2, SCREEN_H/2, 255);

  /* wait for a keypress */
  readkey();

  return 0;
}
```

```
/*
 *    Example program for the Allegro library, by Shawn Hargreaves.
 *
 *    This program demonstrates the use of memory bitmaps. It creates
 *    a small temporary bitmap in memory, draws some circles onto it,
 *    and then blits lots of copies of it onto the screen.
 */


#include <stdlib.h>
#include <stdio.h>

#include "allegro.h"


int main()
{
   BITMAP *memory_bitmap;
   int x, y;

   allegro_init();
   install_keyboard();
   set_gfx_mode(GFX_VGA, 320, 200, 0, 0);
   set_pallete(desktop_pallete);

   /* make a memory bitmap sized 20x20 */
   memory_bitmap = create_bitmap(20, 20);


}
```

```
/* draw some circles onto it */
  clear(memory_bitmap);
  for (x=0; x<16; x++)
    circle(memory_bitmap, 10, 10, x, x);

  /* blit lots of copies of it onto the screen */
  for (y=0; y<SCREEN_H; y+=20)
    for (x=0; x<SCREEN_W; x+=20)
                blit(memory_bitmap, screen, 0, 0, x, y, 20, 20);

  /* free the memory bitmap */
  destroy_bitmap(memory_bitmap);

  readkey();
  return 0;
```

```
/*
 *    Example program for the Allegro library, by Shawn Hargreaves.
 *
 *    This program demonstrates how to get mouse input.
 */


#include <stdlib.h>
#include <stdio.h>

#include "allegro.h"


int main()
{
   int mickeyx = 0;
   int mickeyy = 0;
   BITMAP *custom_cursor;
   char msg[80];
   int c = 0;

   allegro_init();
   install_keyboard();
   install_mouse();
   install_timer();
   set_gfx_mode(GFX_VGA, 320, 200, 0, 0);
   set_pallete(desktop_pallete);

   do {
      /* the mouse position is stored in the variables mouse_x and mouse_y */
      sprintf(msg, "mouse_x = %-5d", mouse_x);
      textout(screen, font, msg, 16, 16, 255);

      sprintf(msg, "mouse_y = %-5d", mouse_y);
      textout(screen, font, msg, 16, 32, 255);

/* or you can use this function to measure the speed of movement.
       * Note that we only call it every fourth time round the loop:
       * there's no need for that other than to slow the numbers down
       * a bit so that you will have time to read them...
       */
      c++;
      if ((c & 3) == 0)
                  get_mouse_mickeys(&mickeyx, &mickeyy);
```

```
      sprintf(msg, "mickey_x = %-7d", mickeyx);
      textout(screen, font, msg, 16, 64, 255);

      sprintf(msg, "mickey_y = %-7d", mickeyy);
      textout(screen, font, msg, 16, 80, 255);

      /* the mouse button state is stored in the variable mouse_b */
      if (mouse_b & 1)
                  textout(screen, font, "left button is pressed ", 16, 112, 255);
      else
                  textout(screen, font, "left button not pressed", 16, 112, 255);

      if (mouse_b & 2)
                  textout(screen, font, "right button is pressed ", 16, 128, 255);
      else
                  textout(screen, font, "right button not pressed", 16, 128, 255);

      if (mouse_b & 4)
                  textout(screen, font, "middle button is pressed ", 16, 144, 255);
      else
                  textout(screen, font, "middle button not pressed", 16, 144, 255);

      vsync();

   } while (!keypressed());

   clear_keybuf();

   /*  To display a mouse pointer, call show_mouse(). There are several
    *  things you should be aware of before you do this, though. For one,
    *  it won't work unless you call install_timer() first. For another,
    *  you must never draw anything onto the screen while the mouse
    *  pointer is visible. So before you draw anything, be sure to turn
    *  the mouse off with show_mouse(NULL), and turn it back on again when
    *  you are done.
    */
   clear(screen);
   textout_centre(screen, font, "Press a key to change cursor", SCREEN_W/2,
SCREEN_H/2, 255);
   show_mouse(screen);
   readkey();
   show_mouse(NULL);
```

```
/* create a custom mouse cursor bitmap... */
   custom_cursor = create_bitmap(32, 32);
   clear(custom_cursor);
   for (c=0; c<8; c++)
      circle(custom_cursor, 16, 16, c*2, c);

   /* select the custom cursor and set the focus point to the middle of it */
   set_mouse_sprite(custom_cursor);
   set_mouse_sprite_focus(16, 16);

   clear(screen);
   textout_centre(screen, font, "Press a key to quit", SCREEN_W/2, SCREEN_H/2, 255);
   show_mouse(screen);
   readkey();
   show_mouse(NULL);

   destroy_bitmap(custom_cursor);

   return 0;
}
```

```
/*
 *    Example program for the Allegro library, by Shawn Hargreaves.
 *    This program demonstrates how to load and display a bitmap file.
 */
#include <stdlib.h>
#include <stdio.h>

#include "allegro.h"

int main(int argc, char *argv[])
{
   BITMAP *the_image;
   PALLETE the_pallete;

   if (argc != 2) {
      printf("Usage: 'ex15 filename.[bmp|lbm|pcx|tga]'\n");
      return 1;
   }

   /* read in the bitmap file */
   the_image = load_bitmap(argv[1], the_pallete);
   if (!the_image) {
      printf("Error reading bitmap file '%s'\n", argv[1]);
      return 1;
   }

   allegro_init();
   install_keyboard();
   set_gfx_mode(GFX_VGA, 320, 200, 0, 0);

   /* select the bitmap pallete */
   set_pallete(the_pallete);

   /* blit the image onto the screen */
   blit(the_image, screen, 0, 0, (SCREEN_W-the_image->w)/2,
                                 (SCREEN_H-the_image->h)/2, the_image->w, the_image->h);

   /* destroy the bitmap */
   destroy_bitmap(the_image);

   readkey();
   return 0;
}
```

```
/*
 *    Example program for the Allegro library, by Shawn Hargreaves.
 *
 *    This program demonstrates how to play MIDI files.
 */


#include <stdlib.h>
#include <stdio.h>

#include "allegro.h"

int main(int argc, char *argv[])
{
   MIDI *the_music;

   if (argc != 2) {
      printf("Usage: 'ex16 filename.mid'\n");
      return 1;
   }

   allegro_init();
   install_keyboard();
   install_timer();

   /* install a MIDI sound driver */
   if (install_sound(DIGI_AUTODETECT, MIDI_AUTODETECT, argv[0]) != 0) {
      printf("Error initialising sound system\n%s\n", allegro_error);
      return 1;
   }

   /* read in the MIDI file */
   the_music = load_midi(argv[1]);
   if (!the_music) {
      printf("Error reading MIDI file '%s'\n", argv[1]);
      return 1;
   }

   printf("Midi driver: %s\n", midi_driver->name);
   printf("Playing %s\n", argv[1]);

   /* start up the MIDI file */
   play_midi(the_music, TRUE);

   /* wait for a keypress */
   readkey();

   /* destroy the MIDI file */
   destroy_midi(the_music);

   return 0;
}
```