Introdução à SDL

por Bruno Bottino Ferreira
tinnus@gmail.com
Adriano Cruz
adriano@nce.ufrj.br

Agenda

- Por que?
- Instalando
- Conceitos Básicos
- Programando

Por que?

Por que SDL?

- Abstração do hardware

Programador

SDL
Simple Directmedia Layer

Por que SDL?

- Portabilidade

O que é SDL?

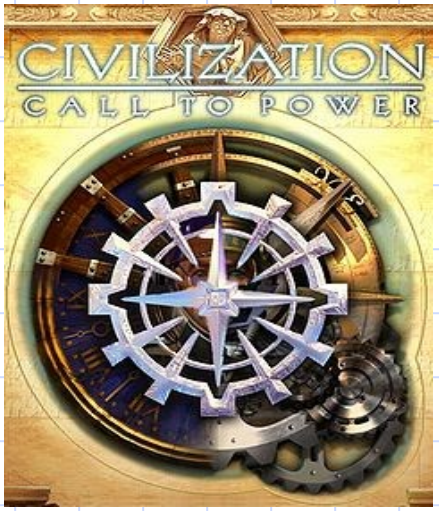
- Simple Directmedia Layer
- Biblioteca multimídia voltada para jogos
- Vídeo
- Som
- Interface com o usuário (teclado, mouse, joystick)
- CD-Áudio
- Threading
- Controle de tempo

Exemplos


QUAKE 4


DOOM 3
"A MASTERPIECE OF THE ART FORM"


H-CRAFT CHAMPIONSHIP
SCIFI RACING SIMULATION


CIVILIZATION
CALL TO POWER


UNREAL
TOURNAMENT


SIM CITY
3000

Requisitos

- Conhecimento de programação em C/C++
- Ambiente de programação C/C++
- Instalar a biblioteca no ambiente de programação
- Preferencialmente hardware de som, mouse, joystick e CD-ROM (para testes)

Instalando

Instalando

- http://www.libsdl.org
- Download -> SDL x.x (mais recente)
- Development Libraries
  (para sua plataforma)

  ou
- Source code (código-fonte)
  (deve ser compilado)

Instalando no MinGW

- Baixar o arquivo .tar.gz correspondente ao MinGW
- Copiar as pastas “Include” e “Lib” para a pasta do MinGW
- Para compilar:

```
gcc prog.c –o prog.exe -Wall
–lmingw32 –lSDLmain -lSDL
```

Subdivisões

- SDL é composta de várias sub-bibliotecas.
- SDL_image trata de imagens
- SDL_mixer trata de sons
- SDL_ttf gerencia fontes
- etc

MinGW – SDL_image

- Baixar o arquivo SDL_image-devel-X.X.X-XX.zip de http://www.libsdl.org/ correspondente ao MinGW windows32
- Copiar as pastas “Include” e “Lib” para a pasta do MinGW
- Para compilar:

```
gcc prog.c –o prog.exe -Wall –lmingw32 –lSDLmain –lSDL –lSDL_image
```

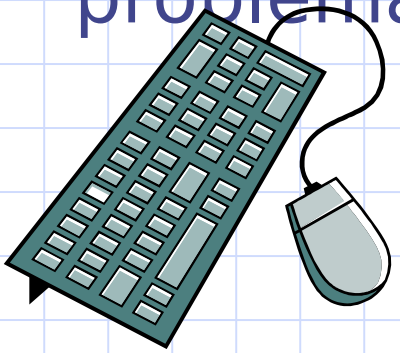
MinGW – SDL_mixer

- Baixar o arquivo SDL_mixer-devel-X.X.X-XX.zip de http://www.libsdl.org/ correspondente ao MinGW windows32
- Copiar as pastas “Include” e “Lib” para a pasta do MinGW
- Para compilar:

```
gcc prog.c –o prog.exe -Wall –lmingw32 –lSDLmain –lSDL –lSDL_image –lSDL_mixer
```

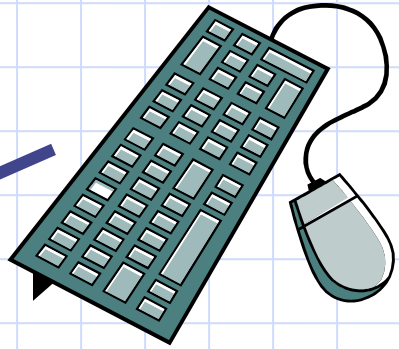
Observações

- Cuidados no Windows
  - A SDL redireciona `stdout` e `stderr` para os arquivos `stdout.txt` e `stderr.txt`
  - Tentar ler de `stdin` geralmente causa problemas.



Conceitos Básicos

Observações



Prog C ⟷ SDL Simple Directmedia Layer

Conceitos básicos

- A função da API SDL de vídeo é simplesmente encontrar e configurar um dispositivo para você usar.
- Uma vez iniciado o display, criada a janela ou colocada a tela em um determinado modo, SDL deve sair do caminho.
- SDL fornece um conjunto mínimo de funções para mover blocos de pixels, gerenciar o teclado, mouse e só.
- SDL não é uma ferramenta de desenho.
- O que você faz depois que ocorre a inicialização é problema seu.

Conceitos básicos

- SDL trabalha baseado em eventos.
- Normalmente o programa *não controla* o seu fluxo.
- Ele fica em um laço esperando eventos acontecerem.
- O programa deve reagir a eventos.
- Um *evento* é uma mensagem enviada do sistema operacional ao seu programa.
- Tipos de eventos: tecla apertada, mouse andou, termina etc.

Conceitos básicos

- SDL especifica tipos quando tamanho dos tipos é importante
- Unsigned: `Uint32, Uint16, Uint8`
- Signed: `Sint32, Sint16, Sint8`
- Algumas vezes: `Uint64, Sint64` (somente se `SDL_HAS_64BIT_TYPE` for definido)
- Também `SDL_Bool` (assume valores `SDL_FALSE` ou `SDL_TRUE`)

Programa básico em SDL

- Loop principal
  -
```
while(!stop) {
    while(SDL_PollEvent(&event)) {
      switch (event.type){
        case SDL_QUIT: stop = 1; break;
        case SDL_KEYDOWN: /*Trata */
        /* case outros eventos */
      }
    }
    /* move imagens */
    SDL_Fip(screen); /*atualiza tela */
  }
```

Superfícies

- SDL usa estruturas chamadas superfícies (do tipo **SDL_Surface**) para representar dados gráficos.
- Uma superfície é somente um bloco de memória para armazenar uma região retangular de pixels.
- Cada superfície tem uma largura, altura e um formato específico para armazenar os pixels.

Conceitos básicos

- Uma *superfície* é uma matriz bidimensional de pontos, onde cada ponto representa uma cor composta por três componentes: R, G e B (vermelho, verde e azul).
- Um *PixelFormat* é uma estrutura que define o formato em que as cores dos pixels são armazenadas.

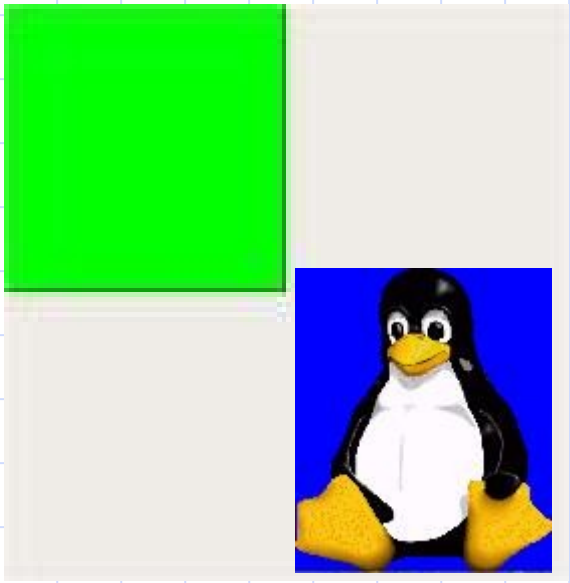
Superfícies

- SDL carrega arquivos de imagens diretamente nas estruturas de superficie.
- A tela (screen) também é uma superfície, embora especial.
- Superfícies podem ser copiadas para cima de outras em uma operação chamada blit (block image transfer).

Superfícies

- Blits são parte fundamental da programação de jogos.
- Com blits é possível fazer imagens a partir de outras imagens pré-desenhadas.
- Desde que a tela é uma superfície como qualquer outra, imagens podem ser copiadas para a tela (screen) com uma única operação de blit.
- A função de blit é muito rápida.

Superfícies



Programando

Programa básico em SDL

- Incluir a biblioteca
  - `#include <SDL.h>`
- Declarar a função **main** como
  - `int main (int argc, char** argv)`
- Declarar variáveis de controle
  - ```
    SDL_Surface* screen;
    SDL_Event event;
    int stop = 0;
    ```
    ```

Programa básico em SDL

- Inicializar
  - `SDL_Init(flags);`
- “flags” é um ou-binário dos valores:
  - `SDL_INIT_VIDEO`
  - `SDL_INIT_AUDIO`
  - `SDL_INIT_JOYSTICK`
  - `SDL_INIT_CDROM`
  - `SDL_INIT_TIMER`
  - `SDL_INIT_EVERYTHING`

```
SDL_Init(SDL_INIT_VIDEO|SDL_INIT_AUDIO);
```

Programa básico em SDL

- Criar a janela
  - `screen = SDL_SetVideoMode(width, height, depth, flags);`
- “flags” é um ou-binário dos valores:
  - `SDL_SWSURFACE`
  - `SDL_HWSURFACE`
  - `SDL_ASYNCBLIT`
  - `SDL_ANYFORMAT`
  - `SDL_HWPALETTE`
  - `SDL_DOUBLEBUF`
  - `SDL_FULLSCREEN`
  - `SDL_OPENGL`
  - `SDL_OPENGLBLIT`

Programa básico em SDL

- Loop principal

```
while(!stop)
{
  while(SDL_PollEvent(&event))
  {
    if(event.type == SDL_QUIT)
    {
      stop = 1;
    }
  }
  SDL_Flip(screen);
}
```

Programa básico em SDL

- Para encerrar, escolher

```
/* colocar no final */
  SDL_Quit();
  return 0;
```

  - `/* ou colocar no inicio, garante que SDL_Quit será executado */`

```
   atexit(SDL_Quit);
```

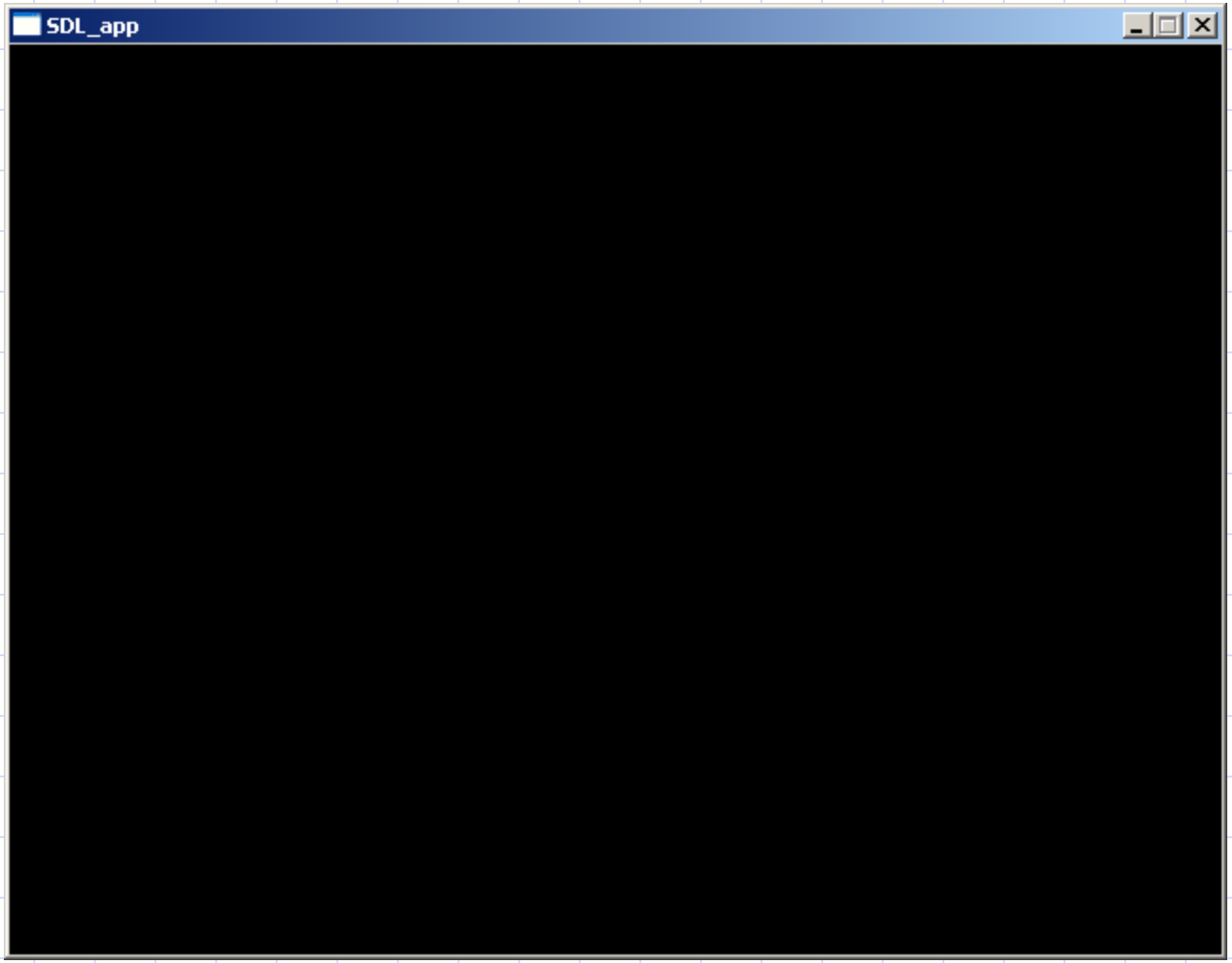
Programa básico em SDL

```
#include<SDL.h>
/* #include <SDL/SDL.h> se manteve a pasta SDL */
#include<stdlib.h>
int main(int argc, char** argv) {
    SDL_Surface* screen;
    SDL_Event event;
    int quit = 0;
    atexit(SDL_Quit);

    SDL_Init(SDL_INIT_VIDEO);
    screen = SDL_SetVideoMode(640, 480, 8, 0);

    while(!quit) {
        while(SDL_PollEvent(&event))
           if(event.type == SDL_QUIT)
                quit = 1;
        SDL_Flip(screen);
    }
    SDL_Quit();
    return 0;
}
```

Programa básico em SDL



Superfícies

- Uma superfície com o PixelFormat da tela
  - ```
    my_surf =  SDL_CreateRGBSurface(flags,
                width, height,
     screen->format->BitsPerPixel,
     screen->format->Rmask,
    screen->format->Gmask,
     screen->format->Bmask,
     screen->format->Amask);
    ```
    ```

Superfícies

- Criando uma superfície a partir de um arquivo
  - BMP
    - `my_surf = SDL_LoadBMP("file.bmp");`
  - BMP, PNM, XPM, LBM, PCX, GIF, JPEG, PNG, TGA, TIFF (usando a biblioteca SDL_image)
    -
```
#include <SDL_image.h>

/* ... */

my_surf = IMG_Load("file.xxx");
```
  - http://www.libsdl.org/projects/SDL_image/

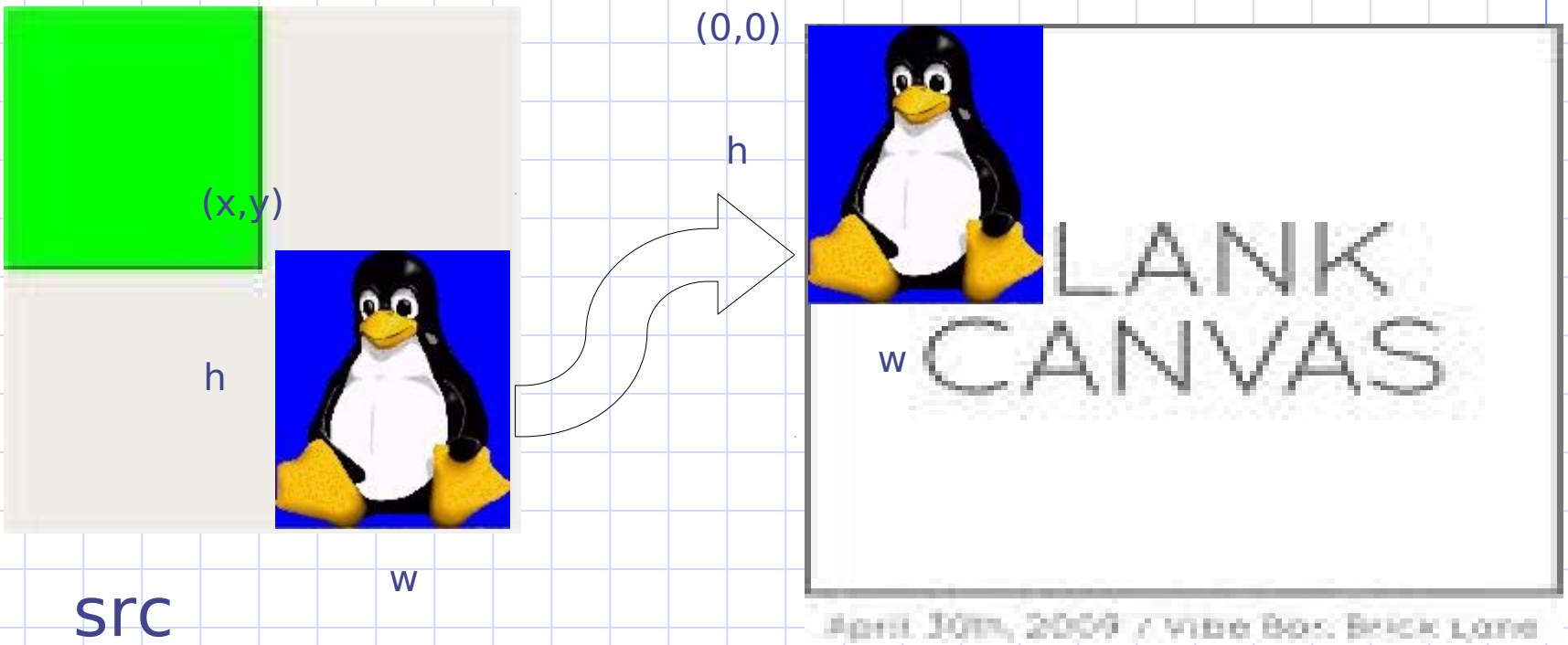
Blits

- Um *blit* é uma cópia de parte da imagem de uma superfície para outra
  - `src` é a superfície de origem
  - `dst` é a superfície de destino
  - `srcrect` é a área de origem
  - `dstrect` é a área de destino
  - Se `srcrect` ou `dstrect` forem NULL, significa toda a área da superfície
- `SDL_BlitSurface(src, srcrect, dst, dstrect);`
- Arquivo: `ex1.c`

Superfícies

srcrect={x,y,w,h};

dstrect={0,0,w,h};



src

dst

- Arquivo: ex1.c

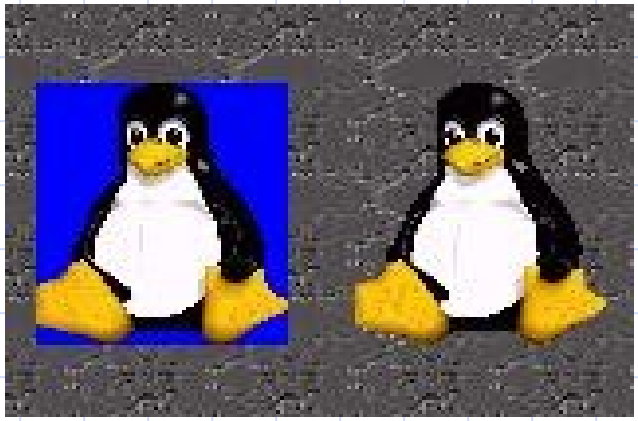
Atualizando a tela

- Para atualizar parte da tela
  - `SDL_UpdateRect(screen, x, y, width, height);`
- Para atualizar toda a tela
  - Sem double buffering (`SDL_DOUBLEBUF`)
    - `SDL_UpdateRect(screen, 0, 0, screen->w, screen->h);`
  - Com ou sem double buffering
    - `SDL_Flip();`

Color-Key

- As vezes é necessário tornar uma cor transparente (color-key).

Original



Com color-key

Color-key

- Tornar uma cor transparente
  - `SDL_SetColorKey(surface, SDL_SRCCOLORKEY, SDL_MapRGB(surface->format, r, g, b));`
- Desligar a cor transparente
  - `SDL_SetColorKey(surface, 0, 0);`

Transparência-alpha

- Em outras situações toda uma imagem precisa se tornar transparente.
- Neste caso podemos ter imagens a frente de outras.
- Exemplo: um fantasma andando por um ambiente.

Transparência

- Dar um valor de transparência geral
  - `SDL_SetAlpha(surface,SDL_SRCALPHA, val);`
  - `val`: 0 (transparente) a 255 (opaco)
- Desligar a transparência geral
  - `SDL_SetAlpha(surface, 0, 0);`
- Arquivos: `ex2xx.c`

Cursor do Mouse

- Mostrar/esconder o cursor
  - `SDL_ShowCursor(toggle);`
    - SDL_ENABLE mostra o cursor
    - SDL_DISABLE esconde o cursor
    - SDL_QUERY retorna o estado atual
- Mover o cursor
  - `SDL_WarpMouse(x, y);`

Eventos

- Tipos de eventos
  - SDL_ACTIVEEVENT
  - SDL_KEYDOWN / SDL_KEYUP
  - SDL_MOUSEMOTION
  - SDL_MOUSEBUTTONDOWN / SDL_MOUSEBUTTONUP
  - SDL_JOYAXISMOTION
  - SDL_JOYBALLMOTION
  - SDL_JOYHATMOTION
  - SDL_JOYBUTTONDOWN / SDL_JOYBUTTONUP
  - SDL_QUIT
  - SDL_SYSWMEVENT
  - SDL_VIDEORESIZE
  - SDL_VIDEOEXPOSE
  - SDL_USEREVENT

Eventos

- Lendo eventos
  - SDL_Event event;

```
while(SDL_PollEvent(&event))
{
  switch(event.type)
  {
    case tipo1: /* ... */ break;
    case tipo2: /* ... */ break;
    /* ... */
    case tipon: /* ... */ break;
  }
}
```

Eventos do Teclado

- Teclado
  - Dados do evento: `event.key`
    - `type`: `SDL_KEYDOWN` ou `SDL_KEYUP`
    - `state`: `SDL_PRESSED` ou `SDL_RELEASED`
  - Dados da tecla: `event.key.keysym`
    - `sym`: constante em `SDLKey`
    - `unicode`: caracter (formato Unicode/ASCII)
    - `mod`: modificadores (ou-binário de constantes em SDLMod)
- Arquivo: `ex3.c`

Eventos do Mouse

- Movimentação do mouse
  - TIPO: `if (event.type==SDL_MOUSEMOTION)`
  - Dados do evento em `event.motion`
  - `event.motion.x, event.motion.y` são as novas coordenadas do mouse
  - `event.motion.xrel,event.motion.yrel` é o deslocamento relativo a última posição
  - `event.motion.state` pode ser igual a `SDL_PRESSED, SDL_RELEASED`

Eventos do Mouse

- Clique do mouse
  - TIPO: `if (event.type == SDL_MOUSEBUTTONDOWN)`
  - Dados do evento em `event.button`
  - `event.button.button` pode ser igual a:
    - SDL_BUTTON_LEFT
    - SDL_BUTTON_MIDDLE
    - SDL_BUTTON_RIGHT
  - `event.button.state` pode ser igual a:
    - SDL_PRESSED
    - SDL_RELEASED
- Arquivo: `ex4.c`

Eventos

- Outros eventos
  - Joysticks
  - Redimensionamento da janela
  - Encerramento
  - Eventos gerados pelo usuário

CD-Áudio

- Número de drives de CD
  - `n_drives = SDL_CDNumDrives();`
- Abrir um drive para acesso
  - `SDL_CD* cdrom = SDL_CDOpen(n);`
- Tocar uma trilha
  - `SDL_CDPlayTracks(cdrom, track, 0, 1, 0);`
  - Trilhas começam em zero
- Fechar um drive para acesso
  - `SDL_CDClose(cdrom);`
- Arquivo: `ex5.c`

Som

- Utilizando a biblioteca SDL_mixer
  - http://www.libsdl.org/projects/SDL_mixer/
- Inicializando
  - `SDL_Init( ... | SDL_INIT_AUDIO);`
  - `Mix_OpenAudio(freq, format, channels, bufsize);`
    - `freq`: frequência de amostragem (ex. 44100Hz)
    - `format`: formato de saída (ex. 16bits)
    - `channels`: 1 (mono) ou 2 (estéreo)
    - `bufsize`: tamanho do buffer (ex. 4096)
  - Tipicamente
    - `Mix_OpenAudio(MIX_DEFAULT_FREQUENCY, MIX_DEFAULT_FORMAT, 2,4096);`

Som

- Carregando um som
  - `Mix_Chunk* sound = Mix_LoadWAV("kaboom.wav");`
- Ajustando o volume de um som
  - `Mix_VolumeChunk(sound, volume);`
  - `0 <= volume <= MIX_MAX_VOLUME`
- Liberando a memória
  - `Mix_FreeChunk(sound);`

Som

- Alocando canais de saída
  - `Mix_AllocateChannels(n_channels);`
- Tocando um som em um canal
  - `Mix_PlayChannel(channel, sound, loops);`
  - `channel`: número do canal
    - `channel = -1`: primeiro disponível
  - `loops`: número de repetições
    - `loops = 0`: tocar uma vez
    - `loops = -1`: tocar para sempre

Som

- Carregando uma música
  - `Mix_Music* music = Mix_LoadMUS("boogie.mp3");`
  - WAVE, MOD, MIDI, OGG, MP3
- Tocando uma música
  - `Mix_PlayMusic(music, loops);`
  - `loops`: número de repetições
    - `loops = 0`: tocar uma vez
    - `loops = -1`: tocar para sempre
- Ajustando o volume da música
  - `Mix_VolumeMusic(sound, volume);`
  - `0 <= volume <= MIX_MAX_VOLUME`

Som

- Encerrando
  - `Mix_CloseAudio();`
- Outras funções
  - Fade in/out
  - Panning
  - Distância
  - Posição (som 2D)
- Arquivo: `ex6.c`

Tempo

- Contagem de tempo
  - `time = SDL_GetTicks();`
  - Retorna o número de ms desde a inicialização
- Criando um temporizador
  - `id = SDL_AddTimer(interval, func, param);`
  - Chama a função `func` a cada `interval` ms passando os parâmetros `interval` e `param`
  - `Uint32 func(Uint32 interval, void *param);`
- Cancelando um temporizador
  - `SDL_RemoveTimer(id);`
- Arquivo: `ex7.c`

Encerrando

- Assuntos não abordados
  - Joysticks
  - Threads
- Onde conseguir ajuda
  - http://www.libsdl.org
    - http://www.libsdl.org/cgi/docwiki.cgi/
    - http://news.gmane.org/thread.php?group=gmane.comp.lib.sdl
  - http://gpwiki.org/index.php/C:SDL_tutorials

O Fim