Introdução à SDL

Prof. MSc. João Ricardo Bittencourt

UNISINOS

Sumário

1. Outros formatos de imagens (PNG,JPEG,...)
2. Texto (Fontes TTF)
3. Eventos
4. Sincronismo
5. SDL & OpenGL
6. Som
7. Rede

Outros formatos de imagens

- Licença LGPL
- Suporta os seguintes formatos: **BMP**, PNM(PPM/PGM/PBM),XPM,LBM,**PCX**,**GIF**,**JPEG**,**PNG**,TGA e **TIFF**
- JPEG, PNG e TIFF requerem bibliotecas adicionais
- Para usá-la deve-se adicionar
  - `#include “SDL_image.h”`
- Download em:
  - http://www.libsdl.org/projects/SDL_image/

Outros formatos de imagens

- Para carregar uma imagem
  - `IMG_Load("nome_da_imagem.png")`
    - Use sempre a extensão
    - Em caso de erro retorna NULL
    - Em caso afirmativo retorna uma SDL_Surface *
- Verifica se ocorreu algum erro
  - `IMG_GetError()`
    - Retorna uma mensagem de erro

Textos (Fontes TTF)

- Licença LGPL
- Usar fontes True Type com alta qualidade inclusive com *antialiasing*
- Para usá-la deve-se adicionar
  - `#include “SDL_ttf.h”`
- Download em:
  - http://www.libsdl.org/projects/SDL_ttf/

Textos (Fontes TTF)

- Inicializa a API para tratar as fontes True Type
  - `TTF_Init()`
    - Em caso de erro retorna -1
    - Em caso afirmativo retorna 0
- Verifica se ocorreu algum erro
  - `TTF_GetError()`
    - Retorna uma mensagem de erro
- Encerra a execução da API
  - `TTF_Quit()`

Textos (Fontes TTF)

- Abrir uma fonte
  - `TTF_OpenFont("fonte.ttf",16)`
    - Em caso de erro retorna NULL
    - Em caso afirmativo retorna TTF_Font *
- Libera a memória usada pela fonte
  - `TTF_CloseFont(font)`
- Definir o estilo da fonte
  - `TTF_SetFontStyle(font,TTF_STYLE_BOLD|TTF_STYLE_ITALIC)`
    - Ainda tem a opção `TTF_STYLE_UNDERLINE` e `TTF_STYLE_NORMAL`

Textos (Fontes TTF)

- Criar um texto a partir da fonte
  - `TTF_RenderText_Solid(font,”Hello”,cor)`
    - Em caso de erro retorna NULL
    - Em caso afirmativo retorna SDL_Surface *
- Sendo que a cor é definida da seguinte forma:
  - `SDL_Color cor = {0,0,0}`
- Pode usar o modo Latin1, Unicode ou UTF-8. E tem três formas de renderização:
  - *Solid*: rápido, sem uma caixa, sem suavidade
  - *Shaded*: lento, com uma caixa, antialiasing
  - *Blended*: muito lento, sem caixa, antialiasing

Textos (Fontes TTF)

- **Outras formas de renderização**
  - `TTF_RenderText_Solid(font,”Hello”,cor)`
  - `TTF_RenderUTF8_Solid(font,”Hello”,cor)`
  - `TTF_RenderUNICODE_Solid(font,”Hello”,cor)`
  - `TTF_RenderText_Shaded(font,”Hello”,cor)`
  - `TTF_RenderUTF8_Shaded(font,”Hello”,cor)`
  - `TTF_RenderUNICODE_Shaded(font,”Hello”,cor)`
  - `TTF_RenderText_Blended(font,”Hello”,cor)`
  - `TTF_RenderUTF8_Blended(font,”Hello”,cor)`
  - `TTF_RenderUNICODE_Blended(font,”Hello”,cor)`

Eventos

- Tratamento geral de eventos:

```
SDL_Event event;
  while (SDL_PollEvent(&event)){
   //Trata o evento recebido
   //Usa event.type
  }
```

- Após o evento ter sido capturado é tratado conforme seu tipo.
  - Básicos: Mouse, teclado, vídeo e eventos do usuário

Eventos

- Alguns tipos de eventos:
  - `SDL_MOUSEMOTION` : Mouse é movimentado
  - `SDL_MOUSEBUTTONDOWN` :Um botão do mouse é pressionado
  - `SDL_MOUSEBUTTONUP` : Um botão do mouse após ser pressionado é solto
  - `SDL_KEYDOWN` : Uma tecla é pressionada
  - `SDL_KEYUP` : Uma tecla após ser pressionada é solta
  - `SDL_VIDEORESIZE` : A janela é redimensionada

Eventos

- Alguns tipos de eventos:
  - `SDL_VIDEOEXPOSE` : A janela precisa ser redesenhada
  - `SDL_QUIT` : Um evento para encerrar a aplicação
  - `SDL_USEREVENT` : Qualquer evento definido pelo usuário
- Detalhes para tratamento de eventos consulte a API da SDL
  - http://www.libsdl.org/cgi/docwiki.cgi/SDL_20API

Sincronismo

- Fundamental para manter o *game loop*
- Retorna o tempo em milissegundos desde a inicialização da SDL
  - `SDL_GetTicks()`
- Para a execução por um determinado tempo X em milissegundos
  - `SDL_Delay(X)`

Sincronismo

- Adiciona um temporizador que a cada intervalo de tempo T executa uma função de *callback*
  - `SDL_AddTimer(T,callback, paramCallBack)`
    - Retorna um identificador SDL_TimerID
    - A assinatura da callback deve ser:
      - `Uint32 (*callback)(Uint32,void*)`
    - E paramCallBack é:
      - `void *`
    - Para continuar executando retorna o intervalo T, caso contrário retorna 0
- Para remover o temporizador (retorna um bool)
  - `SDL_RemoveTimer(id)`

SDL & OpenGL

- Possibilidade de usar as facilidade da SDL junto com a OpenGL
- Se for usada a SDL não precisa usar a GLUT
- Algumas cuidados precisam ser tomados no momento de usar a SDL e a OpenGL
  - A inicialização da OpenGL é normal
  - `SDL_GL_SetAttribute/SDL_GL_GetAttribute (antes de setar o modo de video)`
  - `SDL_SetVideoMode(w,h,bpp,SDL_OPENGL)`
  - `SDL_GL_SwapBuffers()`

Som

- Licença LGPL
- Carregar e produzir sons em diversos canais, entretanto reproduz só uma música
- Existem uma série de funcionalidades. Entretanto nesta introdução só será apresentado o básico
- Formatos: WAVE, MIDI, OGG e MP3
- Para usá-la deve-se adicionar
  - `#include “SDL_mixer.h”`
- Download em:
  - http://www.libsdl.org/projects/SDL_mixer/

Som

- Inicializa a API
  - `Mix_OpenAudio(44100,MIX_DEFAULT_FORMAT, 2,1024)`
    - Em caso de erro retorna -1
    - Em caso afirmativo retorna 0
- Verifica se ocorreu algum erro
  - `Mix_GetError()`
    - Retorna uma mensagem de erro
- Encerra a API de som
  - `Mix_CloseAudio()`

Som

- Carregar um som (WAV, AIFF,RIFF,OGG,VOC)
  - `Mix_LoadWAV("som.wav")`
    - Em caso de erro retorna NULL
    - Em caso afirmativo retorna Mix_Chunk *
- Alocar canais de som
  - `Mix_AllocateChannels(10)`
    - Retorna o número de canais alocados
- Trocar o volume de um canal de som
  - `Mix_Volume(1,MIX_MAX_VOLUME)`
    - Se for usado -1 o valor é usado para todos canais

Som

- Reproduzir som de um canal
  - `Mix_PlayChannel(1,som,0)`
    - No caso 0 é o loop. Toca uma única vez. Se informar -1 é loop infinito
- Pausar um canal
  - `Mix_Pause(1)`
- Continuar a reprodução do som
  - `Mix_Resume(1)`
- Parar de tocar o som de um canal
  - `Mix_HaltChannel(1)`

Som

- Carrega uma música (WAVE,MOD,MIDI,OGG,MP3)
  - `Mix_LoadMUS("triha.mp3")`
    - Em caso de erro retorna NULL
    - Em caso de sucesso retorna Mix_Music *
- Tocar uma música
  - `Mix_PlayMusic(trilha,0)`
    - Sendo que 0 reproduz uma vez. E -1 é loop infinito

Rede

- Licença LGPL
- Executa no Windows, Linux, MacOS e BeOS
- Suporte ao TCP e UDP
- Para usá-la deve-se adicionar
  - `#include “SDL_net.h”`
- Download em:
  - http://www.libsdl.org/projects/SDL_net/

Rede

- Inicializa a API
  - `SDLNet_Init()`
    - Em caso de erro retorna -1
    - Em caso afirmativo retorna 0
- Verifica se ocorreu algum erro
  - `SDLNet_GetError()`
    - Retorna uma mensagem de erro
- Encerra a API de rede
  - `SDLNet_Quit()`

Rede

- Criando o endereço do **servidor**
  - `IPAdress ip;`
  - `SDLNet_ResolveHost(&ip,NULL,7654)`
    - Em caso de erro retorna -1
    - Em caso afirmativo retorna 0
- Criando o endereço no **cliente** para se conectar no **servidor**:
  - `IPAdress ipC;`
  - `SDLNet_ResolveHost(&ipC,”234.100.45.10”,7654)`

Rede

- O **servidor** atende na porta especificada
  - `TPCsocket sock;`
  - `sock = SDLNet_TCP_Open(&ip);`
    - Em caso de erro retorna NULL
    - Em caso afirmativo retorna TCPSocket
- O **cliente** se conecta no **servidor**:
  - `TPCsocket cliente;`
  - `cliente = SDLNet_TCP_Open(&ipC);`
- O **servidor** aceita pedido do cliente:
  - `TPCsocket novo_cliente;`
  - `novo_cliente = SDLNet_TCP_Accept(sock);`

Rede

- **Enviar** mensagens pelo socket
  - `SDLNet_TCP_Send(sock,msg,tam)`
    - Retorna o número de bytes enviados
    - Msg é um void *, ou seja, um ponteiro para qualquer coisa
- **Receber** mensagens pelo socket
  - `SDLNet_TCP_Recv(sock,msg,MAXLEN)`
    - Retorna o número de bytes recebidos. Não pode ser menor ou igual a zero.
    - Msg é um void *, ou seja, um ponteiro para qualquer coisa

Rede

- Encerra um scoket
  - `SDLNet_TCP_Close(sock)`