SDL 1.2 Tutorial

Step by step guidance for beginner

This document is dedicated to everyone who is interested in developing games with SDL 1.2

©Yustinus Widya Wiratama *10/10/2014*

Contents

Introduction & Development Requirement

This document provides a step-by-step tutorial to develop a simple C++ game with Dev-C++ IDE and SDL 1.2 library including SDL_ttf extension library for drawing text and SDL_mixer extension library for playing music.

1. Dev-C++ 4.9.9.2 www.bloodshed.net
2. SDL 1.2 www.libsdl.org/download-1.2.php

3. SDL_ttf http://www.libsdl.org/projects/SDL_ttf/release-1.2.html

Binary:
Linux
SDL_ttf-2.0.11-1.i386.rpm
SDL_ttf-devel-2.0.11-1.i386.rpm
SDL_ttf-2.0.11-1.x86_64.rpm
SDL_ttf-devel-2.0.11-1.x86_64.rpm
Windows
SDL_ttf-2.0.11-win32.zip
SDL_ttf-2.0.11-win32-x64.zip (64-bit Windows)
SDL_ttf-devel-2.0.11-VC.zip
Mac OS X
SDL_ttf-2.0.11.dmg (Intel 10.5+)

Requires:
The latest stable release of SDL 1.2, FreeType 2.0 or newer (except FreeType 2.1.3)

4. SDL 1.2 http://www.libsdl.org/projects/SDL_mixer/release-1.2.html

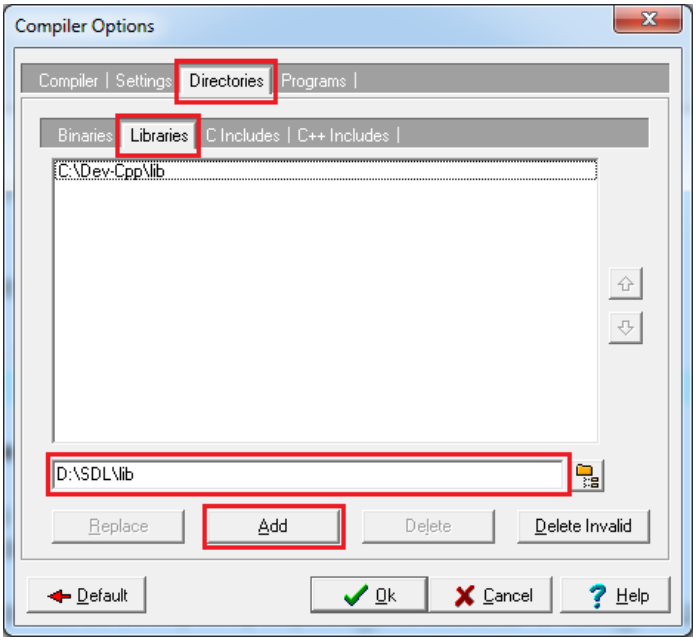
Mercurial Repository:
http://hg.libsdl.org/SDL_mixer/

Binary:
Linux
SDL_mixer-1.2.12-1.i386.rpm
SDL_mixer-devel-1.2.12-1.i386.rpm
SDL_mixer-1.2.12-1.x86_64.rpm
SDL_mixer-devel-1.2.12-1.x86_64.rpm
Windows
SDL_mixer-1.2.12-win32.zip
SDL_mixer-1.2.12-win32-x64.zip (64-bit Windows)
SDL_mixer-devel-1.2.12-VC.zip
Mac OS X
SDL_mixer-1.2.12.dmg (Intel 10.5+)

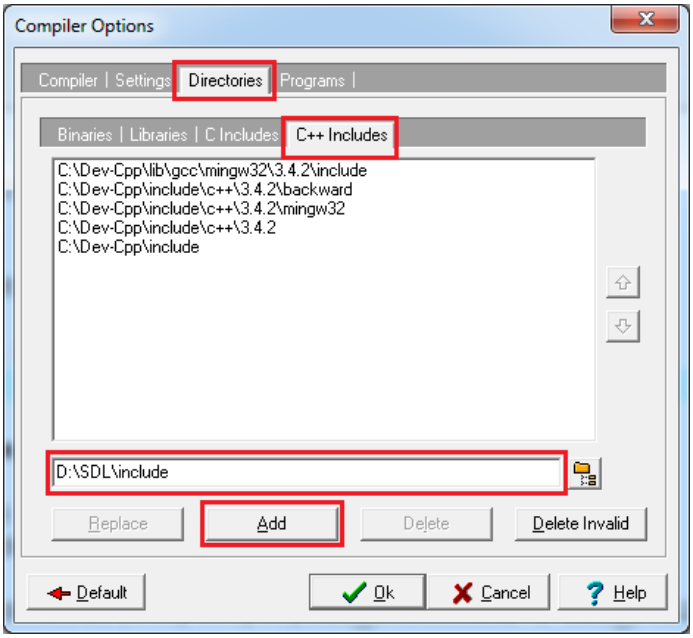
Requires:
The latest stable release of SDL 1.2

SDL Library Settings in Dev-C++

1. Unzip the Development Library and find the SDL.dll inside "bin" folder.
2. Open Dev-C++ IDE to start your first SDL 1.2 project.
3. Select menu "Tools" → "Compiler Options"
4. Within "Compiler Options" window, select "Directories" tab → "Libraries" tab and then add the reference list to the "lib" folder inside the extracted SDL 1.2 folder.

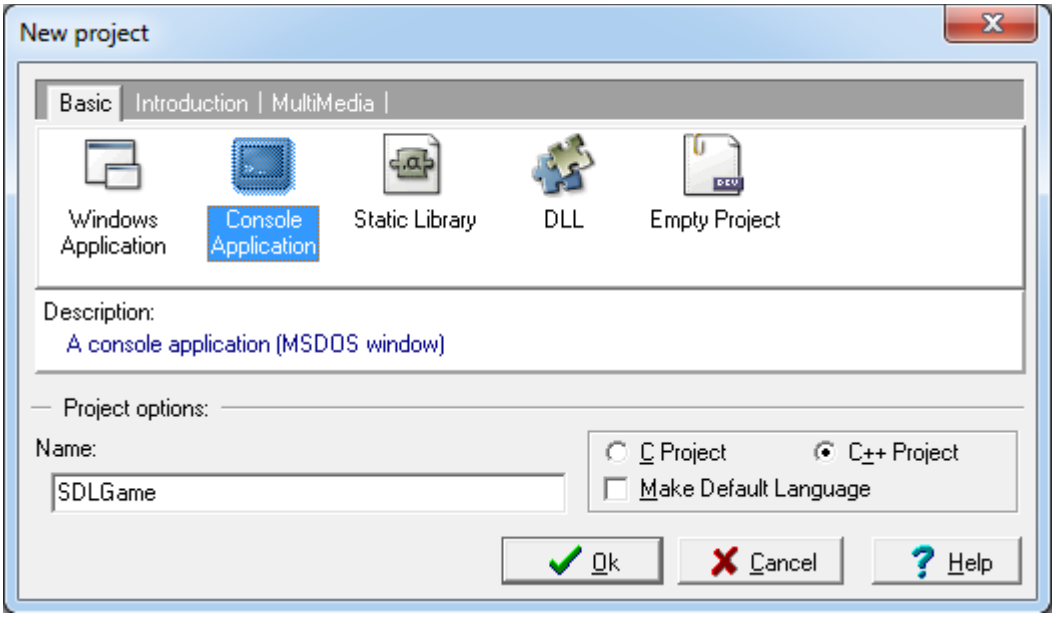


5. Within "Directories" tab, click "C++ Includes" tab and then add the "include" folder inside the extracted SDL 1.2 folder.

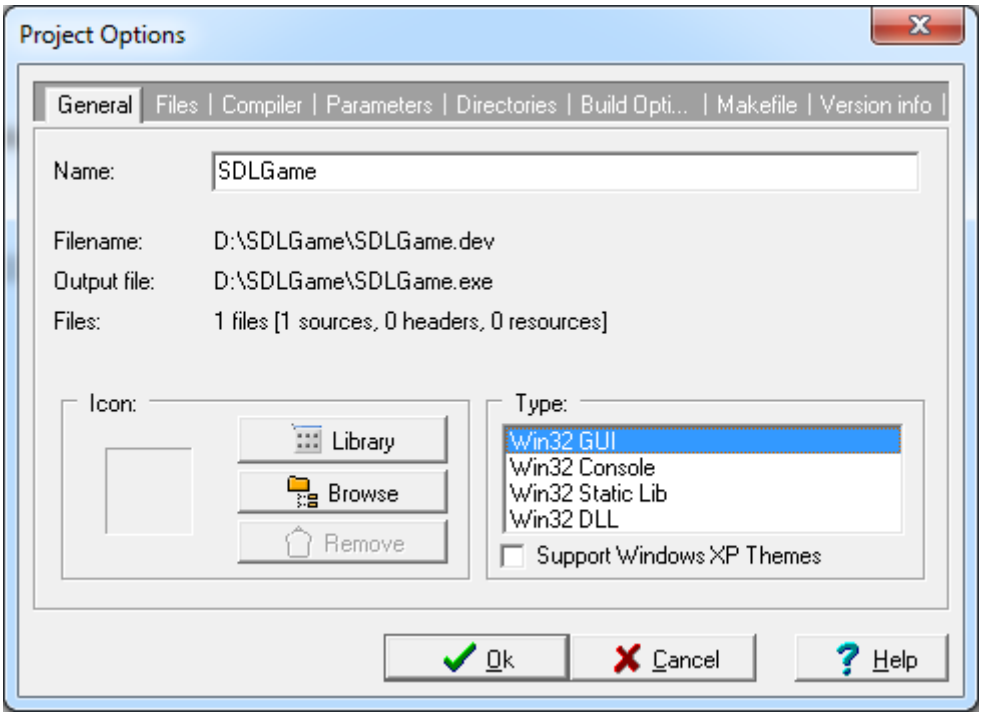


Project Settings

1. Create a new Dev-C++ project.

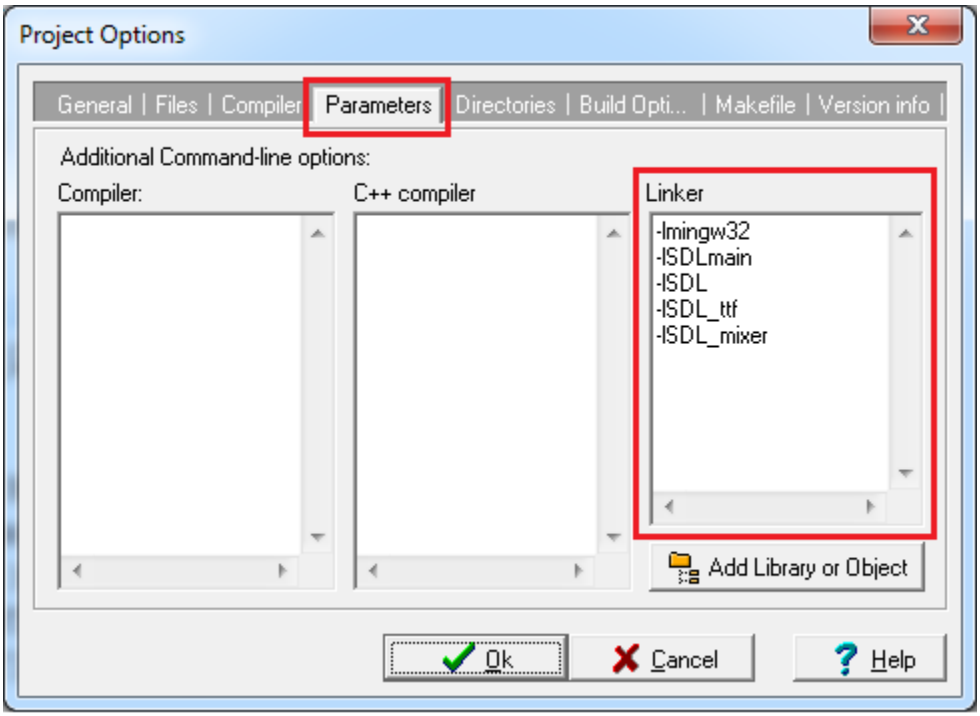


2. Select menu "Project" -> select "Project Options" -> in "General" tab, change the "Type" into "Win32 GUI"

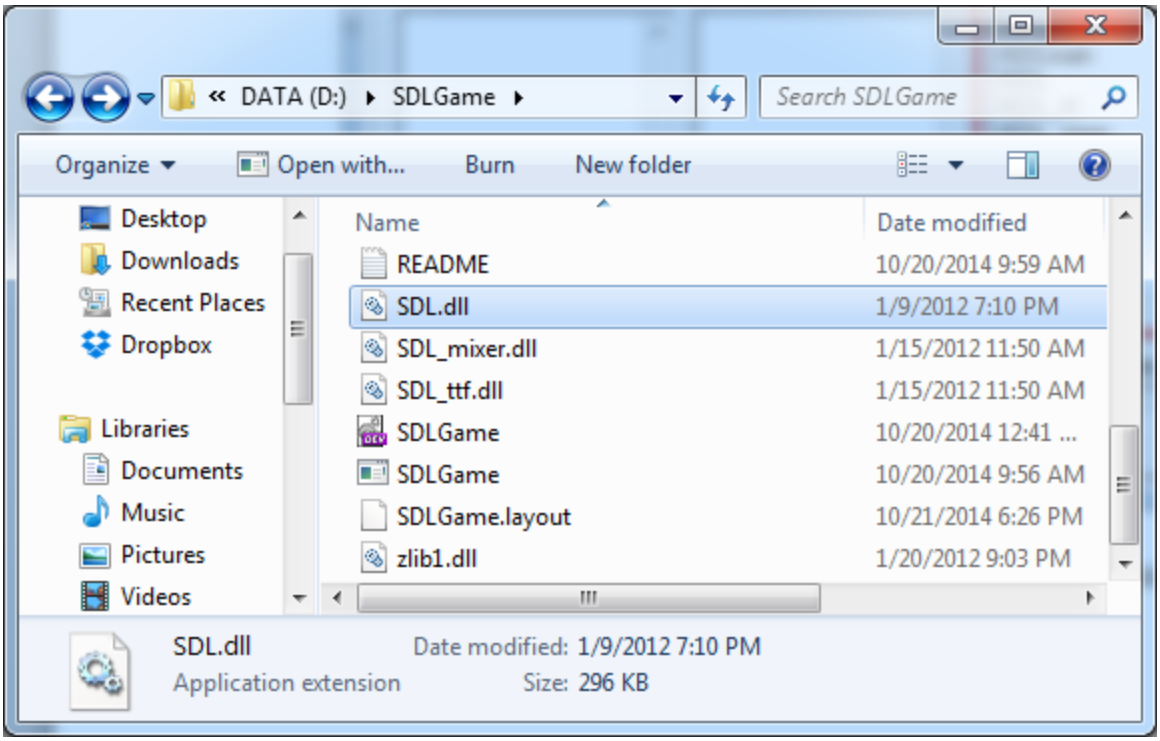


3. Within "Parameters" tab, add these commands into the "Linker"

```
-lmingw32 -lSDLmain -lSDL
```



4. Don't forget to put the SDL.dll from extracted SDL 1.2 "bin" folder together with .exe file that generated by the project.



First step into SDL 1.2

Write this code inside main.cpp of your project as your first step into SDL 1.2.

```
//Include SDL functions and datatypes
#include "SDL/SDL.h"

// SDL library REQUIRES this kind of int main()
int main( int argc, char* args[] )
{
     // The main screen
     SDL_Surface* screen = NULL;

     // Start SDL
     SDL_Init( SDL_INIT_EVERYTHING );

     // Set up screen
     screen = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );

     // Pause
     SDL_Delay( 2000 );

     // Quit SDL
     SDL_Quit();

     return 0;
}
```

Pay attention to these important parts of SDL implementation.

1. Don't forget to include the SDL header file.
2. SDL library requires a specific kind of main() function.
3. SDL needs to initialize itself by invoking SDL_Init() function.
4. SDL implementation needs to define its screen by invoking SDL_SetVideoMode() function.
5. Before you finished the program, it is necessary to quit the SDL first by calling SDL_Quit() method.

Applying Image

Here is an example how to apply images on your screen.

```
#include "SDL/SDL.h"
int main( int argc, char* args[] ){
     SDL_Surface* screen = NULL;
     SDL_Init( SDL_INIT_EVERYTHING );
     screen = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );

     // The tile
     SDL_Surface* tile = NULL;

     //Load the Background Tiles image
     tile = SDL_LoadBMP( "./images/grass_tile.bmp" );

     // Apply the tile to the screen
     for(int y=0; y<480; y+=64){
        for(int x=0; x<640; x+=64){
           //Make a temporary rectangle to hold the offsets
           SDL_Rect offset;

           //Give the offsets to the rectangle
           offset.x = x;
           offset.y = y;

           //Blit the surface
           SDL_BlitSurface( screen, NULL, tile, &offset );
        }
     }

     // Update Screen
     SDL_Flip( screen );

     // Pause
     SDL_Delay( 2000 );

     // Quit SDL
     SDL_Quit();

     return 0;
}
```

Pay attention to these main issues of applying images with SDL.

1. Remember to put an image file in appropriate directory. The code above put the grass_tile.bmp inside a folder named "images" in the same directory as the .exe file.
2. We need another `SDL_Surface*` variable to store the image that we want to show.
3. Simply use `load_image(string image_name)` to load the image.
4. Use `SDL_BlitSurface()` to apply the our `SDL_Surface* image` to the `SDL_Surface* screen`. The first parameter is the screen `SDL_Surface*`, the second is the `SDL_Rect` crop area, the third is tile' `SDL_Surface*`, and the last one is the `SDL_Rect` position where the image will be placed.
5. Don't forget to invoke `SDL_Flip()` function to update the screen.

Keyboard Control

Here is an example how to utilize your keyboard input.

```
#include "SDL/SDL.h"

int main( int argc, char* args[] ){
     SDL_Surface* screen = NULL;
     SDL_Init( SDL_INIT_EVERYTHING );
     screen = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );

     // An option to enable key press repeat
     SDL_EnableKeyRepeat(SDL_DEFAULT_REPEAT_DELAY, SDL_DEFAULT_REPEAT_INTERVAL);

     while ( 1 ){
       // SDL_Event to store the event
       SDL_Event event;
       while ( SDL_PollEvent ( &event ) ) {
          // if "X" Button on Title Bar is hit
          if ( event.type == SDL_QUIT );    // ... quit the game

          // Get Keyboard Input
          Uint8 key = SDL_GetKeyState( NULL );
          if ( key[SDLK_DOWN] );            // me->goDown();
          else if ( key[SDLK_UP] );         // me->goUp();
          else if ( key[SDLK_RIGHT] );      // me->goRight();
          else if ( key[SDLK_LEFT] );       // me->goLeft();
       }

     // Quit SDL
     SDL_Quit();

     return 0;
}
```

Pay attention to these main issues of keyboard control with SDL.

1. We need `SDL_Event` variable to store the event.
2. By default, SDL doesn't allow the users to press and hold keyboard button. However, we can enabling that functionality by invoking

   ```
   SDL_EnableKeyRepeat(int delay, int interval);
   ```

3. While `SDL_PollEvent()` returns `TRUE`, a `Uint8` variables will store the `SDL_GetKeyState( NULL )` function.
4. Some of the most common and important `SDL_GetKeyState()` are shown in the next page.

SDL Key Lists

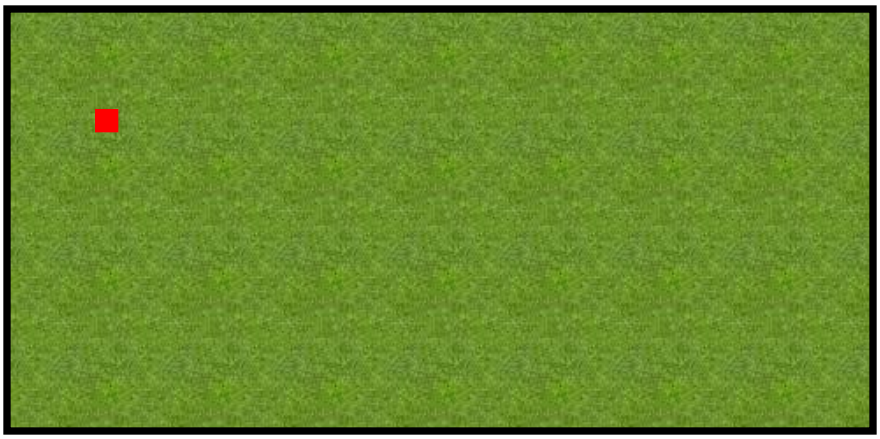
Key Name	What key
SDLK_BACKSPACE	backspace
SDLK_TAB	tab
SDLK_RETURN	return
SDLK_ESCAPE	escape
SDLK_SPACE	space
SDLK_PLUS	plus sign
SDLK_MINUS	minus sign
SDLK_a	The ‘a’ key. Same for other letters
SDLK_KP0	keypad 0. Same for other numbers
SDLK_UP	up arrow
SDLK_F1	F1. Same for other F keys

Simple Motion Animation

An animation is actually a series of images that shown in a sequence. These images can be displayed like a motion pictures automatically or triggered by specific events. Here is the logic to create character movements with a simple animation.

The red box is SDL_Rect crop area.
SDL_Rect crop area has 4 important properties.

SDL_Rect.x → top left X axis position of the crop area.
SDL_Rect.y → top left Y axis position of the crop area.
SDL_Rect.w → crop area width.
SDL_Rect.h → crop area height.



The red dot is SDL_Rect position.
SDL_Rect position has 2 important properties.

SDL_Rect.x → top left X axis position of the object.
SDL_Rect.y → top left Y axis position of the object.

```
void apply_cropped_surface( int posX, int posY, int cropX, int cropY,
                            SDL_Surface* image, SDL_Surface* screen )
{
     SDL_Rect crop;
     crop.x = cropX;
     crop.y = cropY;
     crop.w = 32;
     crop.h = 48;

     SDL_Rect position;
     position.x = posX;
     position.y = posY;

     SDL_BlitSurface( image, &crop, screen, &position );
}

int main( int argc, char* args[] ){
      ...
      while( ctr<10 ){
        apply_cropped_surface( posX, posY, cropX, cropY, image, screen );
        posX += 30;
        SDL_Delay( 200 );
        ctr++;
      }
}
```

Drawing Text with True Type Font

SDL has no native function to write any texts. However, we can utilize SDL_ttf extension library to draw texts within our screen. Follow these steps to draw texts in your program.

1. Get SDL_ttf extension library from http://www.libsdl.org/projects/SDL_ttf/release-1.2.html (as seen in "Introduction & Development Requirement" section of this tutorial)
2. Extract the .zip file, find SDL_ttf.h inside "library" folder and put it into the "lib" folder inside the extracted SDL 1.2 folder (see "SDL Library Settings in Dev-C++" section)
3. Put the SDL_ttf.dll from extracted SDL_ttf "bin" folder together with .exe file that generated by the project.

Here is an example how to draw a text string into your screen.

```
#include "SDL/SDL.h"
#include "SDL/SDL_ttf.h"

int main( int argc, char* args[] ){
     SDL_Surface* screen = NULL;
     SDL_Init( SDL_INIT_EVERYTHING );

     //SDL_Surface* variable to draw the text
     SDL_Surface* message = NULL;

     //Initialize SDL_ttf
     TTF_Init()

     //The font that's going to be used
     TTF_Font *font = NULL;

     //The color of the font
     SDL_Color textColor = { 255, 255, 255 };

     //Open the font by using any .ttf files you have (mind the file directory)
     font = TTF_OpenFont( "KGRedHands.ttf", 28 );

     screen = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
     message = TTF_RenderText_Solid( font, "SDL_ttf Drawing Text" , textColor );

     SDL_Rect offset;
     offset.x = x; offset.y = y;
     SDL_BlitSurface( screen, NULL, message, &offset );
     SDL_Flip( screen );
     SDL_Delay( 2000 );

     //Close the font
     TTF_CloseFont( font );

     //Quit SDL_ttf
     TTF_Quit();

     // Quit SDL
     SDL_Quit();

     return 0;
}
```

Playing Sounds

There is something missing if your game has no sounds. Originally, SDL has no functionality to play sounds. Therefore, to solve this problem, we can employ SDL_mixer extension library to play several sounds format. The following steps will guide you to make this happens.

1. Get SDL_mixer extension library from http://www.libsdl.org/projects/SDL_mixer/release-1.2.html (as seen in "Introduction & Development Requirement" section of this tutorial)
2. Extract the .zip file, find SDL_mixer.h inside "library" folder and put it into the "lib" folder inside the extracted SDL 1.2 folder (see "SDL Library Settings in Dev-C++" section)
3. Put the SDL_mixer.dll from extracted SDL_mixer "bin" folder together with .exe file that generated by the project.

Here is an example how to play background and sound effect sound in SDL.

```
#include "SDL/SDL.h"
#include "SDL/SDL_mixer.h"

int main( int argc, char* args[] ){
     SDL_Surface* screen = NULL;
     SDL_Init( SDL_INIT_EVERYTHING );

     //SDL_Surface* variable to draw the text
     SDL_Surface* message = NULL;

     //The background music that will be played
     Mix_Music *bgm = NULL;

     //The sound effects that will be used
     Mix_Chunk *sfx = NULL;

     //Initialize SDL_mixer
     Mix_OpenAudio( 22050, MIX_DEFAULT_FORMAT, 2, 4096 );

     //Load the .wav music files you have (mind the file directory)
     bgm = Mix_LoadMUS( "./musics/bgm.wav" );

     //Set the BGM volume
     Mix_VolumeMusic(18);

     //Set the BGM volume
     Mix_VolumeChunk(sfx, 60);

     //Play the background music
     Mix_PlayMusic( bgm, -1 );

     int ctr = 0;
```

```
    while( ctr<5 ){
      //Load the .wav sound effects files you have (mind the file directory)
      sfx = Mix_LoadWAV( "./musics/sfx.wav" );

      //Play the scratch effect
      Mix_PlayChannel( -1, sfx, 0 );


      SDL_Delay( 2000 );
    }


    //Free the sound effects
    Mix_FreeChunk( sfx );

    //Free the music
    Mix_FreeMusic( bgm );

    //Quit SDL_mixer
    Mix_CloseAudio();

    // Quit SDL
    SDL_Quit();

    return 0;
}
```