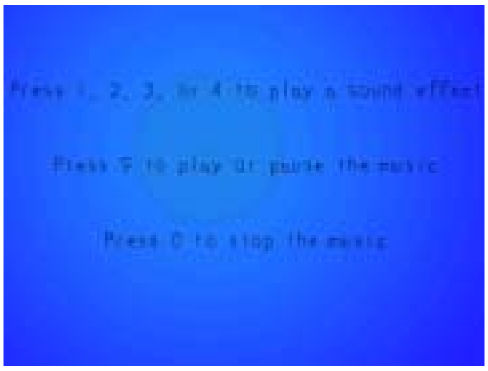
Lazy Foo' Productions

Playing Sounds


Press 1, 2, 3, or 4 to play a sound effect

Press 9 to play or pause the music

Press 0 to stop the music

Last Updated 12/09/07

Playing sound is another key concept of game programming. SDL's native sound functions can be quite confusing. So you're going to learn to play sound effects and music using the extension library SDL_mixer. SDL_mixer is an extension library that makes using sound braindead easy.

You can get SDL_mixer from here.

To install SDL_mixer just follow the extension library tutorial. Installing SDL_mixer is done pretty much the way SDL_image is, so just replace where you see SDL_image with SDL_mixer.

This tutorial covers the basics playing music and sound effects using SDL_mixer to play sound effects and "music" I made by tapping on my monitor.

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

//The sound effects that will be used
Mix_Chunk *scratch = NULL;
Mix_Chunk *high = NULL;
Mix_Chunk *med = NULL;
Mix_Chunk *low = NULL;
```

Here's the new data types we are going to be working with.

Mix_Music is the data type we use for music, and Mix_Chunk is the data type used for sound effects.

```
bool init()
{
    //Initialize all SDL subsystems
    if( SDL_Init( SDL_INIT_EVERYTHING ) == -1 )
    {
        return false;
    }

    //Set up the screen
    screen = SDL_SetVideoMode( SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_BPP, SDL_SWSURFACE

    //If there was an error in setting up the screen
    if( screen == NULL )
    {
        return false;
    }

    //Initialize SDL_ttf
    if( TTF_Init() == -1 )
    {
        return false;
    }

    //Initialize SDL_mixer
    if( Mix_OpenAudio( 22050, MIX_DEFAULT_FORMAT, 2, 4096 ) == -1 )
    {
        return false;
    }

    //Set the window caption
    SDL_WM_SetCaption( "Monitor Music", NULL );

    //If everything initialized fine
    return true;
}
```

In the initialization function, we call Mix_OpenAudio() to initialize SDL_mixer's audio functions.

Mix_OpenAudio()'s first argument is the sound frequency we use, and in this case it's 22050 which is what's recommended. The second argument is the sound format used which we set to the default. The third argument is how many channels we plan to use. We set it to 2 so we have stereo sound, if it was set to one we'd have mono sound. The last argument is the sample size, which is set to 4096.

```
bool load_files()
{
    //Load the background image
    background = load_image( "background.png" );

    //Open the font
    font = TTF_OpenFont( "lazy.ttf", 30 );

    //If there was a problem in loading the background
    if( background == NULL )
    {
        return false;
    }

    //If there was an error in loading the font
    if( font == NULL )
    {
        return false;
    }

    //Load the music
    music = Mix_LoadMUS( "beat.wav" );

    //If there was a problem loading the music
    if( music == NULL )
    {
        return false;
    }

    //Load the sound effects
    scratch = Mix_LoadWAV( "scratch.wav" );
    high = Mix_LoadWAV( "high.wav" );
    med = Mix_LoadWAV( "medium.wav" );
    low = Mix_LoadWAV( "low.wav" );

    //If there was a problem loading the sound effects
    if( ( scratch == NULL ) || ( high == NULL ) || ( med == NULL ) || ( low == NULL ) )
    {
        return false;
    }

    //If everything loaded fine
    return true;
}
```

Here we have the file loading function.

To load the music, we use Mix_LoadMUS(). Mix_LoadMUS() takes in the filename of the music file and returns the music data. It return NULL if there's an error.

To load sound effects, we use Mix_LoadWAV(). It loads the sound file of the given file name, and returns a Mix_Chunk or NULL if there was an error.

```
void clean_up()
{
    //Free the surfaces
    SDL_FreeSurface( background );

    //Free the sound effects
    Mix_FreeChunk( scratch );
    Mix_FreeChunk( high );
    Mix_FreeChunk( med );
    Mix_FreeChunk( low );

    //Free the sound effects
    Mix_FreeMusic( music );

    //Close the font
    TTF_CloseFont( font );

    //Quit SDL_mixer
    Mix_CloseAudio();

    //Quit SDL_ttf
    TTF_Quit();

    //Quit SDL
    SDL_Quit();
}
```

In the clean up function, we call Mix_FreeChunk() to get rid of the sound effects and Mix_FreeMusic() to free the music.

Then after we're done using SDL_mixer, Mix_CloseAudio() is called.

```
    //While the user hasn't quit
    while( quit == false )
    {
        //While there's events to handle
        while( SDL_PollEvent( &event ) )
        {
            //If a key was pressed
            if( event.type == SDL_KEYDOWN )
            {
```

In our main function after the intialization, file loading and background and messages have been blitted, we enter our main loop. Then we start handling events, starting with key presses.

```
                //If 1 was pressed
                if( event.key.keysym.sym == SDLK_1 )
                {
                    //Play the scratch effect
                    if( Mix_PlayChannel( -1, scratch, 0 ) == -1 )
                    {
                        return 1;
                    }
```

```
        }
        //If 2 was pressed
        else if( event.key.keysym.sym == SDLK_2 )
        {
            //Play the high hit effect
            if( Mix_PlayChannel( -1, high, 0 ) == -1 )
            {
                return 1;
            }
        }
        //If 3 was pressed
        else if( event.key.keysym.sym == SDLK_3 )
        {
            //Play the medium hit effect
            if( Mix_PlayChannel( -1, med, 0 ) == -1 )
            {
                return 1;
            }
        }
        //If 4 was pressed
        else if( event.key.keysym.sym == SDLK_4 )
        {
            //Play the low hit effect
            if( Mix_PlayChannel( -1, low, 0 ) == -1 )
            {
                return 1;
            }
        }
```

When checking the key presses, first we check if 1, 2, 3, or 4 have been pressed. These are the keys that play the sound effects.

If one of the these keys have been pressed, we call Mix_PlayChannel() to play the key's associated sound effect.

Mix_PlayChannel()'s first argument is which mixing channel we're going to play the sound in. Since it's set to -1, Mix_PlayChannel() just looks for the first channel that's available and plays the sound.

The second argument in the Mix_Chunk that's going to be played. The third is how many times the sound effect is going to loop. Since it's set to 0, it will only play once.

When there's a problem playing the sound, Mix_PlayChannel() will return -1.

```
        //If 9 was pressed
        else if( event.key.keysym.sym == SDLK_9 )
        {
            //If there is no music playing
            if( Mix_PlayingMusic() == 0 )
            {
                //Play the music
                if( Mix_PlayMusic( music, -1 ) == -1 )
                {
                    return 1;
                }
            }
```

Next we handle when 9 is pressed, which is supposed to play/pause the music.

First we check if the music is playing with Mix_PlayingMusic(). If no music is playing we call Mix_PlayMusic() to play the music.

The first argument of Mix_PlayMusic() is the music we're going to play. The second argument is how many times it will loop the music. Since it's set to -1, it will loop until it is manually stopped.

If there's a problem in playing the music Mix_PlayMusic() will return -1.

```
            //If music is being played
            else
            {
                //If the music is paused
                if( Mix_PausedMusic() == 1 )
                {
                    //Resume the music
                    Mix_ResumeMusic();
                }
                //If the music is playing
                else
                {
                    //Pause the music
                    Mix_PauseMusic();
                }
            }
        }
```

Now if music was playing when the user pressed 9, we either pause or unpause the music.

First we check of the music is playing using Mix_PausedMusic(). If it is, we unpause the music using Mix_ResumeMusic(). If the music is not paused, we pause it using Mix_PauseMusic()

```
        //If 0 was pressed
        else if( event.key.keysym.sym == SDLK_0 )
        {
            //Stop the music
            Mix_HaltMusic();
        }
    }
```

Lastly, We check if the user presses 0.

If the user presses 0, the music is stopped using Mix_HaltMusic().

Download the media and source code for this tutorial here.

I highly recommend that you download the SDL_mixer documentation, and keep it around for reference.

News FAQs Games Tutorials Articles Contact Donations