Lesson 06 – Sound & Extension Libraries

Home

Extension Libraries | Setup | SDL_Image | SDL_Mixer

Extension Libraries

While SDL provides extensive APIs for several systems, it omits some areas, and lacks features in others. Extension libraries solve this problem, adding more functionality in a modular fashion.

Popular Extensions:

Example Program

Download

```
#include <iostream>

#include <SDL.h>
#include <SDL_image.h>
#include <SDL_mixer.h>

using namespace std;

bool init();
void kill();
bool loop();

// Pointers to our window, renderer,
texture, music, and sound
SDL_Window* window;
SDL_Renderer* renderer;
SDL_Texture* texture;
Mix_Music* music;
Mix_Chunk* sound;

int main(int argc, char** args) {

	if ( !init() ) {
		system("pause");
		return 1;
	}
```

```
	}

	while ( loop() ) {
		// wait before processing
the next frame
		SDL_Delay(10);
	}

	kill();
	return 0;
}

bool loop() {

	static const unsigned char*
keys = SDL_GetKeyboardState(
NULL );

	SDL_Event e;
	SDL_Rect dest;

	// Clear the window to white
	SDL_SetRenderDrawColor(
renderer, 255, 255, 255, 255 );
	SDL_RenderClear( renderer );

	// Event loop
	while ( SDL_PollEvent( &e ) !=
0 ) {
		switch (e.type) {
			case SDL_QUIT:
				return false;
			case
SDL_KEYDOWN:
				if (
e.key.keysym.sym == SDLK_SPACE
) {
					if (
Mix_PausedMusic() == 1 ) {

Mix_ResumeMusic();
					} else {

Mix_PauseMusic();
					}
				} else if (
e.key.keysym.sym ==
SDLK_ESCAPE ) {

Mix_HaltMusic();
				}
```

```
                    break;
                case SDL_MOUSEBUTTONDOWN:
                    // Play sound once on the first available channel
                    Mix_PlayChannel( -1, sound, 0 );
                    break;
            }
        }

        // Render texture
        SDL_RenderCopy(renderer, texture, NULL, NULL);

        // Update window
        SDL_RenderPresent( renderer );

        return true;
}

bool init() {
        if ( SDL_Init( SDL_INIT_EVERYTHING ) < 0 ) {
                cout << "Error initializing SDL: " << SDL_GetError() << endl;
                return false;
        }

        // Initialize SDL_image with PNG loading subsystem
        if ( IMG_Init(IMG_INIT_PNG) < 0 ) {
                cout << "Error initializing SDL_image: " << IMG_GetError() << endl;
                system("pause");
                return false;
        }

        // Initialize SDL_mixer with our audio format
        if ( Mix_OpenAudio( 44100, MIX_DEFAULT_FORMAT, 2, 1024 ) < 0 ) {
                cout << "Error initializing SDL_mixer: " << Mix_GetError() << endl;
                return false;
        }
```

```
        window = SDL_CreateWindow(
"Example",
SDL_WINDOWPOS_UNDEFINED,
SDL_WINDOWPOS_UNDEFINED,
640, 480, SDL_WINDOW_SHOWN );
        if ( !window ) {
                cout << "Error creating
window: " << SDL_GetError()  <<
endl;
                return false;
        }

        renderer =
SDL_CreateRenderer( window, -1,
SDL_RENDERER_ACCELERATED
);
        if ( !renderer ) {
                cout << "Error creating
renderer: " << SDL_GetError() <<
endl;
                return false;
        }

        // Load image (PNG) into
surface
        SDL_Surface* buffer =
IMG_Load("test.png");
        if ( !buffer ) {
                cout << "Error loading
image test.png: " << SDL_GetError()
<< endl;
                return false;
        }

        // Create texture
        texture =
SDL_CreateTextureFromSurface(
renderer, buffer );
        // Free surface as it's no longer
needed
        SDL_FreeSurface( buffer );
        buffer = NULL;
        if ( !texture ) {
                cout << "Error creating
texture: " << SDL_GetError() << endl;
                return false;
        }

        // Load music
        music =
```

```
Mix_LoadMUS("music.wav");
        if ( !music ) {
                cout << "Error loading
music: " << Mix_GetError() << endl;
                return false;
        }

        // Load sound
        sound =
Mix_LoadWAV("scratch.wav");
        if ( !sound ) {
                cout << "Error loading
sound: " << Mix_GetError() << endl;
                return false;
        }

        // Play music forever
        Mix_PlayMusic( music, -1 );

        return true;
}

void kill() {
        SDL_DestroyTexture( texture );
        Mix_FreeMusic( music );
        Mix_FreeChunk( sound );
        texture = NULL;
        music = NULL;
        sound = NULL;

        SDL_DestroyRenderer(
renderer );
        SDL_DestroyWindow( window );
        window = NULL;
        renderer = NULL;

        IMG_Quit();
        Mix_Quit();
        SDL_Quit();
}
```

- SDL_Image
  Loads images of various types
- SDL_Mixer
  Provides a sound API
- SDL_TTF
  Provides a font loading and rendering API

- SDL_Net
  Provides a networking API
- SDL_GPU [beta]
  Replaces the rendering API
- SDL_FontCache [beta]
  Provides font caching support

We'll cover setting up and using SDL_Image and SDL_Mixer in this lesson.

Setup

To use an extension library, you must change a couple project settings.

Guides:

> Setup for Visual Studio
> Other Setup Guides

SDL_Image

As of yet, we've only been able to load bitmap images. This is a pain for several reasons—bitmaps cannot be compressed, it's difficult to get them to save alpha data, and most images downloaded from the web will not be in the correct format. SDL_Image adds a small set of functions for loading other image types, including PNG, JPG, GIF, and TIFF.

Startup

Most extensions require a startup call separate from the core SDL_Init(). For SDL_image, this is IMG_Init(). This function takes a number of flags representing the types of images you want to load. Most often, this will be IMG_INIT_JPG and IMG_INIT_PNG.

```
int result = IMG_Init( IMG_INIT_JPG | IMG_INIT_PNG );
```

```
// Check load
if ( result != 0 ) {
	cout << "Failed to initialize SDL_image: " << IMG_GetError() <<
endl;
}
```

Image Loading

The only really important function from SDL_image is IMG_Load(). It works in exactly the same way as SDL_LoadBMP(), except that it can load any image format you initialized SDL_image with.

```
SDL_Surface* image = IMG_Load("image.png");

// Check load
if ( !image ) {
	cout << "Failed to load image.png: " << IMG_GetError() << endl;
}
```

Shutdown

When you shut down your program, simply remember to call IMG_Quit() as well as SDL_Quit().

```
IMG_Quit();
SDL_Quit();
```

That's really all you need to know about SDL_image; it's a very simple and straightforward extension. You can look through the documentation to get a better grasp of how each function works.

SDL_Mixer

SDL provides an API for loading and playing audio, but it is relatively low-level and can be very complicated to use in more advanced contexts. SDL_mixer provides a streamlined sound loading and playback API similar to that of SDL_image. The API documentation can be found here.

Startup

As with SDL_image, SDL_mixer has its own initialization function,

Mix_OpenAudio(). Technically, SDL_mixer is initialized with Mix_Init(), but Mix_OpenAudio() will call it for you. Mix_OpenAudio() describes how to format the audio output. The parameters include the sample frequency (44100 is CD quality, but many games use 22050), the data format, the number of channels (1 = mono, 2 = stereo), and the chunk size (~1024 for sound effects, larger if you need to prevent skipping).

```
int result = Mix_OpenAudio( 44100, MIX_DEFAULT_FORMAT, 2,
1024 );

// Check load
if ( result != 0 ) {
	cout << "Failed to open audio: " << Mix_GetError() << endl;
}
```

Audio Loading

Loading audio is just as simple as loading images. There are two types of audio objects provided by SDL_mixer, Mix_Chunk, which represents a sound clip, and Mix_Music, which represents a longer sound clip to be used as background music. These types are loaded using the functions Mix_LoadMUS() and Mix_LoadWAV(). Both functions take the audio file path. Finally, although the function has WAV in the name, it can actually load many different audio formats, including WAV, MP3, and OGG.

```
Mix_Music* music;
Mix_Chunk* sound;

music = Mix_LoadMUS("music.wav");
sound = Mix_LoadWAV("sound.mp3");

// Check load
if( !music || !sound ) {
	cout << "Failed to load music or sound: " << Mix_GetError() <<
endl;
}
```

Playing Music

Playing music is more straightforward than playing sounds, as you can only have one music stream playing at once. Music is managed by the functions Mix_PlayMusic(), Mix_PauseMusic(), Mix_ResumeMusic(), and Mix_HaltMusic(), among others. These functions all do more or less what you'd expect, and do not take parameters, except for Mix_PlayMusic(). This function takes a pointer to the loaded music you'd like to play, and the number of times to loop it (-1 for

infinitely). If music is already playing, Mix_PlayMusic() will halt the previous stream and start anew.

```
int result = Mix_PlayMusic( music, -1 );

if ( result != 0 ) {
	cout << "Failed to play music: " << Mix_GetError() << endl;
}
```

Playing Sounds

Playing sounds is a little more complicated, as you can play a number of sounds at the same time, and you can adjust a few settings for each. Basically, you can allocate a set number of channels, each of which can represent a currently playing sound. Then, you can set a sound to play on a channel. To pause a sound, change its volume, and the like, you must then reference the channel ID of the currently playing sound.

These features are controlled by the functions Mix_AllocateChannels(), Mix_PlayChannel(), Mix_Volume, Mix_Pause(), Mix_Resume(), and Mix_HaltChannel(), among others. Again, these are mostly self-explanatory.

Mix_PlayChannel() takes three parameters: the channel to play on (or -1 for the first available channel), the Mix_Chunk to play, and the number of times to loop the sound after the first play (i.e. 0 -> plays once). It returns which channel it used.

```
int channel = Mix_PlayChannel( -1, sound, 0 );

Mix_Pause( channel );

SDL_Delay( 1000 );

Mix_Resume( channel );
```

I highly recommend looking through the documentation for information on the rest of the API, and to get a better grasp of how the API functions work together.

Shutdown

As with SDL_image, SDL_mixer has its own quit function, Mix_Quit(). This should be called at the end of your program with your other shut down functions. Additionally, you must free loaded Mix_Musics and Mix_Chunks with the functions Mix_FreeMusic() and Mix_FreeChunk() respectively.

```
Mix_FreeChunk( sound );
Mix_FreeMusic( music );

Mix_Quit();
SDL_Quit();
```

Made by Maxwell Slater © 2015–2017 | Contact me at mslater@nevada.unr.edu |

View this project on GitHub