Tutorial 12 Audio

Before starting this tutorial, you must complete the last one and have it working. Failure to do so will result in not being able to complete these steps. It is your choice but is recommended to crate a new branch each tutorial.

To implement Background Music using SDL requires the SDL Mixer framework. You should ensure you have included this in your project from the first set of SDL tutorials where you set up the project. If you experience any issues when playing Audio or the game simply won't compile or crashes, then you may be missing the SDL Mixer includes. Go back to the first week's tutorials to see how to add the correct includes to your project.

You also need some music to play. Download the two tracks from Blackboard and store them in your project folder in a new folder called Music. This should be located in the same directory as the Images folder.

With this in place, implementing Audio using SDL is a relatively simple process. Firstly, ensure you are including the required header in your Source.cpp file.

```
#include <SDL_mixer.h>
```

As we are just loading a single piece of Background Audio in this tutorial, we will keep it simple and the SDL Mixer library is a great way to do this. In the additional work of this tutorial, there are some suggestions on how to take your Audio Engine further so it can play more than a single sound file.

Now, in Source.cpp you should define a new function prototype called LoadMusic that accepts a string parameter for the file path, it doesn't return anything so that can be left as void.

As we just have a single audio clip, we can make a global variable to store this in.

```
Mix_Music* g_music = nullptr;
```

Now we need to implement the LoadMusic method we just defined. It's a very simple method, it simply should load the file from the given path, using SDL Mixer and do some simple error checking.

```
g_music = Mix_LoadMUS(path.c_str());
if(g_music == nullptr)
{
	cout << "Failed to load music. Error: " << Mix_GetError() << endl;
}
```

Now for the Mixer to function correctly, we must initialise it like we do with all of the SDL features. We should do this along with the rest of our initialisation in InitSDL(). If the Window and Renderer are successfully set up, you should try to initialise the Audio. However, if you experience lag during debugging and the audio isn't starting right away, place the code above the window creation.

```
//initialise the mixer
if (Mix_OpenAudio(44100, MIX_DEFAULT_FORMAT, 2, 2048) < 0)
{
	cout << "Mixer could not init. Error: " << Mix_GetError();
	return false;
}
```

Now it is just a simple case of loading and playing the music! As this is a background sound, we will just load it once and then set it to play forever as a loop. If InitSDL() is successful in your main() method, then you can try to load the sound file.

```
if(InitSDL())
{
	LoadMusic("Music/Mario.mp3");
	if(Mix_PlayingMusic() ==0)
	{
		Mix_PlayMusic(g_music, -1);
	}
```

Lastly, as always, we should tidy up after ourselves in the CloseSDL() method.

```
//clear up music
Mix_FreeMusic(g_music);
g_music = nullptr;
```

Great! Our code is now in a place where our Audio should play. Run your application and see if you have background music!

Additional Work

This is just a simple implementation of Audio. It currently uses the SDL_Mixer which is designed to play music in your game, in fact it is a specialised channel devoted entirely to music but other audio channels are available to play other audio clips at the same time. The idea below is similar to what you did with the Textures by placing them in their own Texture class.

Try to create a new SoundEffect class and that can play itself by giving it a Play() method. It should also be able to load via a Load(string path) method. You may wish to give the Play method some parameters to allow you to specify further details on how to play the clip (e.g. looping).

You will need to play SoundEffects on different channels. When researching how to do this look for Mix_Chunk.

You may wish to add further sounds to your game such as popping or firing noises.