**Spike:** Task 23
**Title:** Sound Board
**Author:** Sam Huffer, 101633177

## Goals / deliverables:

- A simple SDL2 application that demonstrates the following features:
  - Keys 1, 2 and 3 will each play a unique sample sound as soon as each key is pressed even if that sound is already playing.
  - Play (or pause) background music in response to key-down press “0” (zero) being used as a toggle.

## Technologies, Tools, and Resources used:

- Visual Studio 2019
- Microsoft Word
- SDL2
- SDL2 Mixer
- Online Resources
  1) Setting up SDL2 and creating an SDL2 window: https://www.youtube.com/watch?v=QQzAHcojEKg
  2) Registering keyboard input: https://www.geeksforgeeks.org/sdl-library-in-c-c-with-examples/
  3) Loading and playing music (didn’t work): https://www.youtube.com/watch?v=6IX6873J1Y8&t=605s&list=PLEETnX-uPtBVpZvp-R2daNfy9k3-L-Q3u&index=3
  4) Setting up SDL Mixer, and loading and playing music: https://www.youtube.com/watch?v=x77Rbny5iBA
  5) Fixing issue with playing music: https://www.libsdl.org/projects/SDL_mixer/docs/SDL_mixer_55.html#SEC55

## Tasks undertaken:

```
int main(int args, char* argv[])
{
    if (SDL_Init(SDL_INIT_EVERYTHING) != 0)
    {
        std::cerr << "error initialising SDL: " << SDL_GetError() << "\n" << std::endl;
    }

    SDL_Window* window = SDL_CreateWindow("SDL2 Tasks", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, 600, 400, SDL_WINDOW_SHOWN);
    SDL_Renderer* renderer = SDL_CreateRenderer(window, -1, 0);
    bool running = true;

    while (running)
    {
        SDL_Event event;

        //Events management
        while (SDL_PollEvent(&event))
        {
            switch (event.type) { ... }
        }

        SDL_RenderClear(renderer);
        //SDL_RenderCopy(renderer, text, NULL, &dest);
        SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255);
        SDL_RenderPresent(renderer);
        SDL_Delay(1000/60);
    }

    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    return 0;
}
```

*Figure 1: the main() functionality required to display a window. Note: event handling is not required for that, but is used for other stuff.*

- I copied the task 18 spike report into the task folder, stripping out the spike report’s original content and replacing it with goals and resources pertaining to the task at hand.

- I looked up how to set up SDL2 in a Visual Studio project, and found this video (res. 1). Following its instructions, I managed to set up SDL2 and create a window.

```
//Events management
while (SDL_PollEvent(&event))
{
    switch (event.type)
    {
        case SDL_QUIT:
            running = false;
            break;
        case SDL_KEYDOWN:
            switch (event.key.keysym.scancode)
            {
                case SDL_SCANCODE_ESCAPE:
                    running = false;
                    break;
                case SDL_SCANCODE_0:
                    //Play / pause background music. Don't stop it, just pause it.
                    break;
                case SDL_SCANCODE_1:
                    //Play sound 1 even if it's already playing
                    break;
                case SDL_SCANCODE_2:
                    //Play sound 2 even if it's already playing
                    break;
                case SDL_SCANCODE_3:
                    //Play sound 3 even if it's already playing
                    break;
            }

            break;
    }
}
```

*Figure 2: the keyboard input switch statement skeleton.*

```cpp
//Load a .mp3 music file
Mix_Music* AssetManager::GetMusic(std::string filename)
{
    std::string fullPath = SDL_GetBasePath();
    fullPath.append("Assets/" + filename);

    if (music[fullPath] == nullptr)
    {
        music[fullPath] = Mix_LoadMUS(fullPath.c_str());

        if (music[fullPath] == NULL)
        {
            std::cout << "Music Loading Error: " << filename.c_str() << " Error " << Mix_GetError() << std::endl;
        }
    }

    return music[fullPath];
}

//Load a .wav audio file
Mix_Chunk* AssetManager::GetChunk(std::string filename)
{
    std::string fullPath = SDL_GetBasePath();
    fullPath.append("Assets/" + filename);

    if (sfx[fullPath] == nullptr)
    {
        sfx[fullPath] = Mix_LoadWAV(fullPath.c_str());

        if (sfx[fullPath] == NULL)
        {
            std::cout << "Music Loading Error: File: " << filename.c_str() << "; Error: " << Mix_GetError() << ";" << std::endl;
        }
    }

    return sfx[fullPath];
}
```

*Figure 3: AssetManager.GetMusic() and GetSFX().*

```cpp
void AudioManager::PlayMusic(std::string filename, int loops)
{
    currentlyPlaying = filename;
    Mix_PlayMusic(assetManager->GetMusic(filename), loops);
}

bool AudioManager::IsCurrentlyPlaying(std::string filename)
{
    return filename == currentlyPlaying;
}

void AudioManager::PauseMusic()
{
    if (Mix_PlayingMusic() != 0)
    {
        Mix_PauseMusic();
    }
}

void AudioManager::ResumeMusic()
{
    if (Mix_PausedMusic() != 0)
    {
        Mix_ResumeMusic();
    }
}

void AudioManager::PlaySFX(std::string filename, int loops, int channel)
{
    Mix_PlayChannel(channel, assetManager->GetChunk(filename), loops);
}
```

*Figure 4: AudioManager's PlayMusic(), IsCurrentlyPlaying(), PauseMusic(), ResumeMusic() and PlaySFX() methods, which wrap up the underlying SDL Mixer methods and use AssetManager to load audio files.*

- I looked up how to register input using SDL2 and found this tutorial (res. 2) that outlined how to register keyboard input. From there, I put together a skeleton of a switch statement for managing the inputs required of this task (fig. 2).
- I looked up a tutorial of how to load and play music in C++, and found and tried one (res. 3), but it didn't work; it would act like it was successfully loading and playing the music, except that no sound was coming out.
- I looked around for another tutorial for how to load and play music, and found another that did eventually work (res 4). Following this tutorial, I set up SDL Mixer in my project, then put together a singleton AssetManager class that would load and .wav files on request. I took those methods and ones provided by SDL Mixer, and put together a singleton AudioManager class with methods wrapping up SDL Mixer methods and using AssetManager to retrieve audio files to play. Then I updated the skeleton switch of keyboard inputs to play SFX's on their own channels when the 1, 2 or 3 keys were pressed, and to play, pause or unpause one of two music files when the 9 or 0 keys were pressed. For this, I borrowed a few .wav files that are in use in my capstone project, adding them to the project's Debug folder (the one outside the folder with all the source code) in the sub-folder "Assets/Audio".
- I ran into an issue here where the SFX files were all playing properly, but the music file was displaying an error message when it was being loaded. I had a look at the documentation for Mix_LoadMUS() (res. 5) and found that it doesn't load .wav files but does load .mp3 files. I swapped out the .wav file I was using for a .mp3, rewrote the filenames and tried again, and found that it worked now.

## What we found out:

- How to set up SDL2 and SDL Mixer in Visual Studio (res. 1, res. 4).
- How to create an empty window using SDL2 (res. 1).
- How to register keyboard input using SDL (res. 2).

```cpp
case SDL_SCANCODE_0:
    if (!audioManager->IsCurrentlyPlaying("Audio/Var_3.mp3"))
    {
        audioManager->PlayMusic("Audio/Var_3.mp3");
    }
    else if (Mix_PausedMusic())
    {
        audioManager->ResumeMusic();
    }
    else if (Mix_PlayingMusic())
    {
        audioManager->PauseMusic();
    }
    else
    {
        audioManager->PlayMusic("Audio/Var_3.mp3");
    }
    break;
case SDL_SCANCODE_1:
    audioManager->PlaySFX("Audio/BuildingDestroyed.wav", 0, 1);
    break;
```

*Figure 5: main() playing SFX and Music audio files on appropriate keystrokes.*

- How to load and play music using SDL Mixer (res. 4, res. 5).
    - Using the SDL Mixer library rather than trying to use the base SDL2 library for loading and playing audio seems to be the better option. Certainly SDL Mixer seems to be higher-level and easier to understand than a manual implementation.
    - The tutorial I followed for this seems to indicate that it might be a good idea to have loading of all types of assets (images, audio, etc.) handled by a dedicated AssetManager while their use once loaded is handled by a manager class for that asset type.