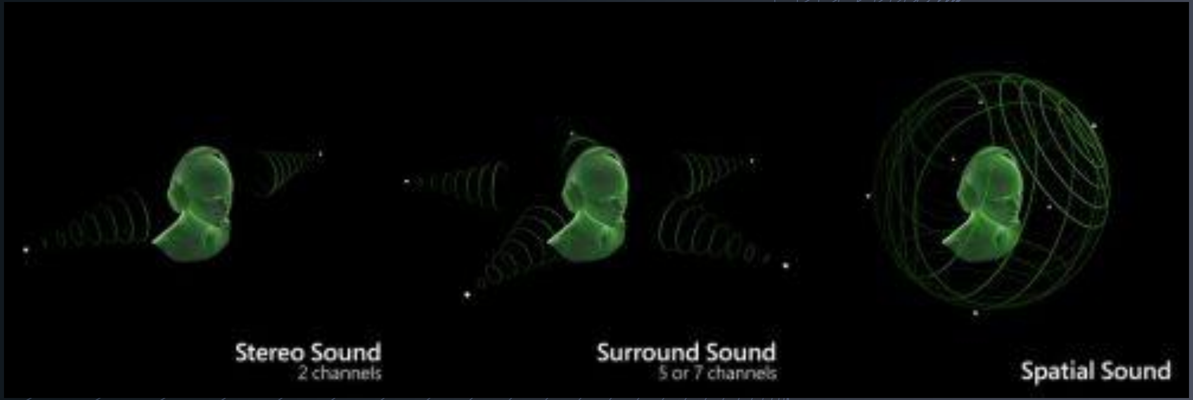
Spatial Audio & Music Manager Research

By: Aitor Luque Bodet

What is Spatial Audio?

- Spatial audio gives the illusion that sounds are coming directly from an object in a 3 dimensional space

- When you hear something as if you would in real life, in an application, that is spatial audio



Game approach

- Only some games implement real spatial audio

- We rely on audio to tell us where we should be looking

- Highly improves immersion

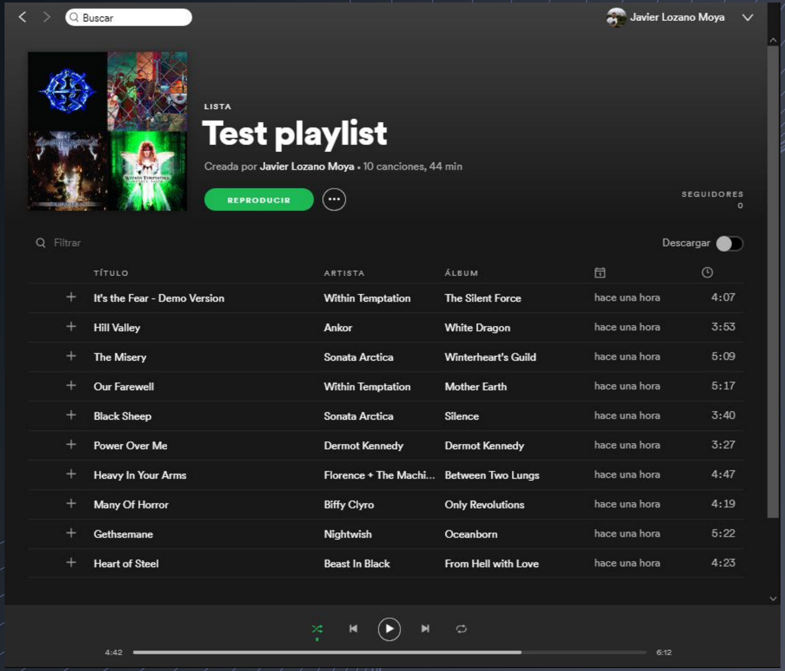
  and the player experience

In the first minute of the video linked, we can really appreciate the difference when it comes to implement spatial audio to our games.

Music Manager

- Handle, load and erase all audio files at will.

- Chain a playlist with fading



Test playlist

Creada por Javier Lozano Moya • 10 canciones, 44 min

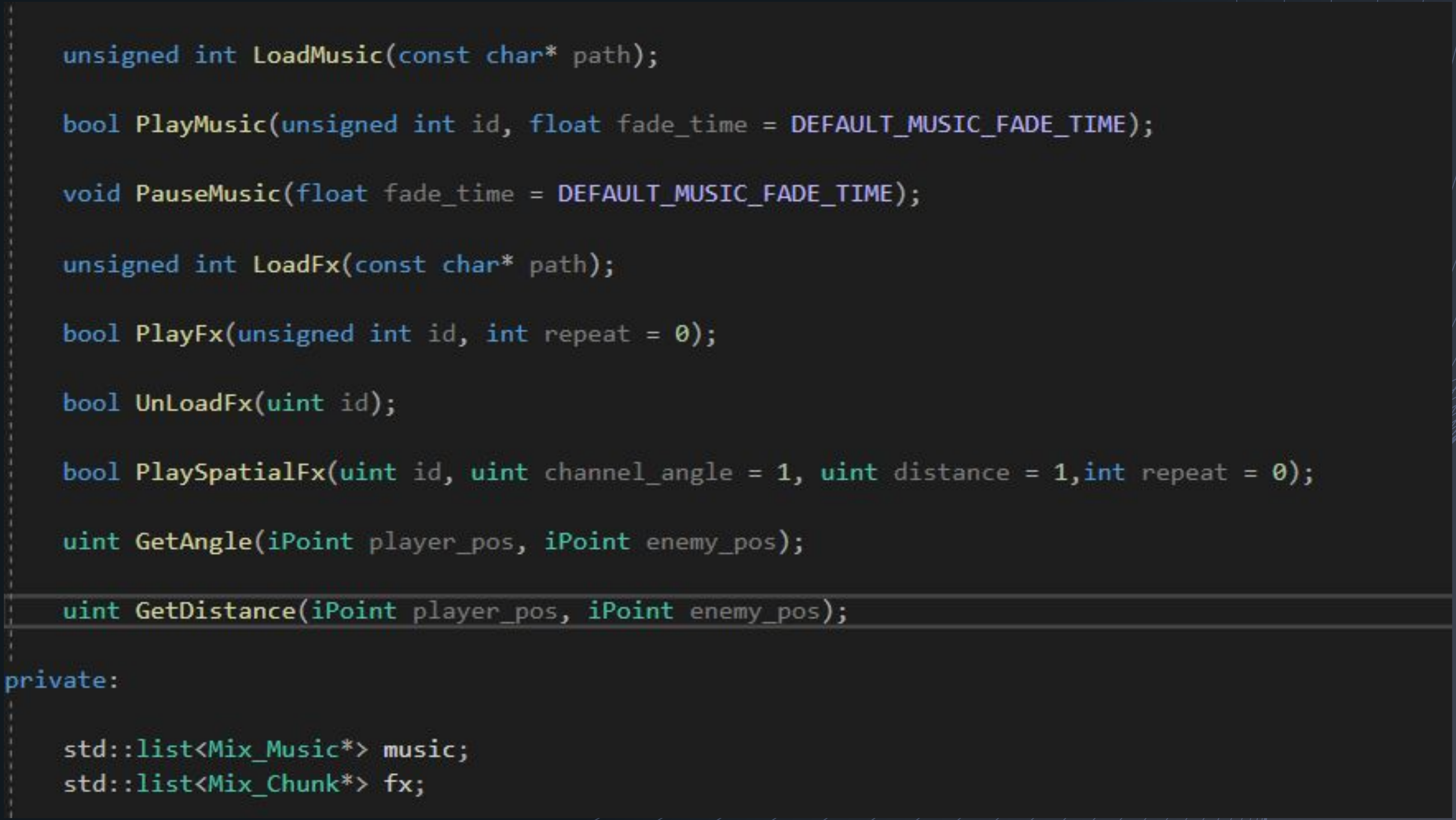
TÍTULO	ARTISTA	ÁLBUM		
It's the Fear - Demo Version	Within Temptation	The Silent Force	hace una hora	4:07
Hill Valley	Ankor	White Dragon	hace una hora	3:53
The Misery	Sonata Arctica	Winterheart's Guild	hace una hora	5:09
Our Farewell	Within Temptation	Mother Earth	hace una hora	5:17
Black Sheep	Sonata Arctica	Silence	hace una hora	3:40
Power Over Me	Dermot Kennedy	Dermot Kennedy	hace una hora	3:27
Heavy In Your Arms	Florence + The Machi...	Between Two Lungs	hace una hora	4:47
Many Of Horror	Biffy Clyro	Only Revolutions	hace una hora	4:19
Gethsemane	Nightwish	Oceanborn	hace una hora	5:22
Heart of Steel	Beast In Black	From Hell with Love	hace una hora	4:23

My approach and implementation

Core SDL_Mixer:

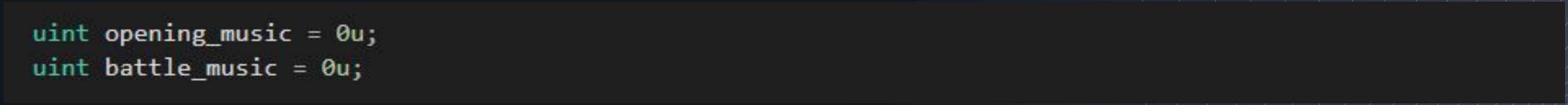
- Mix_Chunk
- Mix_Music
- Channel
- Mix_SetPositioning()
- Mix_PlayChannel()

Audio Class



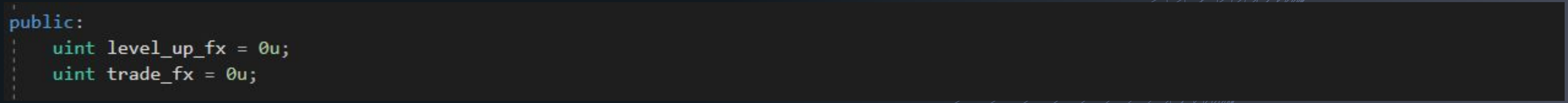
```
    unsigned int LoadMusic(const char* path);

    bool PlayMusic(unsigned int id, float fade_time = DEFAULT_MUSIC_FADE_TIME);

    void PauseMusic(float fade_time = DEFAULT_MUSIC_FADE_TIME);

    unsigned int LoadFx(const char* path);

    bool PlayFx(unsigned int id, int repeat = 0);

    bool UnLoadFx(uint id);

    bool PlaySpatialFx(uint id, uint channel_angle = 1, uint distance = 1,int repeat = 0);

    uint GetAngle(iPoint player_pos, iPoint enemy_pos);

    uint GetDistance(iPoint player_pos, iPoint enemy_pos);

private:

    std::list<Mix_Music*> music;
    std::list<Mix_Chunk*> fx;
```

Scene Class



```
uint opening_music = 0u;
uint battle_music = 0u;
```

Enemy Class

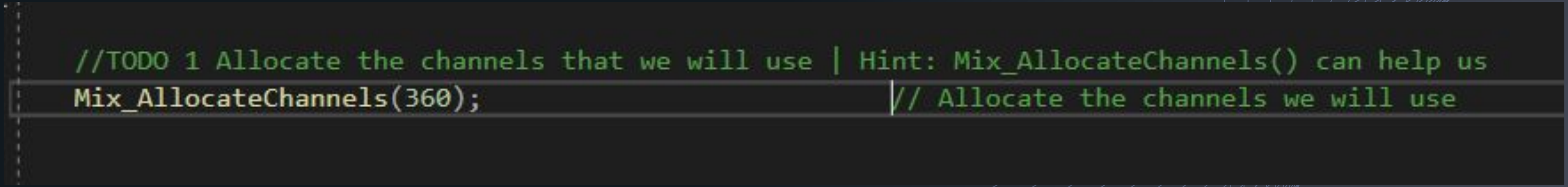


```
public:
    uint level_up_fx = 0u;
    uint trade_fx = 0u;
```

Exercises - TODOs

TODO 1

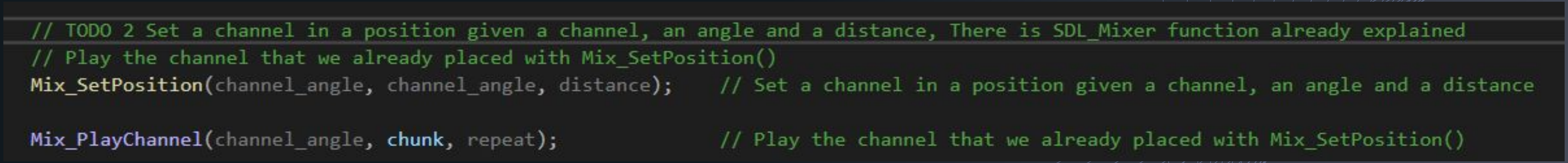
- First of all we need to allocate those channels that we will be using later on



```
//TODO 1 Allocate the channels that we will use | Hint: Mix_AllocateChannels() can help us
Mix_AllocateChannels(360);                          // Allocate the channels we will use
```

TODO 2

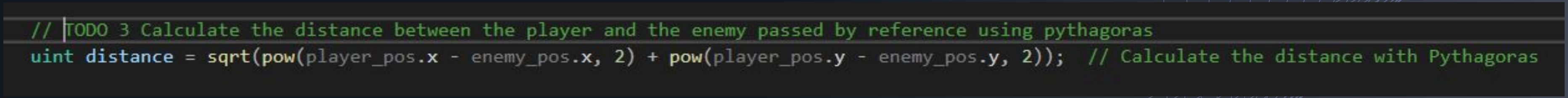
- Given a channel, an angle and a distance we need to set the new channel that we are going to use.



```
// TODO 2 Set a channel in a position given a channel, an angle and a distance, There is SDL_Mixer function already explained
// Play the channel that we already placed with Mix_SetPosition()
Mix_SetPosition(channel_angle, channel_angle, distance);    // Set a channel in a position given a channel, an angle and a distance

Mix_PlayChannel(channel_angle, chunk, repeat);              // Play the channel that we already placed with Mix_SetPosition()
```

TODO 3

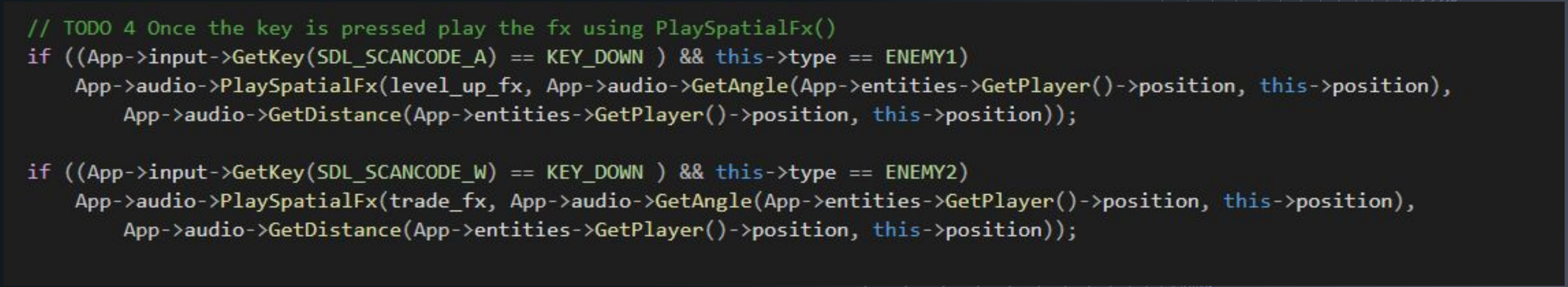
- Calculate the distance between the emitter and the player, we will use an enemy as the emitter of the fx



```
// TODO 3 Calculate the distance between the player and the enemy passed by reference using pythagoras
uint distance = sqrt(pow(player_pos.x - enemy_pos.x, 2) + pow(player_pos.y - enemy_pos.y, 2));  // Calculate the distance with Pythagoras
```

TODO 4

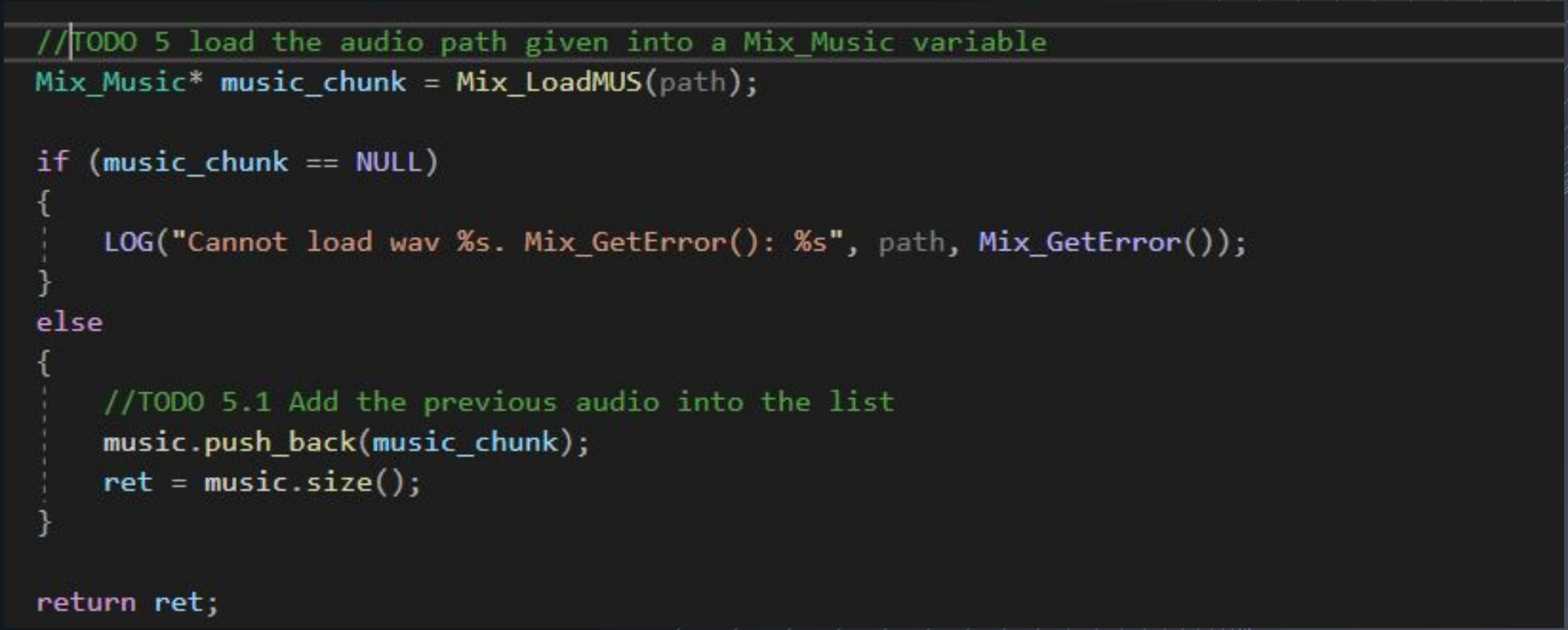
- Play the convenient Fx after pressing the debug key using the function PlaySpatialFx()



```
// TODO 4 Once the key is pressed play the fx using PlaySpatialFx()
if ((App->input->GetKey(SDL_SCANCODE_A) == KEY_DOWN ) && this->type == ENEMY1)
    App->audio->PlaySpatialFx(level_up_fx, App->audio->GetAngle(App->entities->GetPlayer()->position, this->position),
        App->audio->GetDistance(App->entities->GetPlayer()->position, this->position));

if ((App->input->GetKey(SDL_SCANCODE_W) == KEY_DOWN ) && this->type == ENEMY2)
    App->audio->PlaySpatialFx(trade_fx, App->audio->GetAngle(App->entities->GetPlayer()->position, this->position),
        App->audio->GetDistance(App->entities->GetPlayer()->position, this->position));
```

TODO 5

- Given a path passed by reference, load it into a Mix_Music variable.



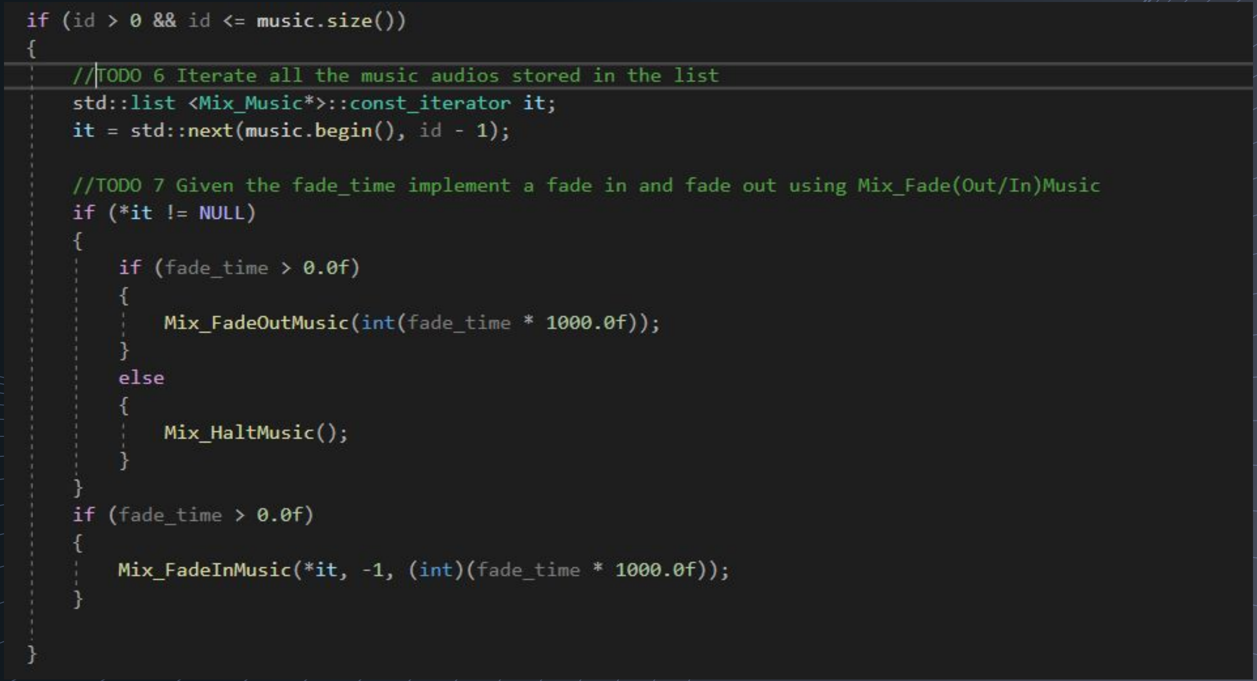
```
//TODO 5 load the audio path given into a Mix_Music variable
Mix_Music* music_chunk = Mix_LoadMUS(path);

if (music_chunk == NULL)
{
    LOG("Cannot load wav %s. Mix_GetError(): %s", path, Mix_GetError());
}
else
{
    //TODO 5.1 Add the previous audio into the list
    music.push_back(music_chunk);
    ret = music.size();
}

return ret;
```

TODO 6

- Iterate the audios previously loaded into the list

TODO 7

- Given an specific fade time fade out and face in the music when changing between them



```
if (id > 0 && id <= music.size())
{
    //TODO 6 Iterate all the music audios stored in the list
    std::list <Mix_Music*>::const_iterator it;
    it = std::next(music.begin(), id - 1);

    //TODO 7 Given the fade_time implement a fade in and fade out using Mix_Fade(Out/In)Music
    if (*it != NULL)
    {
        if (fade_time > 0.0f)
        {
            Mix_FadeOutMusic(int(fade_time * 1000.0f));
        }
        else
        {
            Mix_HaltMusic();
        }
    }
    if (fade_time > 0.0f)
    {
        Mix_FadeInMusic(*it, -1, (int)(fade_time * 1000.0f));
    }

}
```

Thank you for your attention!

Contact:

Mail: aitor97alb@gmail.com

Github: https://github.com/Aitorlb7