DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTER SCIENCE

Audio in SFML

Sound and Music

Lecture Time!

- SFML: Audio
- Pitch and Duration: The Connection
- Homework: Duh, Sound and Music

DISCS

Sound VS Music

- Consider audio in a game:
  - Metallic clangs, grunts, laser fire, etc.
  - Background music, theme songs, speeches, etc.
- Differences between the two?

Sound VS Music

- Despite the actual meaning of the word, *sound* can be used to refer to the (usually) short audio snippets that are played in response to an in-game event

Nyanko Days, episode 1

Sound VS Music

- The term *music* is often used to refer to (usually) long audio assets that are not as time-sensitive as sound and are often looped

Diablo 3

Sound VS Music

- SFML provides two classes for playing audio, differing in how they handle the actual asset
- `sf::Sound` requires “pre-loaded” audio data in the form of a `sf::SoundBuffer`
- `sf::Music` streams audio data from a file

Sound VS Music

- `sf::Sound` is good for small/short audio assets that can fit in memory and that should have no delay when played
- `sf::Music` is good for large/compressed audio assets that would otherwise take up huge chunks of memory and have long load times

Sound

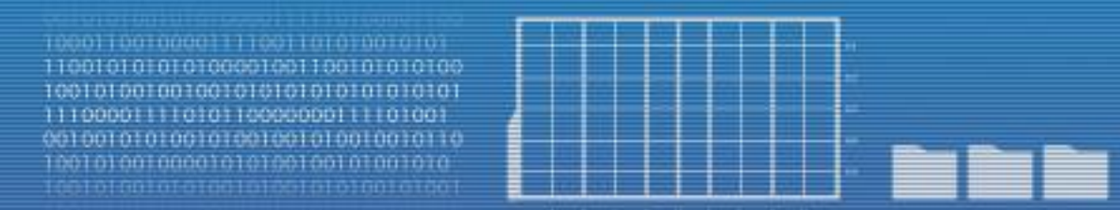
```
sf::SoundBuffer buffer;
if( !buffer.loadFromFile( "ex.wav" ) )
{    // failed to load sound file
     // ...
}

sf::Sound sound;
sound.setBuffer( buffer );
sound.play();
```

Music

```
sf::Music music;
if( !music.openFromFile( "ex.wav" ) )
{    // failed to load music file
     // ...
}

music.play();
// other than initialization,
// sf::Sound and sf::Music have
// almost exactly the same functions
```

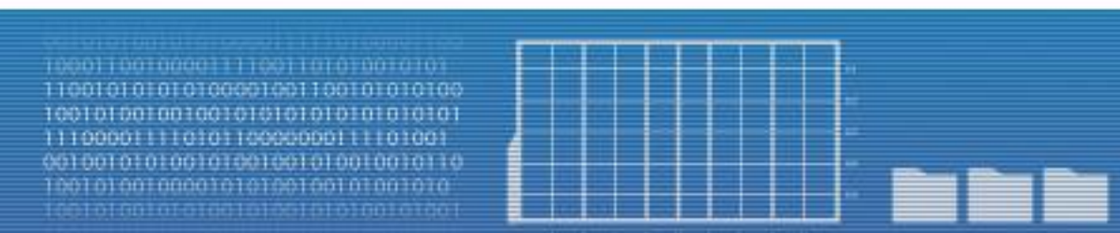


How Convenient

- Each `sf::Sound` / `sf::Music` instance plays its assigned audio asset on a separate thread, so `play()` does not block
- If you need your program to “wait” until the audio asset is finished playing, you’re going to have to get creative

How Convenient

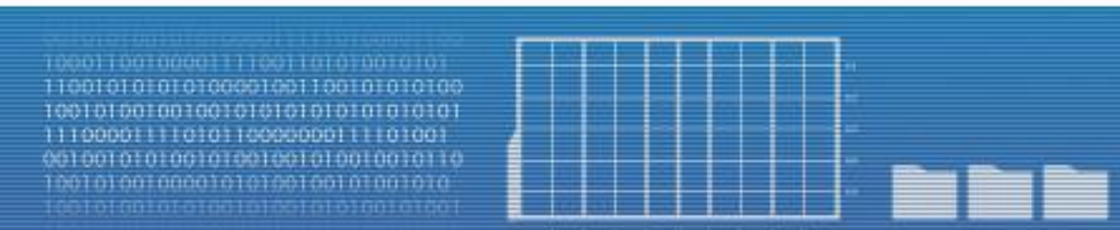
- You can `pause()` SFML audio instances
- Calling `play()` on paused audio will resume it
- If you want to halt and “rewind” an instance, you can use `stop()`
- You can also set it to loop through a `setLoop( true )` call
  - Audio does not loop by default



How Convenient

- If you need to check the status of an audio instance, you can call `getStatus()`, which returns one of the following (no explanations necessary)
  - `sf::SoundSource::Playing`
  - `sf::SoundSource::Paused`
  - `sf::SoundSource::Stopped`

```
if( music.getStatus() == sf::SoundSource::Playing )
{   // ...
```



How Convenient

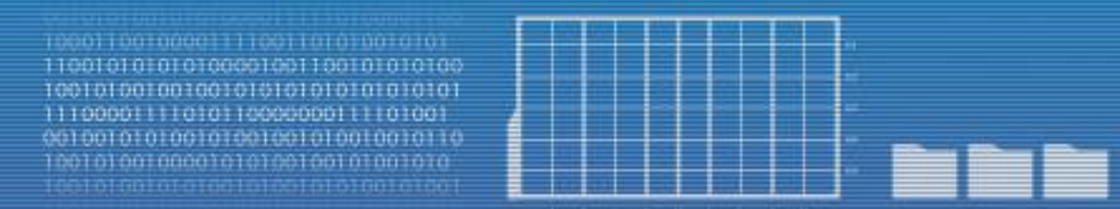
- You can control where audio will play through a `setPlayingOffset()` call
  - Example: `music.setPlayingOffset( sf::seconds( 0 ) );`
- Useful to reduce number of `SoundBuffer`s (preferably just one) but still allow a wide variety of `Sound`s

How Convenient

- Changing various audio instance characteristics:
  - `setPitch( float );`
    `// arg > 0`
    `// default 1.0`
  - `setVolume( float );`
    `// 0 <= arg <= 100`
    `// default 100`

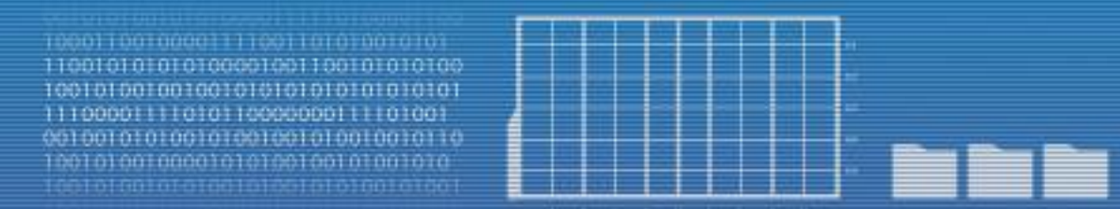
Warning

- SFML supports WAV, OGG/Vorbis and FLAC
- SFML does NOT support MP3



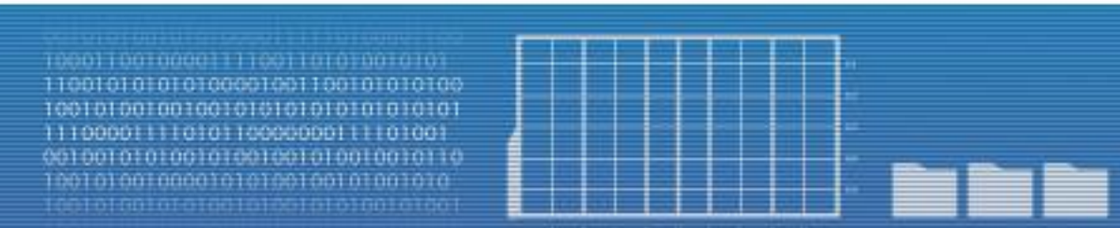
Warning

- There is a maximum number of `Sound` instances that you can have
  - SFML dev website recommends that you do not exceed 256



Warning

- If you need multiple sounds playing at the same time, you will need multiple `Sound` instances
  - Having just one means being able to play only one audio asset at a time



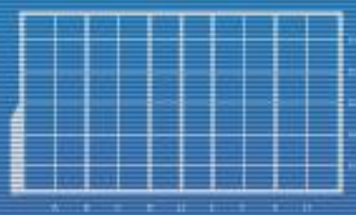
Warning

- There is also an issue with creating SFML audio instances using `new`
- On termination, this may cause the following error:

```
AL lib: (EE) alc_cleanup: 1 device not closed
```

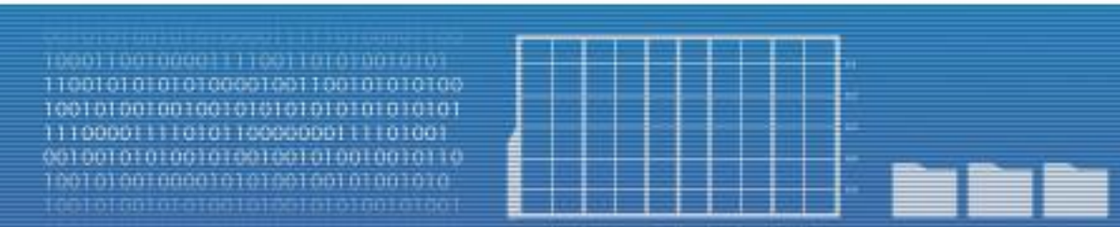
Warning

- You may need to use fixed-size arrays
- If you still want to use `new`, you must manually delete the instances before exiting to prevent the error



Warning

- `Music` instances cannot be copied
- If you want to pass a `Music` instance to a function, pass a pointer to it instead



Warning

- Do not destroy `SoundBuffer`s or `Music` sources when you still need the corresponding `Sound` or `Music` instances to play
- Even by accident
  - Example: A local variable in a function call after returning from that function
  - Applies to almost every data type in C++

Warning

- Make sure you clean up before program termination
- `stop()` all SFML audio instances
  - Might not be necessary, but it's good programming practice

Pitch (Im)perfect?

- Ever watched a movie, with audio still playing, at double or even quadruple speed?

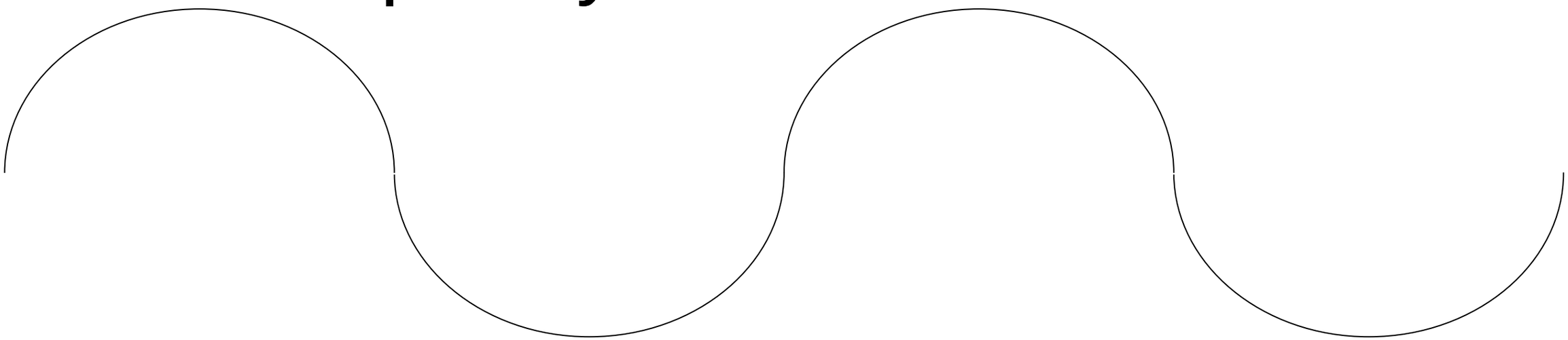
https://upload.wikimedia.org/wikipedia/en/4/42/Alvin_and_the_chipmunks1958.jpg

Pitch (Im)perfect?

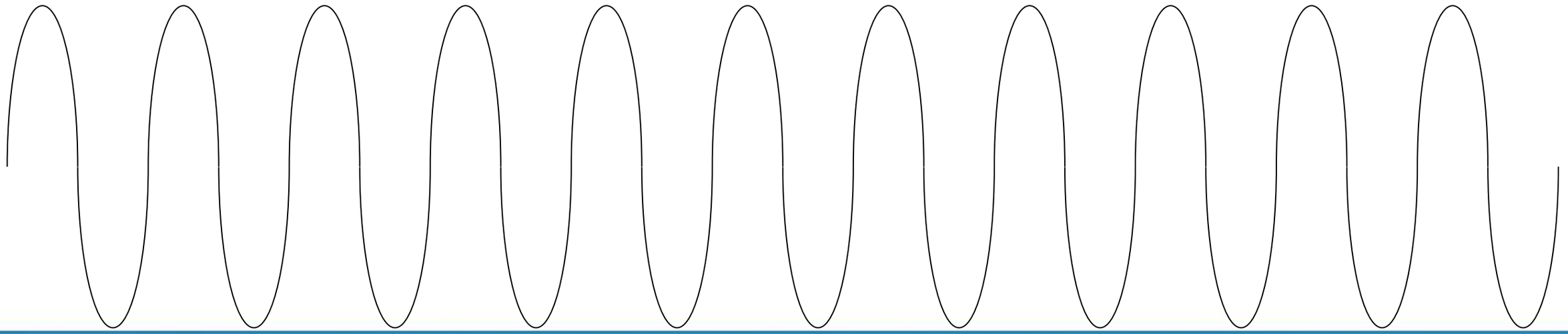
- Sound is the result of a wave caused by vibrations
- Waves have a characteristic called *frequency* which refers to the number of oscillations / periods over time

Pitch (Im)perfect?

- Low frequency



- High frequency



Pitch (Im)perfect?

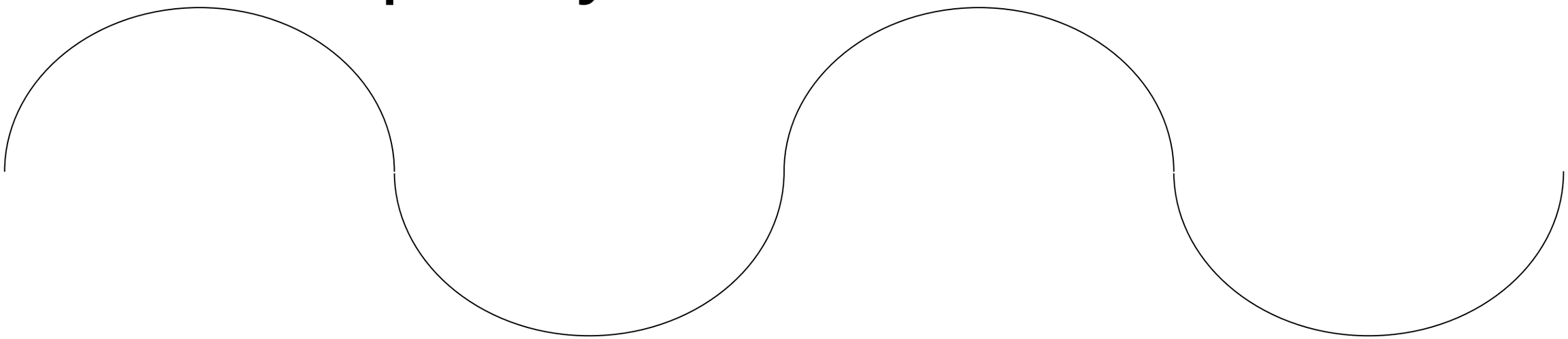
- Pitch is actually just frequency
- Changing the pitch simply changes the number of oscillations per unit time but does NOT change the total number of oscillations

Pitch (Im)perfect?

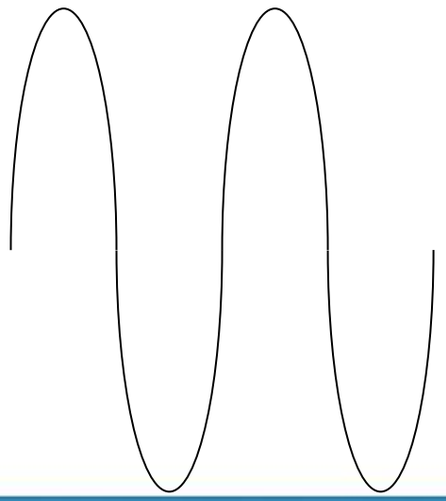
- Therefore, changing pitch will affect the audio's duration
  - Increased pitch will result in a shorter duration
  - Decreased pitch will result in a longer duration

Pitch (Im)perfect?

- Low frequency



- High frequency



Homework

- Create a program that loads two files: a music file and a sound file

  `Music.wav & Sound.wav`

- It should support the commands found in the succeeding slides
  - You may assign one key to each command

Homework

- Help
  - Displays list of commands
  - Also displays the key for each command
  - Should be automatically triggered once, when the program has finished loading and is ready to play audio assets
- Yes, this is a command

Homework

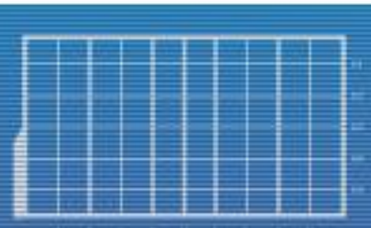
- Play/pause music
  - By default, plays the music from the beginning
  - Pauses the music if it's already playing
    - Resumes the music if it's paused
- Note: This is ONE command (meaning only ONE key gets used for this)

Homework

- Restart music
  - Plays the music from the beginning, regardless of its current status
  - Only ONE music instance, so there shouldn’t be any overlapping music

Homework

- Increase music pitch
  - Increases pitch by 0.1
  - No maximum pitch
  - Only affects music, even while it’s playing



Homework

- Decrease music pitch
  - Decreases pitch by 0.1
  - Minimum of 0.1 (can be less but do not let it hit zero)
- Only affects music, even while it’s playing

Homework

- Increase music volume
  - Increases volume by 2.5
  - Maximum of 100
  - Only affects music, even while it’s playing

Homework

- Decrease music volume
  - Decreases volume by 2.5
  - Minimum of 0
  - Only affects music, even while it’s playing

Homework

- Play sound
  - Plays the sound from the beginning
  - Can overlap music
  - Can overlap previously played sounds
- Note: This is ONE command (meaning only ONE key gets used for this)

Homework

- Increase sound pitch
  - Increases pitch by 0.1
  - No maximum pitch
  - Only affects sounds played after this command input

Homework

- Decrease sound pitch
  - Decreases pitch by 0.1
  - Minimum of 0.1 (can be less but do not let it hit zero)
- Only affects sounds played after this command input

Homework

- Increase sound volume
  - Increases volume by 2.5
  - Maximum of 100
  - Only affects sounds played after this command input

Homework

- Decrease sound volume
  - Decreases volume by 2.5
  - Minimum of 0
  - Only affects sounds played after this command input

References

- http://www.sfml-dev.org/tutorials/2.5/audio-sounds.php

DISCS