SFML - Zvuk i glazba

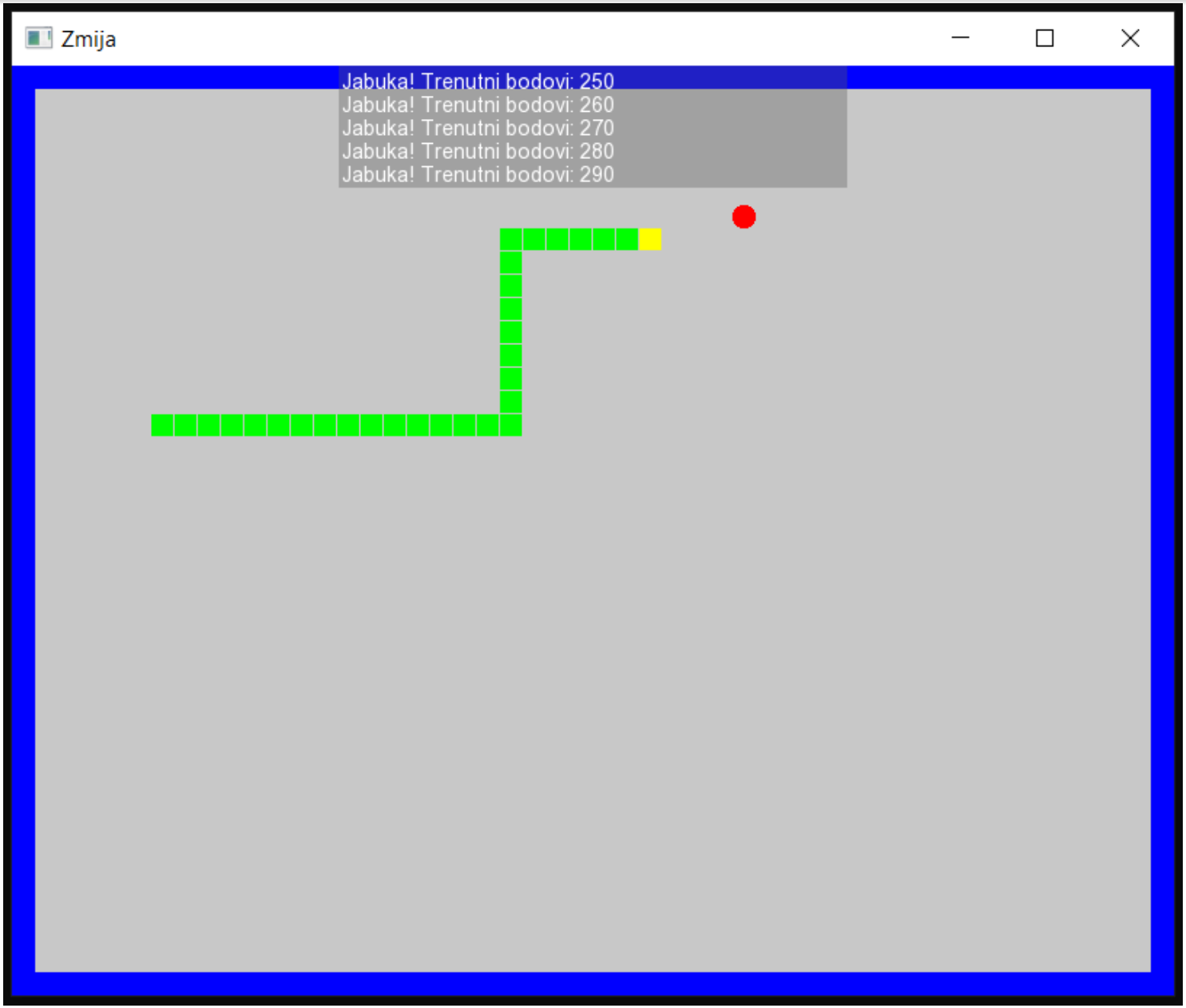
Objektno programiranje - 13. vježbe

Sebastijan Horvat

Prirodoslovno-matematički fakultet,
Sveučilište u Zagrebu

14. lipnja 2023. godine

Igra `Zmija` (kod se može preuzeti na web-stranici kolegija)

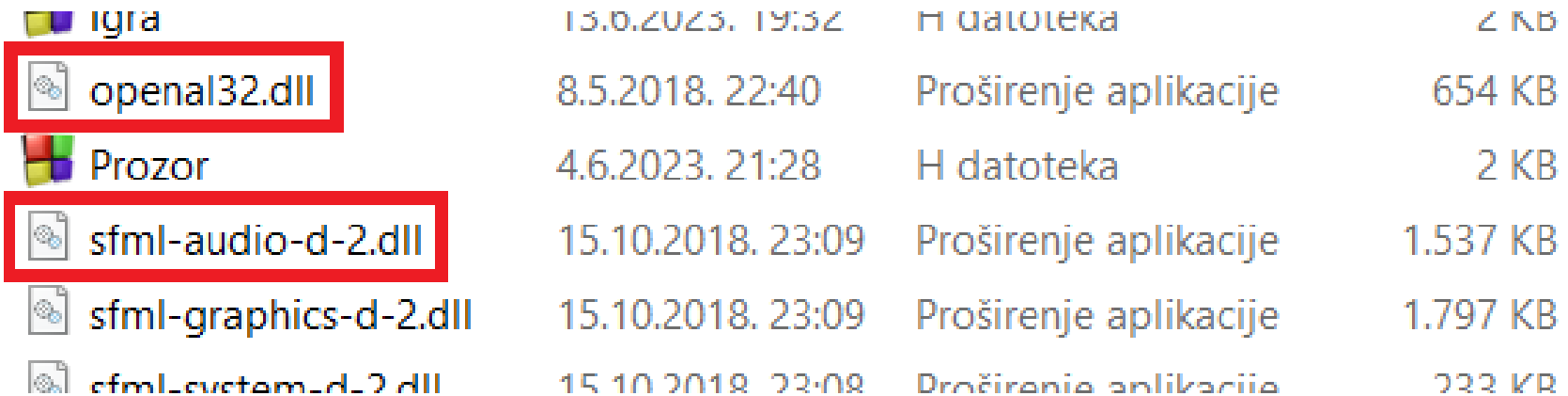


Rad s modulom `Audio`

- u kodu:

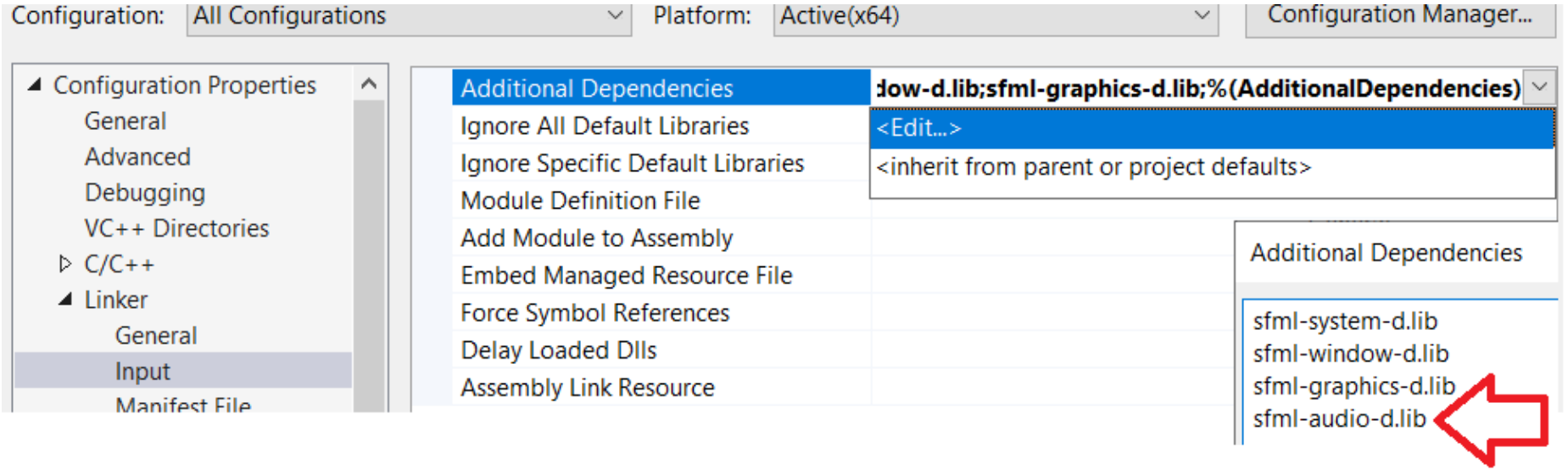
```
#include <SFML/Audio.hpp>
```

- u mapi našeg projekta (gdje je `.vcxproj` datoteka), uz dosadašnje `sfml-graphics-d-2.dll`, `sfml-system-d-2.dll` i `sfml-window-d-2.dll`, trebamo još i ove: **`sfml-audio-d-2.dll`** i **`openal32.dll`**



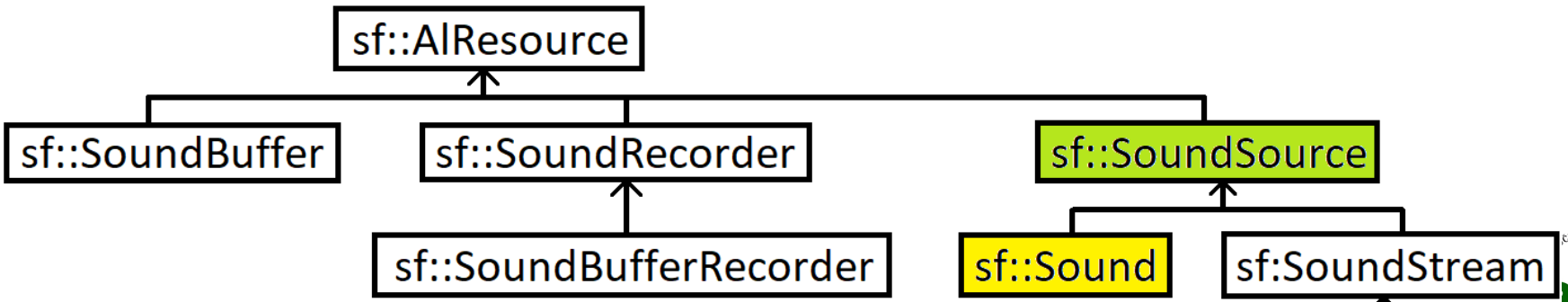
Rad s modulom `Audio` (nastavak)

- u svojstvima projekta pod *Linker* → *Input* → *Additional Dependencies* dodati **`sfml-audio-d.lib`**



Dvije klase za audio zapise: `sf::Sound` i `sf::Music`

- pružaju slične funkcije, a razlikuju se u načinu rada
- **`sf::Sound`** - reproducira **učitane** audio podatke iz **`sf::SoundBuffer`** objekta
- koristi se za kratke zvučne efekte (koraci, pucnjevi i sl.) koji zauzimaju malo memorije i za koje ne želimo da potecijalno zastajkuju pri izvođenju
- **`sf::Music`** - **ne učitava** audio podatke u memoriju nego ih reproducira iz izvorne datoteke
- najčešće se koristi za izvođenje komprimirane glazbe koja traje više minuta (i koja bi se predugo učitavala i trošila previše memorije)



(Napomena: A-L-R-E-S-O-R-C-E)

Zvuk - klasa `sf::Sound`

- audio podaci spremljeni u posebnoj klasi `sf::SoundBuffer`

Dio koda iz datoteke **`Sound.hpp`**:

```
class SFML_AUDIO_API Sound : public SoundSource {
  public:
    ...
  private:
    const SoundBuffer* m_buffer;
};
```

- odnos `sf::Sound` i `sf::SoundBuffer` klasa je kao odnos `sf::Sprite` i `sf::Texture` klase

$\Rightarrow$ slična česta greška: (slučajno) uništimo *sound buffer* kojeg zvuk još uvijek koristi (primjer na sljedećem slajdu)

Primjer greške: uništimo zvučni spremnik kojeg zvuk još treba

Primjer. Zvučni spremnik kojeg još trebamo uništen (uočiti doseg!):

```
sf::Sound ucitajZvuk(std::string putanja) {
    sf::SoundBuffer spremnik;    //lokalno!
    spremnik.loadFromFile(putanja);
    return sf::Sound(spremnik);
}

...

sf::Sound zvuk = ucitajZvuk("datoteka.wav");
sound.play();    ✗
```

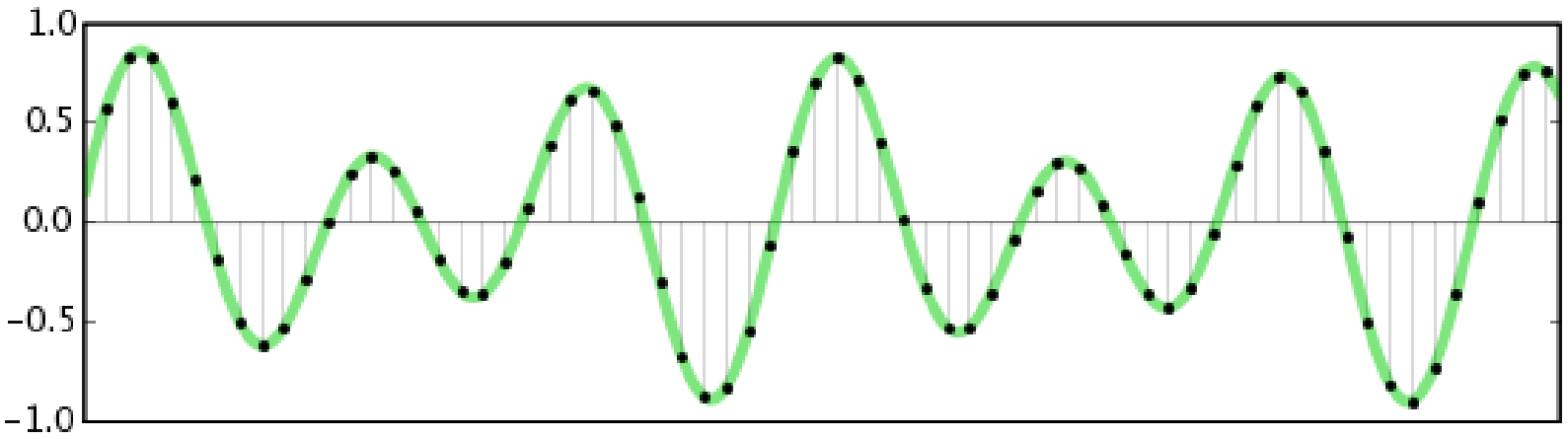
Klasa `sf::SoundBuffer`

- samo spremnik za audio uzorke koji definiraju zvuk (ne može reproducirati audio podatke - za to trebamo `sf::Sound` klasu i njene funkcije)
- uzorak (*sample*) je 16-bitni cijeli broj (s predznakom) = amplituda zvuka u određenom trenutku
- iz datoteke `SoundBuffer.hpp`:

```
class SFML_AUDIO_API SoundBuffer : AlResource {

  ...

  private:

      ...

      std::vector<Int16> m_samples;

      ...

};
```

Klasa `sf::SoundBuffer` (nastavak)

- zvuk se rekonstruira izvođenjem tih uzoraka velikom brzinom (primjerice, 44100 uzoraka u sekundi je standardna brzina pri izvođenju CD-a)
- usporedba: audio uzorci poput piksela, a `sf::SoundBuffer` kao `sf::Texture`



https://manual.audacityteam.org/man/digital_audio.html

Učitavanje u spremnik zvuka iz datoteke

- **`loadFromFile`** funkcija:

```
bool loadFromFile(const std::string &filename)
```

- prima putanju do datoteke i vraća `true` ako učitavanje uspjelo (inače vraća `false`)
- zbog problema oko licenciranja **ne podržava mp3 format**

Podržani formati:

- WAV (*Waveform Audio File Format*) s PCM kodiranjem (kakvo većina WAV-ova i koristi)
- OGG/Vorbis
- FLAC (*Free Lossless Audio Codec*)
- za FLAC i WAV podržane rezolucije su 8, 16, 24 i 32 bita (više ovdje: link)

Reprodukcija zvuka nakon što je spremnik spreman

- nakon što su audio podaci učitani u spremnik, koristimo instancu klase `sf::Sound` kako bi ih reproducirali
- spremnik se postavlja pomoću funkcije:

```
void sf::Sound::setBuffer(const SoundBuffer& spr)
```

- `void sf::Sound::play()` - pokreće reproduciranje zvuka

Primjer.

```
sf::SoundBuffer buffer;
buffer.loadFromFile("nekiZvuk.wav"));
sf::Sound zvuk;
zvuk.setBuffer(buffer);
zvuk.play();
```

Dodavanje zvuka u igru kad zmija jede jabuku

- dodamo u datoteku `Zmija.h`:

```
#include <SFML/Audio.hpp>
...
class Svijet {
    ...
    private:
        sf::SoundBuffer audioSpremnik;
        sf::Sound efektJabuke;
        ...
};
```

Dodavanje zvuka u igru kad zmija jede jabuku (nastavak)

- u konstruktoru učitamo datoteku `"jabuka-efekt.wav"` (može se preuzeti na web-stranici kolegija) - zvuk reproduciramo svaki put kad zmija pojede neku jabuku

```
Svijet::Svijet(...)   :   ...   {
    audioSpremnik.loadFromFile("jabuka-efekt.wav");
    efektJabuke.setBuffer(audioSpremnik);
    ...
}

void Svijet::Update(Zmija& igrac) {
    if(igrac.KoordinateGlave() == jabukaKoord) {
        efektJabuke.play();
        ...
}
```

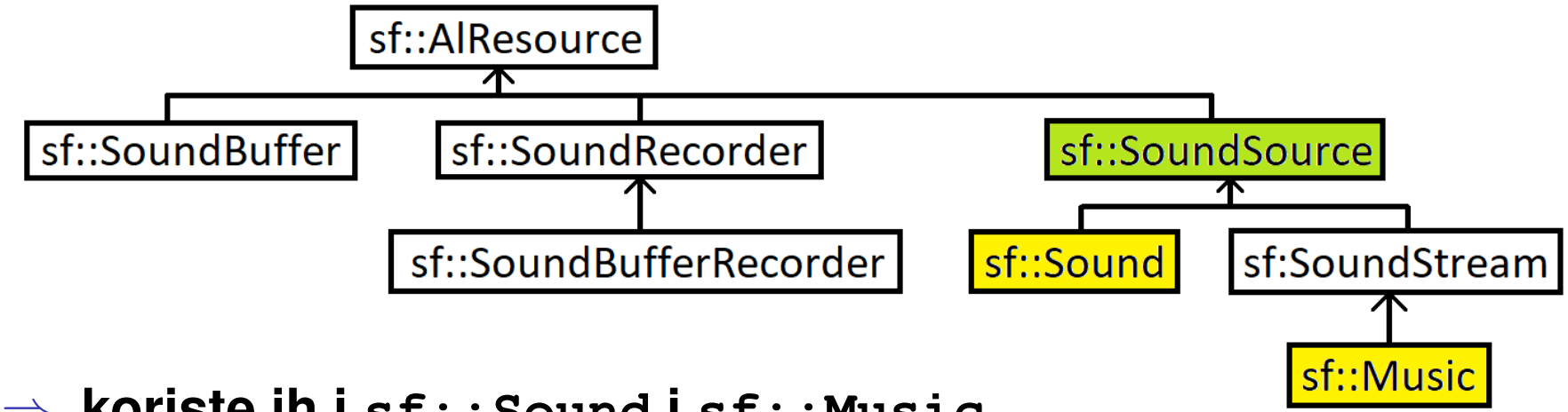
SFML koristi dretve pri reprodukciji zvuka/glazbe

- možemo uočiti da se prethodni zvuk reproducira do kraja iako je zmija nastavila dalje (tj. kod se normalno dalje izvršava)
- zvukovi (i glazba) se reproduciraju u **odvojenim dretvama**

$\Rightarrow$ nakon poziva `play()` možemo dalje normalno svašta raditi (osim naravno uništiti taj zvuk ili njegove audio podatke) - zvuk se reproducira dok ne završi ili se eksplicitno zaustavi

Upravljanje reproduciranjem

- funkcije `play`, `pause` i `stop` implementira klasa **`sf::SoundSource`** (bazna klasa za postavke zvuka)



⇒ **koriste ih i `sf::Sound` i `sf::Music`**

- **`play()`** - započinje ili nastavlja reprodukciju zvuka
- **`pause()`** - pauzira reprodukciju zvuka (ako se reproducirao - inače ne radi ništa)
- **`stop()`** - zaustavlja reprodukciju zvuka (ako se reproducirao ili je bio pauziran - za već zaustavljeni ne radi ništa) - također za razliku od `pause()` vraća poziciju reprodukcije na početak

Klasa `sf::Music`

- nasljeđuje klasu `sf::SoundStream` - *stream* podataka izravno iz izvora (za razliku od `sf::Sound` nema prvo učitavanje pa onda reproduciranje audio zapisa!)

Primjer.

```
sf::Music glazba;
if(!glazba.openFromFile("pjesma.ogg"));
    return -1; //doslo do greske
glazba.play();
```

Napomena. Za razliku od ostalih SFML resursa nema funkciju `loadFromFile` nego `opnFromFile` jer ne učita podatke iz datoteke nego ih čita iz datoteke pri reproduciranju.
$\Rightarrow$ Audio datoteka mora biti dostupna dok reprodukcija ne završi!

Reprodukcija pozadinske glazbe za igru Zmija

- link za preuzimanje datoteke "primjer.wav" (18.7MB): link

```
...
#include <SFML/Audio.hpp>

class Igra {
    ...
    private:
        sf::Music music;
        ...
};
...
Igra::Igra() :  ...  {
    ...
    music.openFromFile("primjer.wav");
    music.setLoop(true);
    music.play();
}
```

Neke mogućnosti (klasa `sf::SoundStream`)

Neke funkcije klase `sf::SoundStream` (klasa `sf::Music` ju nasljeđuje $\Rightarrow$ ona isto može koristiti te funkcije):

`void sf::SoundStream::setLoop(bool loop)`

- funkcija korištena na prethodnom slajdu - nakon dolaska do kraja *streama*, treba li se on opet pokrenuti ispočetka (*loop* način)

`bool sf::SoundStream::getLoop() const`

- nalazi li se *stream* u *loop* načinu

`void sf::SoundStream::setPlayingOffset(Time timeOffset)`

- postavlja poziciju reproduciranja (od početka *streama*) za *stream* koji se izvodi ili koji je pauziran (na zaustavljeni nema utjecaj)
- trenutnu poziciju streama dobivamo funkcijom:
  `Time sf::SoundStream::getPlayingOffset() const`

Primjer za `setPlayingOffset`

```
Igra::Igra() :  ...  {
    ...
    music.openFromFile("primjer.wav");
    music.setLoop(true);
    music.play();
    music.setPlayingOffset(sf::seconds(4.f));
}
```

Još jedan primjer

Primjer. Što bismo postigli sljedećom promjenom u kodu:

```
void Igra::update() {
    ...
    if(vrijeme.asSeconds() >= vrijemeIteracije) {
        ...
        if(zmija.JelIzgubio()) {
            music.setPlayingOffset(sf::seconds(4.f));
            zmija.Reset();
        }
    }
}
```

Postavljanje glasnoće

`void sf::SoundSource::setVolume(float volume)`

- `volume` je vrijednost između 0 (*mute*) i 100 (puna glasnoća)
- *defaultna* vrijednost glasnoće je 100 ($\Rightarrow$ ne možemo imati glasniji audio od početne glasnoće)

Primjer. U igri *Zmija* smanjimo glasnoću pozadinske glazbe (kako bi se jače čuo efekt koji imamo kad zmija jede jabuku):

```
Igra::Igra() : ... {
    music.openFromFile("primjer.wav");
    music.setLoop(true);
    music.play();
    music.setPlayingOffset(sf::seconds(4.f));
    music.setVolume(20.f);
}
```

Pitch - promjena frekvencije

`void sf::SoundSource::setPitch(float pitch)`

- `pitch` - predstavlja osnovnu frekvenciju zvuka koju percipiramo
- vrijednost 1 je *defaultna* vrijednost
- promjena utječe i na brzinu reproduciranja

Primjer. U igri *Zmija* probati ovako i s vrijednosti `1.5f`:

```
Igra::Igra() : ... {
    music.openFromFile("primjer.wav");
    music.setLoop(true);
    music.play();
    music.setPlayingOffset(sf::seconds(4.f));
    music.setVolume(20.f);
    music.setPitch(0.6f);
}
```