Template ~~Meta~~ Music Programing

or `constexpr` Composition

Ashley Roll

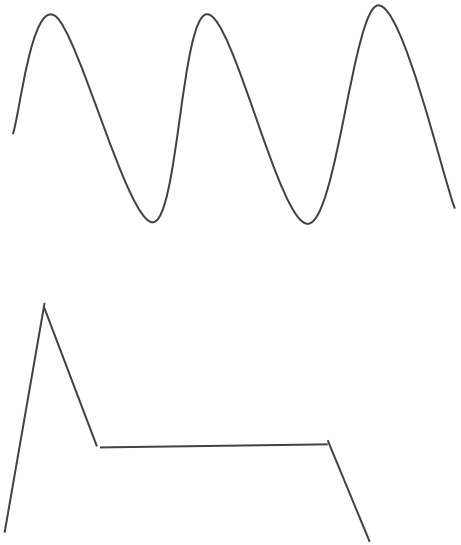
How to make a Synthesiser

- Oscillator
- Envelope Generator
- Notes and Instrument
- Sequencer and Parser
- WAV generator

https://github.com/AshleyRoll/template_music_programming

Oscillator and Envelope Generator

- Simple std::sin() wave generator
- Envelope ramps volume over time
  - Attack - initial impulse to high volume
  - Decay - reduce to normal level
  - Sustain - same level until note released
  - Release - fade to zero



Notes and Instrument

- Notes define the pitch (frequency) and duration
- An Instrument takes notes and renders them using its settings (oscillator and envelope generator)

```
static constexpr sample_rate Rate{ 8'000 };

sin_synth<Rate> synth{
 envelope{ 0.005_sec, 0.0_dBfs, 0.02_sec, -3.0_dBfs, 0.005_sec },
 -1.0_dBfs
};

synth.play_note("A4"_note, (0.1_sec).to_samples(Rate), (1.5_sec).to_samples(Rate));
```

Sequencer and Parser

- Sequencer triggers the instrument at the right time for each note
- Parses the Music Notation and generates a list of notes
- WAV file renderer makes blocks of audio for instrument to write into; combines into one giant std::array<std::byte>

```
sequencer sequencer{ synth };

static constexpr auto music_length = parse_music_length(musicSource);
sequencer.parse_music(musicSource);

wav_renderer_mono<Rate, music_length> wav{};
wav.render(sequencer);
```

Music Notation

```
auto musicSource = [] -> tmp::music {
 return tmp::music{ tmp::beats_per_minute{ 120 },
   R"(
    | 1                   | 2                   | 3                   | 4                   |
    |----+----+----+----|----+----+----+----|----+----+----+----|----+----+----+----|
G#4|    :    :    :    |    :    :    :    |    :    :    :    |    :    :    :    |
G4 |    :    :    :    |    :    :    :    |    :    :    :    |    :    :    :    |
F#4|    :    :    :    |    :    :    :    |    :    :    :    |    :    :    :    |
F4 |    :## #:##  :    |    :    :    :    |    :    :    :    |    :    :    :    |
E4 |    :    :    :    |    :    :    :    |    :    :    :    |    :    :    :    |
D#4|    :    :  ##:##  |    :## #:##  :    |    :    :##  :    |    :##  :    :    | ETC…
D4 |    :    :    :    |    :    :    :    |    :    :    :    |    :    :    :    |
C#4|  # :    :    :    |    :    :  ##:##  |  # :####:    :    |    :    :####:    |
C4 |    :    :    :    |  # :    :    :    |    :    :   #:    |    :    :    :    |
B3 |    :    :    :    |    :    :    :    |    :    :    :    |    :    :    :    |
A#3| # #:    :    :    | # #:    :    :    | # #:    :  # :##  |    :    :    :    |
A3 |    :    :    :    |    :    :    :    |    :    :    :    |    :    :    :    |
G#3|#   :    :    :    |#   :    :    :    |#   :    :    :  ##|  ##:    :    :    |
    |----+----+----+----|----+----+----+----|----+----+----+----|----+----+----+----|
)" };
};
```

All together

```
[[gnu::section(".wavefile"), gnu::used]]
constinit auto const WaveData = [] {
 using namespace tmp;
 using namespace tmp::literals;
 using namespace tmp::instruments;

 static constexpr sample_rate Rate{ 8'000 };

 sin synth<Rate> synth{
   envelope{ 0.005_sec, 0.0_dBfs, 0.02_sec, -3.0_dBfs, 0.005_sec },
   -1.0_dBfs
 };
 sequencer sequencer{ synth };

 static constexpr auto music length = parse_music_length(musicSource);
 sequencer.parse_music(musicSource);

 wav_renderer_mono<Rate, music_length> wav{};

 wav.render(sequencer);
 return wav.data;
}();
```

Extract WAV file

```
gcc --std=c++23 -O3 \
    -fconstexpr-ops-limit=9999999999999 \
    -c song.cpp -o song.o

objcopy --only-section=.wavefile \
    -O binary song.o song.wav
```