语法规则

一首音乐称为一个“Track”（音轨）。一个 Track 一一对应一个音频文件。一个 Track 是由一个已有声音（例如钢琴声）通过根据乐谱调音得来的，乐谱用 Score 类型存储。Score 就是小节（Bar）的序列，Bar 就是音符（Note）的序列。默认情况下，已有声音的文件播放一遍作为一拍。

下面列出了所有 SysY Live 支持的函数。

1. Note 相关
   Note 是 immutable 的。
   - Note x(syllablename = "syllablename", duration = "u/v"); 新建一个音名为 syllablename 的音符，拍数 u/v 拍。例如：Note x(syllablename = ":b6", duration = "2/3"); 创建一个降 la，升八度，长 2/3 拍。其中前缀 `:` 表示升八度，后缀 `$` 表示降八度，`b` 表示降半音，`#` 表示升半音。所有数都是整数（下同）。
2. Bar 相关
   Bar 是 mutable 的。
   - Bar x; 新建一个空小节。
   - Bar x(y); 新建一个小节，与 y 内容相同（是 y 的复制而非引用）。
   - Bar x(syllablename = "1 {1 {1 #5}} b5 - {:#1 b5$}"); 新建一个小节，音符如 syllablename 所示。单个音符表示方法与 Note 相同，可以用 - 或 _ 表示延续上个音符，用 {} 表示把当前音符的时值等距分割。
   - x.inc_bar_pitch(semitones); 小节整体升 semitones 个半音（可以是负数）。
   - x.set_duration(len_ms); 小节速度调整为使得刚好总共播放 len_ms 毫秒。
   - x.set_bar_bpm(bpm); 小节速度调整为 bpm beats per minute。
   - x.push_note(note); 小节末尾加入音符 note。
3. Score 相关
   Score 是 mutable 的。
   - Score x; 新建一个空乐谱。
   - Score x(y); 从乐谱 y 复制一个乐谱，具体同 Bar。
   - Score x(syllablename = "1 {1 {#1 b5}} 5 {1 5} | 6. - :1 - "); 根据谱建立乐谱，| 表示小节线。
   - x.inc_score_pitch(semitones); 同 Bar。

- x.set_duration(len_ms); 同 Bar。
- x.set_score_bpm(bpm); 同 Bar。
- x.push_bar(bar); 在乐谱末尾加一个小节，内容为 bar。
- x.append(score); 在乐谱末尾复制一个 score 乐谱的内容。
- x.replace_bar(k, bar); 将乐谱的第 k 个小节（从 0 开始）换为小节 bar。
- x.sing(name1, name2, samplerate, bytes, channels); 利用 name1 数组存放的字符串作为音源文件名（必须是 .wav），它的采样率为 samplerate Hz、bytes kb/s，channels 声道，根据乐谱 x，生成音频。音频名字为 name2。

4. Track 相关

   Track 是 mutable 的。

   - Track x; 新建一段空声音。
   - Track x(y); 将声音绑定上 y 字符串为名字的音源。
   - Track x(y, z); 将 Track y 的声音复制一份，到 z 字符串作为名字的音源。
   - x.append_track(track); 在 x 音源文件的末尾加上 y 音源。
   - x.append_silence(offset_ms); 在 x 音源文件的末尾加上 offset_ms 毫秒的沉默。
   - x.stack(y); 把 x 音源和 y 音源进行混音，得到 x 音源。
   - x.setvol(fz, fm); 把 x 音源的声音大小调整为原来的 fz/fm 倍。

用法

示例代码

下列代码是 hello.c 可以存放的内容。在给定 `1.wav` 后，会自动生成 `2.wav` 作为结果。笔者自己生成的 `1.wav` `2.wav` 供参考，放于代码中了。

```
int main() {
        Score x(syllablename = ":1 :1 :5 :5 | :6 :6 :5 - | :4 :4 :3 :3 | :2 :2 :1 - ");
    Score x1(syllablename = "1 - - -  | 4 - - - | 5$ - 7$ -  | 2 5 1 - ");
    Score x2(syllablename = "{::1 :5} {::1 ::3} {::5 ::1} {::3 ::5} | {::6 ::4} {:::1 ::6} {::5 ::
        int a[10] = {49, 46, 119, 97, 118}; // 1.wav
        int b[10] = {50, 46, 119, 97, 118}; // 2.wav
        int c[10] = {51, 46, 119, 97, 118}; // 3.wav
        int d[10] = {52, 46, 119, 97, 118}; // 4.wav
        x.sing(a, b, 44100, 16, 2);
    x1.sing(a, c, 44100, 16, 2);
    x2.sing(a, d, 44100, 16, 2);
    Track t1(b);
    Track t2(c);
    Track t3(d);
    t1.stack(t2);
    t1.stack(t3);
    return 0;
}
```

结果：利用钢琴弹奏了一曲三个音轨的小星星。

运行方法

本项目只能在 Windows 上运行。需要把所有音源文件（例如 1.wav）放于工作目录下。同时，需要安装课程报告中的一切依赖，并加入环境变量。

具体用法：

1. 用 -llvm 参数编译 hello.cpp 至 hello.llvm。
2. 运行目录下的 test.cpp，执行编译 imp.cpp 以及链接。（或者复制其中的命令手动运行）

运行实例视频已经附在提交的文件里了。