

Midi 音乐示例

vcg

2022 年 6 月 1 日

1 midi 音乐

两个功能

读入 txt 文件，转换成 midi 格式数据，通过 `midiOutShortMsg` 函数进行播放
用键盘进行演奏

三个主要类:

Music: 乐谱类。由 MusicNote 音符元素构成

MusicNote, 音符类。

MidiPlayer, Midi 播放类, 负责 midi 设备的打开关闭, Music 播放, 以及弹奏
测试函数

PlayMusic: music 播放函数

PlayPiano: 弹奏函数

Source.cpp

```

1  #include <Windows.h>
2  #include <iostream>
3  #include <fstream>
4  #include <map>
5  #include <string>
6  #include <vector>
7  #include <sstream> //字符串流
8
9
10 #pragma comment(lib, "winmm.lib")
11
12 //音阶范围 0x0-7f
13 std::map<std::string, int> c_midis = {
14     {"C7", 96}, {"C7s", 97}, {"D7", 98}, {"D7s", 99}, {"E7", 100}, {"F7", 101}, {"
15         F7s", 102}, {"G7", 103}, {"G7s", 104}, {"A7", 105}, {"A7s", 106}, {"B7",
16         107},
17     {"C6", 84}, {"C6s", 85}, {"D6", 86}, {"D6s", 87}, {"E6", 88}, {"F6", 89}, {"F6s
18         ", 90}, {"G6", 91}, {"G6s", 92}, {"A6", 93}, {"A6s", 94}, {"B6", 95},
19     {"C5", 72}, {"C5s", 73}, {"D5", 74}, {"D5s", 75}, {"E5", 76}, {"F5", 77}, {"F5s
20         ", 78}, {"G5", 79}, {"G5s", 80}, {"A5", 81}, {"A5s", 82}, {"B5", 83},
21     {"C4", 60}, {"C4s", 61}, {"D4", 62}, {"D4s", 63}, {"E4", 64}, {"F4", 65}, {"F4s
22         ", 66}, {"G4", 67}, {"G4s", 68}, {"A4", 69}, {"A4s", 70}, {"B4", 71},
23     {"C3", 48}, {"C3s", 49}, {"D3", 50}, {"D3s", 51}, {"E3", 52}, {"F3", 53}, {"F3s
24         ", 54}, {"G3", 55}, {"G3s", 56}, {"A3", 57}, {"A3s", 58}, {"B3", 59},
25     {"C2", 36}, {"C2s", 37}, {"D2", 38}, {"D2s", 39}, {"E2", 40}, {"F2", 41}, {"F2s
26         ", 42}, {"G2", 43}, {"G2s", 44}, {"A2", 45}, {"A2s", 46}, {"B2", 47},
27     {"C1", 24}, {"C1s", 25}, {"D1", 26}, {"D1s", 27}, {"E1", 28}, {"F1", 29}, {"F1s
28         ", 30}, {"G1", 31}, {"G1s", 32}, {"A1", 33}, {"A1s", 34}, {"B1", 35},
29     {"C0", 12}, {"C0s", 13}, {"D0", 14}, {"D0s", 15}, {"E0", 16}, {"F0", 17}, {"F0s
30         ", 18}, {"G0", 19}, {"G0s", 20}, {"A0", 21}, {"A0s", 22}, {"B0", 23},
31 };
32
33
34
35 struct MelodicSound {
36     std::string name;
37     int number;
38 };
39
40
41 const int c_max_sound = 5;
42 MelodicSound c_melodic_sounds[c_max_sound] = { {"Grand Piano", 0}, {"Music box" ,
43     10}, {"Electric Guitar (jazz)", 26},
44     {"Steel Drums", 114}, {"Woodblock", 115} };
45
46
47 //基本音符, numbered musical notation
48 class MusicNote
49 {
50     friend class Music;

```

```

39     friend class MidiPlayer;
40     friend std::istream& operator>>(std::istream& is, MusicNote& note)
41     {
42         is >> note.octave_jianpu >> note.notation_jianpu >> note.duration_jianpu;
43         note.UpdateNotation();
44         return is;
45     }
46     friend std::ostream& operator<<(std::ostream& os, const MusicNote& note)
47     {
48         if (note.commnet == "")//非注释行
49             os << note.octave_jianpu << " " << note.notation_jianpu << " " << note.
                duration_jianpu << "\n";
50         else
51             os << "\n" << note.commnet << "\n";
52         return os;
53     }
54 private:
55     //音高, 0, 1, 2代表上面0, 1, 2个高音点, -1, -2代表, 底下低音点个数
56     int octave_jianpu;
57     int notation_jianpu;        //1,2,3,4,5,6
58     double duration_jianpu;    //拍为单位
59
60     int midi_pitch;            //midi表示的值
61     std::string midi_msg;      //音符表示, C4 D4 E4 F4 A4 B4
62
63     //注释信息, 文件中用#开头
64     std::string commnet;
65 public:
66     MusicNote()
67     {
68         octave_jianpu = 0;
69         notation_jianpu = 0;
70         duration_jianpu = 0.0;
71     }
72
73     //简谱信息转换成频率, 时长
74     void UpdateNotation()
75     {
76         if (notation_jianpu == 0)
77         {
78             midi_pitch = 0;
79             return;
80         }
81
82         std::string arrNotes[] = { "C", "D", "E", "F", "G", "A", "B" };
83         midi_msg = arrNotes[notation_jianpu - 1] + std::to_string(octave_jianpu +
            4);
84         midi_pitch = c_midis[midi_msg];
85     }
86

```

```

87     void SetComment(const std::string& str) { commnet = str; }
88 };
89
90
91 class Music
92 {
93     friend class MidiPlayer;
94 private:
95     std::vector<MusicNote> notes;
96 public:
97     void LoadFromFile(const std::string& file)
98     {
99         std::ifstream infile(file);
100         if (!infile.is_open())return;
101
102         std::string content;
103         while (std::getline(infile, content))
104         {
105             if (content[0] == '#')
106             { // 注释, 提取信息
107                 std::string msg(content.c_str() + 1);
108                 MusicNote note;
109                 note.SetComment(msg);
110                 notes.push_back(note);
111                 continue;
112             }
113             else if (content == "")
114             { // 掠过空行
115                 continue;
116             }
117
118             std::stringstream ss(content);
119
120             MusicNote note;
121             ss >> note;
122             notes.push_back(note);
123         }
124     }
125 };
126
127
128 class MidiPlayer
129 {
130 private:
131     // 设备
132     HMIDIOUT handle;
133     HANDLE hIn;
134
135     // 音色转换编号
136     int program_number;

```

```

137 private:
138     void SetMelodicSound();
139     void SetProgramNumber(int numbe) { program_number = numbe; }
140
141     DWORD CodeOn(int pitch, int velocity = 100, int channel = 0)
142     {
143         return (channel + 0x90) + (pitch << 8) + (velocity << 16);
144     }
145
146     DWORD CodeOff(int pitch, int velocity = 100, int channel = 0)
147     {
148         return (channel + 0x80) + (pitch << 8) + (velocity << 16);
149     }
150
151     //change instrument,
152     inline DWORD ChangeInstrument(int number = 0)
153     {
154         return (0 + 0xC0) + (number << 8); //115,25,3
155     }
156 public:
157     MidiPlayer()
158     {
159         program_number = 0;
160
161         midiOutOpen(&handle, 0, 0, 0, CALLBACK_NULL);
162         hIn = GetStdHandle(STD_INPUT_HANDLE);
163     }
164     ~MidiPlayer()
165     {
166         if (hIn)
167             CloseHandle(hIn); // 关闭标准输入设备句柄
168         if (handle)
169             midiOutClose(handle);
170     }
171
172     void PlayNote(int key)
173     {
174         std::map<char, std::string> vMsg = {
175             {'Z', "C3"},{'X', "D3"},{'C', "E3"}, {'V', "F3"},{'B', "G3"},{'N', "A3"}
176             },{'M', "B3"},
177             {'A', "C4"},{'S', "D4"},{'D', "E4"}, {'F', "F4"},{'G', "G4"},{'H', "A4"}
178             },{'J', "B4"},
179             {'Q', "C5"},{'W', "D5"},{'E', "E5"}, {'R', "F5"},{'T', "G5"},{'Y', "A5"}
180             },{'U', "B5"},
181         };
182
183         char ckey = 'A' - 65 + key;
184
185         std::string msg;
186         if (vMsg.find(ckey) == vMsg.end()) return;

```

```

184
185     msg = vMsg[ckey];
186
187     midiOutShortMsg(handle, CodeOn(c_midis[msg]));
188     std::cout << msg << " ";
189
190     std::vector<std::string> pressedKey;
191     pressedKey.push_back(msg);
192
193     static int count = 0;
194     count++;
195     if (count % 10 == 0) std::cout << std::endl;
196 }
197
198 void SetMelodicSound(int index)
199 {
200     program_number = c_melodic_sounds[index].number;
201     if (handle) midiOutShortMsg(handle, ChangeInstrument(program_number));
202 }
203
204 void Play(const Music& music)
205 {
206     for (auto itr = music.notes.begin(); itr != music.notes.end(); itr++)
207     {
208         const MusicNote& note = *itr;
209         std::cout << note;
210         midiOutShortMsg(handle, CodeOn(note.midi_pitch, 100));
211         Sleep(DWORD(note.duration_jianpu * 500)); //125*4
212     }
213 }
214
215 };
216
217 MidiPlayer g_playPiano;
218
219
220 void PlayPiano()
221 {
222     HANDLE hIn = GetStdHandle(STD_INPUT_HANDLE);
223     std::cout << "钢琴已开启, 敲击键盘Q-U,A-J,Z-M, ESC键退出\n";
224
225     int keyPressed[1024] = { 0 };
226     while (true)
227     {
228         DWORD dwRes = 0;
229         INPUT_RECORD keyRec;
230         ReadConsoleInput(hIn, &keyRec, 1, &dwRes);
231         if (keyRec.EventType == KEY_EVENT)
232         {

```

```

233         if (keyRec.Event.KeyEvent.wVirtualKeyCode == VK_ESCAPE) break; // 按ESC
                键时退出
234
235         if (keyRec.Event.KeyEvent.bKeyDown)
236         {
237             char ch = keyRec.Event.KeyEvent.uChar.AsciiChar;
238
239             char key = ch;
240             if (key >= 'a' && key <= 'z') key = key + 'A' - 'a';
241             if (key >= 'A' && key <= 'Z')
242             {
243                 g_playPiano.PlayNote(key);
244             }
245             keyPressed[ch] = true;
246         }
247         else
248         { // 按键起
249             char ch = keyRec.Event.KeyEvent.uChar.AsciiChar;
250             keyPressed[ch] = false;
251         }
252     }
253 }
254
255 CloseHandle(hIn); // 关闭标准输入设备句柄
256 }
257
258 void PlayMusic(const std::string& name)
259 {
260     Music music;
261     music.LoadFromFile(name);
262     g_playPiano.Play(music);
263 }
264
265
266 void SelectSoundNumber()
267 {
268     std::cout << "输入一个整数, 选择乐器~" << std::endl;
269     int sound_number;
270     std::cin >> sound_number;
271
272     sound_number = sound_number % c_max_sound;
273     std::cout << "您选了 " << c_melodic_sounds[sound_number].name << std::endl;
274     g_playPiano.SetMelodicSound(sound_number);
275 }
276
277
278 int main()
279 {
280     SetConsoleOutputCP(CP_UTF8);
281

```

```
282     std::string menu = "a: 播放New boy, b: 播放春光美, c: 键盘电子琴。输入选择";
283     std::cout << menu << std::endl;
284     switch (std::cin.get())
285     {
286     case 'a':
287         SelectSoundNumber();
288         PlayMusic("music/newboy.txt");
289         break;
290     case 'b':
291         SelectSoundNumber();
292         PlayMusic("music/spring2.txt");
293         break;
294     case 'c':
295         SelectSoundNumber();
296         PlayPiano();
297         break;
298     }
299
300     return 0;
301 }
```