程序设计实习 Qt 作业报告

曹彧 Y. Cao[1], 朱丽烨 L.-Y. Zhu[2], and 王睿妍 R.-Y. Wang[3]

[1] 北京大学信息科学技术学院, 2300013198@stu.pku.edu.cn

[2] 北京大学元培学院, 2300017848@stu.pku.edu.cn

[3] 北京大学信息科学技术学院, 2300013234@stu.pku.edu.cn

目录

1 程序功能介绍

Doremi 是一款音乐编辑器，可以实现对音乐的自由创作和编辑，也可以进行文件存储和导入。9 条音轨，128 款乐器，在创作者的指尖演绎出华美的乐章。

1.1 创作功能

演奏者可以尽情发挥创造力，选择 1 至 9 条音轨进行创作。在每条音轨上，他们都有机会从精心挑选的 128 种乐器音色中选择最适合的音色。这种选择不仅限于传统乐器，还包括了现代合成音色，为创作提供了无限的可能性。

一旦选定了音色，演奏者只需点击录制按钮，便可以开始演奏并记录下每一个音符。钢琴键与键盘上的键一一对应，使得演奏者能够迅速上手，专注于创作本身。

1.2 播放功能

在播放环节，每条音轨都享有高度的自主性。演奏者可以单独播放某一条音轨，也可以随时更换音效、调节音量，甚至将某条音轨静音。当所有音轨同时播放时，它们会和谐地交织在一起，形成丰富的和声效果。

程序还提供了整体升降调和调节速度的功能，让演奏者能够根据需要调整歌曲的调性和节奏。此外，当前播放音符所在的小节数会实时显示在界面上方，为演奏者提供清晰的进度参考。

1.3 文件存储功能

为了让创作成果得以保存和分享，程序特别设计了文件存储功能。一旦点击“开始创作”，新创建的谱子就会被自动保存至指定文件夹中。当需要继续编辑或与他人分享时，只需在“打开文件”的文本框里输入相应乐谱名，即可轻松调用并继续更新。这种便捷的文件管理方式，让演奏者能够随时随地回顾和分享自己的创作成果。

1.4 教弹功能

程序特别设计了一个教弹功能，为学习者提供了一个直观且易于跟随的学习平台。程序内置了四首经典且受欢迎的乐曲：《小星星》、《打上花火》、《我们都拥有海洋》以及《fight song》。每首歌曲都有其独特的旋律和风格，旨在满足不同学习者的喜好和需求。

当用户在程序中选择并勾选教弹功能后，系统会立即引导用户进入教弹主界面。即将播放的主旋律音符会被特别标记出来，这种视觉上的提示能够帮助演奏者迅速找到需要按下的键位。一旦按下正确的键位，系统便会立即播放该音符及其伴奏。

整个教弹过程会按照乐谱的顺序依次循环进行，从乐曲的第一个音符开始，一直到最后一个音符结束。在这个过程中，演奏者可以逐步熟悉和掌握乐曲的旋律和节奏，从而更加自信地演奏出完整的作品。

2 项目各模块与类设计细节

如何充分应用面向对象程序设计的思想，实现我们想要达到的功能呢？下面将逐一介绍项目各个模块和类的设计细节。

2.1 音乐的表示

制作音乐编辑器的第一步，是将“音乐”通过合适的方式抽象成代码语言。通过什么样的方式表示一首歌曲、用什么样的方式把乐曲演奏出来，是我们需要解决的首要问题。

通过上网查询，我们发现 Windows.h 中的 midi 库是一个音乐演奏库，自带 128 种音色，通过 midiOut 短消息与程序交互，能够播放不同的音高，这为我们实现乐曲演奏提供了极大的便利。在该库中，音高由数字表示，21 到 108 代表 A0 到 C8 全部 88 个音，同时 1009 代表连续，1013 代表休止。因此我们定义了枚举类 Pitch，用字母表示音高。

多个音高构成音符，多个音符构成音轨，多条音轨构成音乐。这是我们表示音乐的核心框架。

2.1.1 音符：class v_spo

音符是音乐最基本的组成部分，一个音符可以是单音，也可以由多音构成（和弦），同时还可以是特殊音符（如休止符、连接符等）。

我们将音符抽象为 v_spo 类。一个音符最多可以容纳 4 个音高，默认为连接符，同时每个音符还有对应的音量，实现了对音符强弱程度的独立控制。同时以 v_spo 为基类，我们派生出 stop_spo 和 rest_spo，分别表示休止符和连续符。

2.1.2 音轨：class Channel

音轨，顾名思义，即装载有音符的轨道，相当于五线谱中的一行、合奏中的一个声部。如果仅使用一个简单的音轨数组来代替轨道是不够的，而且不方便统一管理。由此我们定义音轨类 channel。

channel 中存有音符数组，同时存有音轨的编号、乐器的类型、强弱度，可以实现对音轨中所有音符音量的统一调整、在指定的位置添加删除音符、改变该音轨对应的乐器等等功能。我们将 16 个音符定义为一个小节，每小节用音符填充满，没有音符的地方默认填充连接符。

2.1.3 乐曲：class Music

有了如上铺垫后，定义 Music 类也变得十分方便。

Music 中存有乐曲名、速度、音符数、长度，以及一个音轨数组。当接受到播放、修改等命令时，Music 如同“总管”，将任务分配给“下属”音轨，再进一步落实到“基层”音符，实现了对音乐的逐级管理。

2.2 播放器的表示

实现了 Music 的表示后，下一步就需考虑音乐与 midi 库之间的交互方式。如果直接交互可能略显混乱，于是我们定义了交互的“中介”mediaplayer，联系底层的 midi 二进制短消息和上层的 Music 音乐类。

Mediaplayer 类是整个乐曲的“总指挥”，实现了对乐曲播放的控制。它存有音乐指针及当前的播放位置，可以命令乐曲暂停、继续播放，调节播放的速度，打开关闭节拍器，同时可以修改 Music 中的音符。

当收到开始播放的信号后，计时器会启动。根据当前 Music 的设定速度，计时器会调节时长，计时器到时则会发送“播放下一音符”的信号并重启。如果 MediaPlayer 收到了播放下一音符的信号，则会将当前的所有音符通过 Midi 短消息传输到声音输出设备并发声，随后将当前的播放位置向后推进一个。当曲终或者暂停时，MediaPlayer 便会控制计时器暂停。

实现 Mediaplayer 后，我们已经能用代码写一些最基本的乐曲并播放。底层框架搭建的阶段顺利结束。

2.3 文件存储

接着我们考虑音乐的存储问题。由于音乐类是我们自定义的，我们没有找到合适的方式将 Midi 播放的乐曲转换成 mp3 或 wav 格式，因此我们使用二进制文件存储音乐。

二进制文件易于程序读取，难以修改，具有比文本文件更强的稳定性和通用性。一首乐曲用一个文件单独存放，能够实现对乐曲持续编辑。我们在 Channel 中定义了单独 channel 读写的方式，并在读写 Music 的函数中调用它们，这也体现了 Music 多层级管理的便捷性。

2.4 可视化设计

底层逻辑完善后，我们开始专注于界面设计，进行 UI 界面的美化工作。

2.4.1 开始界面与文件选择界面

开始界面与文件选择见面的设计较为简单。文件选择界面能够完成音乐编辑的准备工作，其中创建新的乐曲即为创建新的 Music 类对象，打开模板乐曲和已经存储的乐曲会调用读取文件的模块。当“确认”键点击时，程序会打开 createpage 设计界面并将乐曲传输到 createpage 中。

2.4.2 音乐设计主界面

Createpage 是音乐设计的主界面。我们把预先设计好的 Mediaplayer 播放器放在这一界面中，并且将上一界面传来的一个 Music 对象放入 Mediaplayer，随时准备音乐的播放和改写工作。

界面上方的操纵台与播放器相联系，播放、停止、节拍器开关、速度调整、升降调都由操控台发出信号，MediaPlayer 接收，调整音乐的播放情况。左上角的文件存储键则与文件读写模块相交互，可存储音乐到特定的位置。

界面的左侧是多条音轨的可视化，由多个 ChannelEdit 控件构成，一个控件控制音乐的一个音轨，将音轨中存有的乐器种类、音量进行可视化，同时静音按钮和 MediaPlayer 之间也通过信号交互，可以实现音轨的静音，有利于音乐编辑。

2.4.3 钢琴界面

Piano 钢琴界面是 ChannelEdit 控件的扩展，一次只负责控制一个音轨。在 Piano 界面中，我们绘制了钢琴键盘，同时还有音量旋钮可以调节弹奏音符的音量，实现了一台模拟“电子琴”。

播放乐曲的过程中，如果该声道遇到了新的音符，这个音符就会由 Mediaplayer 传递过来，通过改变对应音高钢琴键盘的颜色“点亮”正在演奏的音，模拟弹琴的过程。

同时钢琴界面还有弹奏功能。通过处理键盘信息，可以将钢琴键和键盘对应的按键相联系，从而实现在键盘上“弹琴”的功能，按下键盘就可以发出相应的音。

随后我们扩展了录制功能，如果开启录制功能，弹奏的音符则会传到 Music 类的对象之中，在对应的音轨和位置添加音符。这样我们就实现了对乐曲的可视化编辑。

2.5 教弹功能设计

前期的设计逻辑和框架为程序提供了很强的可塑性和兼容性，我们决定扩展现有功能，探索该音乐编辑软件更多的创新点和可能性。

教弹功能的实现就是典型一例。其基本框架逻辑和正常的演奏基本相同，只不过添加了特定的教弹音轨。我们在 MediaPlayer 的基础上重载了 GameMediaPlayer 类，存储使用者指定的教弹音轨，并且重载了“播放下一音符”的函数。如果在播放的时候当前教弹音轨有音，则会将 MediaPlayer 暂停，并点亮键盘上该音符对应的音高（但不发声）。直到使用者在键盘上弹对正确的音 MediaPlayer 才会重启，乐曲继续播放。

2.6 AI 伴奏设计的构想

由于时间原因，我们最后的 AI 伴奏部分尚未完成，但对于 AI 伴奏已经有了大致的构思，下面进行简要介绍。

AI 伴奏和音乐的生成，和自然语言处理之间存在很多的类似之处。我们的构想是运用与自然语言处理相近的方式来实现音乐生成的功能。

首先是调性的判断。调性对于伴奏和弦的选取具有直至关重要的作用，而一首乐曲的调性与各个音出现的频率是密切相关的。因此我们选用类似“词袋模型”的方法，统计乐曲中各个音出现的频率，随后运用朴素贝叶斯模型判断乐曲的调性。

其次是和弦主音的选择。一个小节一般会有 1 到 2 个主和弦，和弦的选取和音高、音符的位置相关，即便是同样的音在不同小节之间重复，最适合的和弦伴奏也是不一样的。因此我们考虑首先通过升降调将乐曲统一为 C 大调，随后利用神经网络进行模型的训练，对小节进行位置编码，使得模型能够通过给定的一段音乐和对应小节位置生成对应的主和弦。有了主和弦，就可以从伴奏库中选取特定风格的伴奏进行搭配。

由于训练数据需要我们手动提取、训练模型的搭建和机器学习的处理运用 Python 语言相较 C++ 更为方便、小组成员的时间有限，我们并没有实际完成这一部分。但合适的数据结构、完善的程序框架，使得这一部分与主程序之间的对接即为方便。我们已经预留好该部分的接口，待后期时间充裕再将这一部分补全。

3 小组成员分工情况

曹彧（组长）：项目统筹、Music 和 Mediaplayer 的建构；辅助组员共同完成各项任务

朱丽烨：可视化界面设计

王睿妍：资源收集、文件存储部分

4 项目总结与反思

在本次大作业中，我们使用 github 进行合作编程。由曹彧同学先搭建好整个程序的框架，并预设好所需功能的接口，小组三人再分别完成分配好的功能，大大提高了工作效率。

不过，由于本项目所使用的 midi 库函数仅限在 windows 系统使用，而小组中曹彧，王睿妍同学使用的都是 mac，在进行功能的合并时出现了系统不兼容的问题。在今后的工作中，我们在编写程序时应该更加注意系统之间的差别，选择更加通用，稳定性更好的算法。

此外，在设计图形界面时，我们尝试使用人工智能进行绘制，但人工智能无法仅根据关键词设计出符合要求的图片，导致我们只能选择搜索图片并下载，造成整个程序图案风格的不一致。在今后的程序设计中，为了设计出更加美观，风格协调统一的软件，我们一方面应学习一些简单的画图软件来绘制独特的按钮，另一方面学习更好的利用人工智能生成更加符合要求的图片。

我们最终的程序可以实现创作，存储，播放音乐以及教弹功能，是一个功能相对齐全的音乐软件。美中不足之处在于后期小组成员参加招生活动，已经计划好的 AI 伴奏功能没能完全实现，只是有了实现思路并进行了数据录入，并未开始训练模型。后期我们将在课后努力将其实现，以后也加强规划，更加重视时间的分配与安排。

本次大作业极大的提高了我们设计程序，代码编写和 QT 应用的能力，更是一次小组合作的宝贵经验，为我们接下来的学习奠定了良好的基础。