摘 要

当计算机技术在 20 世纪迅猛发展时，人们生活水平的提高使得对于精神层面的需求变得越来越强烈，特别是对音乐和多媒体方面的需求。计算机技术与这些需求相融合，不断创造出新的事物，并带来更多的需求。

因此，我为了满足这些需求，为不同的人群开发了一款 MIDI 音乐播放器，它提供了三大功能：音乐可视化、黑乐谱播放和视频录制。这些功能可以满足一般大众、音乐爱好者、音乐教学者和学习者，以及黑乐谱爱好者的需求。通过 MIDI，它可以帮助这些人更加方便、高效地欣赏和学习音乐。

关键词：MIDI 播放器，音乐可视化，黑乐谱，视频录制

Abstract

As computer technology rapidly developed in the 20th century, the increase in people's standard of living led to a growing demand for cultural and artistic needs, particularly in the areas of music and multimedia. The integration of computer technology with these requirements has continuously created new products and generated even more requirements.

Therefore, I have developed a MIDI music player that provides three major features: music visualization, MIDI sheet music playback, and video recording, catering to the needs of the general public, music enthusiasts, music educators and learners, as well as MIDI sheet music enthusiasts. Through MIDI technology, it can help these people to enjoy and learn music more conveniently and efficiently.

Keywords: MIDI player, Music visualization, Black MIDI, Video recording

目 录

1 研究背景

1.1 音乐需求

1.1.1 音乐发展

在上世纪 90 年代，随着 DVD 机、音响、电视机等音乐电子产品的兴起和普及，人们对于音乐的娱乐方式和需求变得更加多元化。人们不再只通过传统的线下方式如音乐会、演唱会、卡拉 OK 等来欣赏音乐，更希望能够在家中或者随时随地便捷地享受音乐所带来的魅力[1]。

1.1.2 计算机音乐与创新

随着信息技术的迅速发展，计算机逐渐走进普通家庭。作为通用的电子设备，计算机具有高水平的数字信息处理能力。因此，越来越多的人开始使用计算机进行音乐创作，并由此衍生出多种计算机音乐的娱乐方式，这些方式也变得越来越多样化。

现在的音乐创作者无需在音乐会现场进行录音，也不必准备录音棚等专业设备来录制和调试声音，而是可以在后期通过计算机合成音乐。电子音乐等计算机生成的音乐形式也具有独特的风格。计算机音乐也越来越多地融入了电影、游戏、综艺节目等其他产业中。可以说，现代音乐已经无法离开计算机[2]。

1.2 音频分析需求

1.2.1 音乐与数学

作为人类古老的娱乐方式，音乐已经存在了上千年。音乐与一般声音的区别在于它可以表达和传递情感，并且它通常包含创作、演奏和聆听三个过程。

人们创作和演奏音乐并不是毫无规则的。音乐作为一种传达情感的媒介，人们创作它所遵从的规律和数学有着密切的关系。人们发现，当不同频率的声音比率更为简单时，其共同发出的声音也更加悦耳。通过不同比率的频率组合，人们发明了五声音阶、十二平均律等音乐规则。

计算机的起源是为了解决数学问题，因此它也可以很好地处理音乐方面的问题。通过设计计算机软件，可以进一步研究和理解音乐与数学之间的关系。

1.2.2 音乐创作与教学

旋律、和弦和节奏是音乐中重要的组成部分。因此，对这三者的分析和理解就显得尤为重要。

在创作音乐的过程中，如果可以通过计算机软件将旋律、和弦和节奏等信息展示出

来，那么创作音乐的效率将大大提高。在音乐教学中，如果有更生动的方式将这些音乐理论知识展现出来，那对于音乐的教学活动显然是很有帮助的[3]。因此，计算机音乐软件的需求也随之而生。

1.3 视频播放需求

1.3.1 可视化解决方案

人类接受信息的方式是多种多样的，而眼睛作为人类最主要的感知器官，接收了绝大部分的信息。尽管音乐作为一种声音只能被耳朵所接受，但如果可以将声音可视化，使人通过不同感觉器官感受音乐，那对于感受音乐的人来说将会是一种独特的体验。

如今市面上越来越多的音频播放软件开始支持音乐可视化，例如 Windows 媒体播放器就可以将音乐以粒子形式随着音乐以变化的颜色、形状和样式展现出来。大多数音乐都是由波形文件记录的，而市面上大多数音乐可视化软件都是基于傅里叶变换将波形函数变换为频谱显示出来的。相较于记录波形的音频文件，直接记录音高频率的 MIDI 文件在音乐可视化方面具有天然优势。但是，大部分 MIDI 音频播放软件却并没有这样的功能，因此开发这样一款软件就显得非常有必要性[4]。

1.3.2 视频娱乐与黑乐谱

如今网络视频行业发展迅速，各大视频网站崛起，由视频展示的诸如综艺、动画、游戏、舞蹈、鬼畜等娱乐形式增多，其中音乐可视化在一小众的黑乐谱文化中得到迅速发展并逐渐流行开来。

黑乐谱可以通过在 MIDI 文件中放置大量的音符，使文件在可视化播放的时候展现出不同颜色的文字，图像甚至视频。由于黑乐谱使用的音符数量极多，在五线谱上看起来是黑色的，如图 1.1 所示，因此得名。同时也因为黑乐谱的音符数量极多，大多数的 MIDI 播放软件无法播放黑乐谱，也无法对其进行可视化。

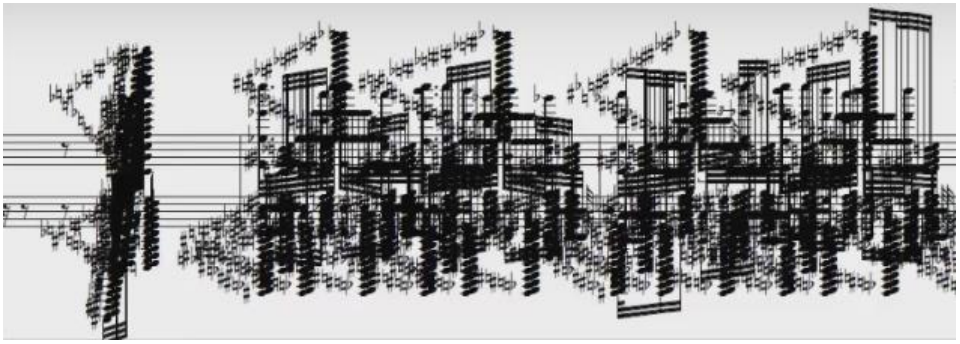

图 1.1 黑乐谱在五线谱上

随着越来越多的人接触和了解黑乐谱，对黑乐谱的播放需求也在不断增加，因此，专门设计和开发这样的黑乐谱软件就变得愈加重要。

2 研究现状

2.1 MIDI 与 MIDI 文件概述

2.1.1 MIDI 概述

MIDI（Musical Instrument Digital Interface，音乐数字接口）是一个工业标准的电子通信协议。通过 MIDI，计算机和各种电子乐器之间可以互相控制、交换消息。通过计算机向 MIDI 演奏设备传输音符和演奏码，可以逼真地模拟乐器的声音[5]。

MIDI 的产生带来了以下优势：

MIDI 使得计算机和各电子乐器之间有共同的语言标准，让设备之间可以互相理解和轻易的链接。

1) MIDI 简化了设备之间的插口和连接线，减少了搭建设备环境的复杂性。

2) MIDI 需要的演奏者的数量更少。通过 MIDI，一位演奏者可以使用计算机同时控制和演奏多台 MIDI 设备。

3) MIDI 是一个开放的简易的标准。创作者和演奏者无需向 MIDI 机构付费，人们可以更专注于创作、编辑、制作高质量的数字音乐。音乐爱好者通过 MIDI 进行交流也变得更加简单。

2.1.2 MIDI 文件概述

MIDI 在计算机方面还定义了标准 MIDI 文件格式（SMF）并带来了 MID 文件。同时，其它公司和组织也根据自身需求对 MIDI 文件格式进行扩展。例如，微软将其与可下载声音格式（DLS）捆绑在资源交换文件格式（RIFF）中，产生的 RMID 文件格式。此外，电子乐器制造商还将其与 Tune 1000 研发的软卡拉 OK 格式相结合，产生了 KAR 文件格式等。

MIDI 文件的出现带来了以下优势[6]：

1) MIDI 文件记录了音符和演奏码等内容，相比于传统的波形的音频模拟信号，所需要的存储空间极小，为前者的千分之一左右。这使得 MIDI 文件在设备上的存储极为廉价和便利，甚至在低端设备和简易芯片中也可以存在，如老年手机的铃声和有声名片贺卡等。MIDI 文件的小尺寸也使得音乐爱好者在互联网上传播 MIDI 文件也变得更快速和便捷。在早期的游戏开发中，受限于硬件容量的限制，MIDI 文件也备受欢迎。

2) 使用 MIDI 格式，音乐创作者也可以随意修改 MIDI 文件内容，使得乐谱的修改和更加方便。音乐爱好者和创作者可以根据自己喜好选择不同音色的音乐设备即时演奏音乐，也可以选择需要的乐器进行演奏。音乐爱好者可以随意改变演奏的速度，调式和

音量，也可以根据自己的喜好随意即兴发挥自己的音乐才华。

3) 有了 MIDI 格式后，乐器可以自动进行演奏，例如自动演奏钢琴，从而达到现场演奏的效果。传统的波形信号只能通过扬声器发出声音。

2.2 MIDI 播放器的现状

2.2.1 MIDI 播放软件现状

由于 MIDI 诞生于 80 年代，至今已有 40 多年的历史，因此基于 MIDI 的应用非常广泛。MIDI 最初并不是为计算机设计的，而是为所有电子设备服务的。因此不同设备上的 MIDI 播放软件实现基本上都是基于硬件的，例如电子琴、电子鼓和早期的手机。这些设备内置了所需的合成器和采样器，通过植入 MIDI 芯片来实现 MIDI 文件的解析、合成和播放的功能[7]。

由于计算机的多样性，现在有很多软件可以在计算机上播放 MIDI 文件。例如，Windows 自带的 Windows Media Player 软件就可以播放 MIDI 文件。由于 MIDI 的实现并不复杂，一些早期的音乐播放器在实现文件格式兼容性的同时，也将 MIDI 播放视为一项功能。然而，专门针对 MIDI 播放而开发设计的软件并不多，大多数 MIDI 工具都是为音乐创作者设计的。

2.2.2 MIDI 软件合成器现状

由于 MIDI 只是记录了音符信息，将其转换成波形的声音需要软件合成器。在 Windows 系统中，DirectX 在 Windows 98 中被引入，播放器可以使用 DirectMusic 系列 API 来播放 MIDI 音乐文件。在播放过程中，DirectMusic 使用微软 GS 软件波表合成器将 MIDI 音符转换为波形信息进行输出。此外，早期 Windows 系统还可以使用一些第三方合成器，例如 WinGroove、Orangator 和 SimSynth 等。

在 Linux 系统中，我们也可以安装 ZynAddSubFX 开源合成器来合成音色。早期的手机用于播放铃声的合成器基于芯片，其播放能力较弱并且效果较为单一，例如通常只有一种音色并且只能播放单一声道的音乐。现在的智能机由于技术和机能的发展可以安装更为高级的虚拟合成器以实现更好的播放效果。

2.3 MIDI 可视化播放器的现状

2.3.1 MIDI 音乐可视化

随着计算机性能的提升，音乐播放器的功能也变得多样化，例如 Windows Media Player 可以将音频文件的波形信息转换成频谱，并通过粒子效果展现出来，如图 2.1 所示，这一功能被称为音乐的可视化。

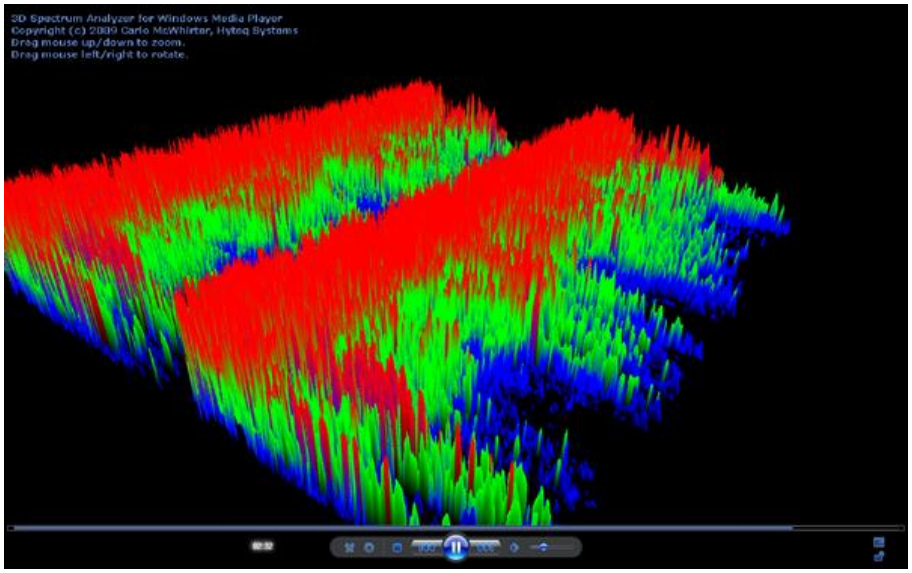

图 2.1 Windows Media Player 的粒子效果

由于 MIDI 的记录方式和普通的音频文件不同，Windows Media Player 并不支持直接将 MIDI 文件转换为波形图进行可视化。然而，基于音符的 MIDI 在音乐的数字化方面更为纯粹，这促使 MIDI 爱好者寻求 MIDI 可视化的播放软件，使得 MIDI 可以以钢琴卷帘的形式展现出来[8][9]。

2.3.2 MIDI 可视化软件现状

为了满足不同用户的需求，目前有许多支持 MIDI 可视化的软件可供选择，其中主流的包括：

1) Synthesia：功能齐全、应用广泛的 MIDI 可视化播放器，付费后可连接电子琴，帮助初学者学习曲谱，也适用于普通 MIDI 爱好者。

2) MAMPlayer：横轴的 MIDI 钢琴卷帘播放器，颜色效果出众，但与其他软件相比功能较为简单。

3) MIDITrail：一款 3D 的 MIDI 播放器，通过纵深显示不同的音轨，提供更加丰富的可视化效果。但由于需要更强大的显卡支持，渲染效果可能会受到限制。

4) Piano From Above（PFA）：一款免费的 MIDI 可视化播放器，与 Synthesia 类似，但更加轻量化且优化效果更好，适用于专业 MIDI 制作者和黑乐谱爱好者。

2.4 MIDI 黑乐谱播放器的现状

2.4.1 MIDI 黑乐谱软件现状

目前在国内外，基于黑乐谱制作的 MIDI 播放器非常少，大多数情况下音乐爱好者使用第三方软件来去除现有 MIDI 播放器的一些限制，通过加大内存、提升 CPU 和显卡的速度等方式来实现黑乐谱的播放。例如，Piano From Above 就是一款非常优秀的 MIDI

可视化播放器，音乐爱好者可以通过该软件定制音轨的颜色，修改可视化的播放效果。然而，对于黑乐谱来说，由于其需要实时渲染的图形非常多，因此 Piano From Above 这款软件在图形渲染的优化上有所欠缺，需要强大的显卡才能进行渲染。由于其未针对黑乐谱进行优化，加载 MIDI 文件也需要较长的时间。

2.4.2 MIDI 黑乐谱合成器现状

除了 MIDI 播放器本身之外，选择合适的 MIDI 合成器也是十分重要的。这是因为音频的合成是由合成器来实现的。MIDI 播放器通常只是将需要播放的音频信息通过 API 发送给合成器，合成器将波形合成后交由系统进行输出。由于黑乐谱需要合成输出的音符数量极多，选择一个轻量化的高效的合成器就显得十分重要。使用性能较差的合成器（例如微软 GS 软件波表合成器）会造成播放的卡顿，丢音，爆音，甚至无法播放黑乐谱。目前主流的 MIDI 合成器包括：

1) BASSMIDI：属于 BASS 系列音乐合成器的一部分，非商业化使用免费。

2) VirtualMIDISynth：免费的商用 MIDI 合成器，为了解决微软 GS 软件波表合成器性能低下的问题而诞生。

3) OmniMIDI：免费开源的软件合成器，如图 2.2 所示。前身为 Keppy's MIDI Converter，目前被认为是性能最好的 MIDI 合成器之一。

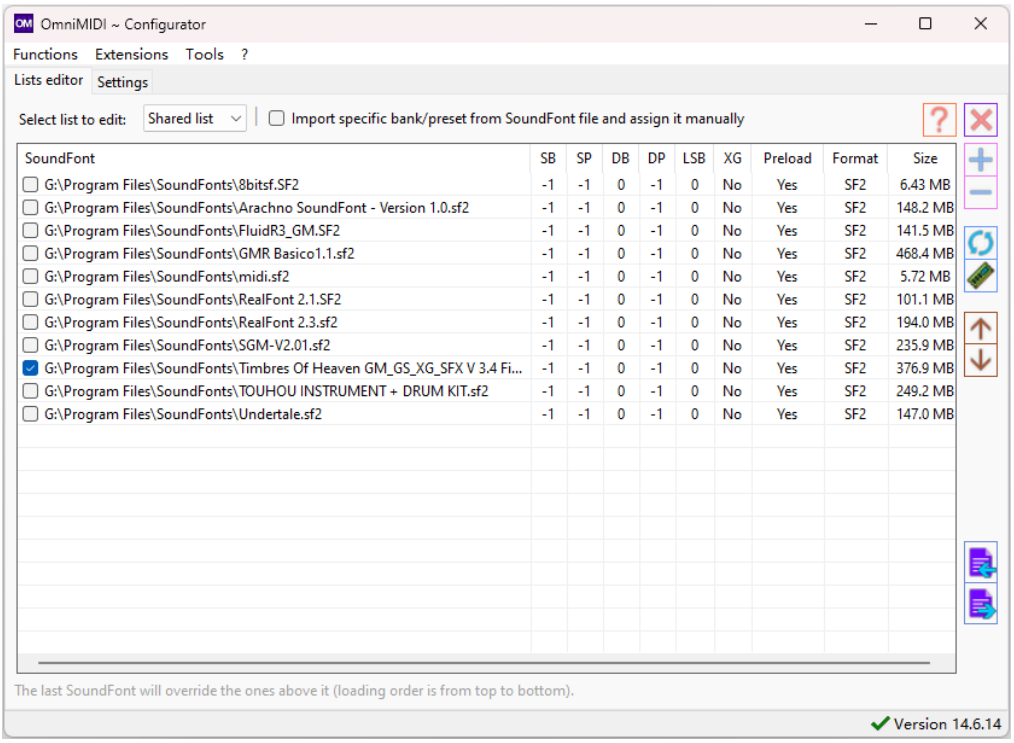

图 2.2 OmniMIDI 合成器配置界面

通过配合优化后的 MIDI 播放器和最新的 MIDI 合成器，可以实现较好的黑乐谱播放效果。现在，几乎所有的黑乐谱爱好者都是采用这种方式来播放黑乐谱的。

2.5 MIDI 视频录制的现状

2.5.1 MIDI 视频录制与创作

由于市面上大多数 MIDI 播放器播放黑乐谱的效果欠佳，尤其是可视化黑乐谱具有一定的困难。因此，许多黑乐谱爱好者选择将黑乐谱手动渲染绘制。这种方法的步骤包括使用视频录制软件将 MIDI 播放器播放的画面逐帧录制，然后使用 MIDI 合成器合成 MIDI 的音频部分。最后，他们将视频和音频部分混合制作成一个视频文件。

随着视频网站的兴起，越来越多的黑乐谱爱好者将自己制作的黑乐谱视频上传到这些网站，供其他人观看。这种方法消除了观看黑乐谱所需的强大硬件的需求。因为视频在制作时已经根据黑乐谱的特性进行了设置并渲染，所以其所展现来的效果最佳。这种方法对于那些不想自己下载并配置 MIDI 软件来播放黑乐谱的爱好者非常有用，因为他们只需要在视频网站上打开黑乐谱视频并直接观看即可。

2.5.2 MIDI 视频录制软件现状

录制黑乐谱视频时，播放和渲染软件需要进行大量的计算，因此需要具备较高的 CPU 和显卡性能。此外，由于录制通常需要较长时间，因此通常需要将 MIDI 文件按时间分割成多个部分，并由多人同时录制，最后再将这些部分合并，这种录制方式被称为合录。许多黑乐谱爱好者会以团队形式使用这种方式制作黑乐谱视频。

由于绝大多数的 MIDI 播放器都是基于播放的需求来设计的，因此目前市面上暂时还没有软件自带录制视频的功能。黑乐谱爱好者只能通过视频录制软件来录制，例如 OBS Studio、ShareX 和 Captura 等[10]。

3 设计与实现

3.1 编程语言与总体设计框架

3.1.1 编程语言，编译器和操作系统平台的选择

计算机编程语言包括机器语言、汇编语言和高级语言等。自从第一款高级计算机编程语言 Fortran 于 1957 年诞生以来，经过几十年的发展，编程语言的种类和数量已经非常丰富[11]。目前主流的编程语言包括 Python、C、Java、C++、C#、Visual Basic 等。

Pascal 语言是上世纪 90 年代流行的编程语言之一，其语法接近自然语言，易于理解。作为一款结构化的过程化语言，Pascal 具有丰富的数据类型和简洁灵活的操作语句等[12]。因此，选择 Pascal 语言来开发 MIDI 播放器是十分合适的。

在上世纪 90 年代，Turbo Pascal 编译器因其丰富和功能广受程序员的欢迎。现如今，基于 Pascal 语言的编译器和整合开发环境越来越多。自 2003 年以来，Free Pascal 编译器发展了多年。其自编译、开源、跨平台的特性广受青少年和入门程序员的欢迎。因此，本程序选择 Free Pascal 编译器作为 MIDI 播放器的开发环境也是基于以上因素的考虑。

由于 MIDI 播放器需要将钢琴键盘可视化，因此窗体应用程序是开发的目标。Windows 操作系统是一款专为窗体应用程序设计的系统，因此成为了首选。此外，Windows 操作系统具有良好的兼容性和开发环境，为编程提供了便利。由于 Windows 10 是目前市场上主流的操作系统，因此程序是在该系统中开发的。

3.1.2 MIDI 黑乐谱播放器的总体设计

作为一款音乐播放应用程序，最重要的是能够打开并播放音乐文件。在 Windows 系统中，我们可以通过以下方式来打开文件：

1) 双击打开音乐文件
2) 带参数运行应用程序
3) 将音乐文件拖拽至应用程序窗体

然而，由于 Windows 系统的限制，打开音乐文件时系统会选择默认的应用程序进行播放。如果用户想要使用 MIDI 播放器打开文件，则需要选择并修改打开方式，此时，系统会以第二种带参数的方式执行应用程序。

在获取音乐文件路径后，需要打开并读取其中的内容并进行解析。通过使用 CreateFile()函数，我们可以以读或写的方式打开文件。鉴于每次文件读写的数据量可能很少，而读写操作会很频繁，同时文件读写的速度很慢，因此可以将整块内容放入缓存，然后在缓存中进行读写操作。只有需要读写文件时，才将整块内容读写到文件中。

读取 MIDI 文件时，需要将其格式内容解析并存入内存中。例如，需要将每个 MIDI 事件的实际发生时间、钢琴卷帘的颜色和长短等内容转换为所需的数据和信息。这一步需要充分理解 MIDI 文件格式，以便能够正确地解析文件内容。

作为一款音乐播放器，对音乐播放和控制是必不可少的。通常，我们需要实现以下功能：

1) 暂停和继续
2) 快进和快退
3) 速度调整
4) 音调调整
5) 上一首和下一首曲目
6) 循环播放的设定

在播放音乐文件时，由于 MIDI 文件是基于音符数据进行播放的，因此可以通过调整音符的音高并将其发送到设备，实现音调的调整。同样地，可以通过修改音符的发生时间来调整播放速度。

作为一款可视化播放器，钢琴卷帘是必不可少的。拟定的功能如下：

1) 显示钢琴卷帘
2) 调整卷帘的速度
3) 调整卷帘的样式和颜色
4) 显示音符的音名或数字
5) 显示小节线和和弦线
6) 显示当然播放状态和其它数据

由于通常需要绘制的音符数量很多，而每次绘制图形都会消耗大量的 CPU 或显卡资源，因此绘制钢琴卷帘的过程不会实时进行。相反，我们可以在 MIDI 文件解析完成后一次性将钢琴卷帘绘制到图形缓冲区中。播放时，我们可以直接从缓冲区中裁切相应的片段内容到屏幕窗体中，而小节线和和弦线也可以一并绘制在缓冲区中。由于其他内容不需要重复绘制，因此可以单独绘制。这项技术在游戏中被广泛应用，例如在横版卷轴游戏中的超级马里奥和纵版射击游戏中的雷电等。

3.1.3 MIDI 黑乐谱播放器的界面设计

作为一款可视化的 MIDI 播放器，本软件的钢琴卷帘采用了纵向的自上而下的方式。这是因为钢琴键盘位于屏幕下方，对于需要使用本软件学习音乐和操作电子键盘的音乐爱好者来说更方便。与一些基于 3D 或带有更多特效的播放器相比，这种设计更适合这种情况。

此外，本软件提供了两种钢琴卷帘的颜色模式，以满足不同音乐爱好者的需求：

1) 和弦模式：音乐学习者需要根据不同的和弦和音色演奏和分析乐曲，将不同音色标记成不同颜色有助于他们的学习和演奏。

2) 展示模式：一般情况下，MIDI 文件中的事件被分为不同的音轨和声道，因此播放乐谱时，根据音轨和声道标记音符的颜色，使得不同的音色被分割开来，视觉效果也更美观。

除了钢琴卷帘，作为一款可视化的 MIDI 播放器，钢琴键盘也是必不可少的组成部分。为了方便音乐爱好者进行学习和调试，使用鼠标按下屏幕上的按键时会发出对应的声音。同时，键盘按键的颜色也会与钢琴卷帘保持一致，并在用户按下对应琴键时显示按键的效果。

为了方便用户实时调整设置，程序还设计了一个菜单系统。基本菜单包含常用的切换曲目的功能，用户可以展开详细菜单，并直接点击屏幕上的按钮来调整音乐播放的速度、音调，并调整钢琴卷帘的颜色模式。同时，为了方便操作，程序也设计了快捷键，使得调整更加便捷。

3.1.4 MIDI 黑乐谱播放器的核心功能设计

在播放黑谱时，由于需要播放的音符较多，为避免声音卡顿和播放速度跟不上乐曲的实际速度等情况，需要针对黑谱进行优化。可以从以下三个角度进行优化：

1) 根据 MIDI 事件类型判断是否需要播放的音符事件，并排除所有不发出声音或声音非常轻的事件。

2) 将音符事件打包成一个整体，通过一次性发送到 Win32 API 接口中的方式，减少 Win32 API 的函数调用次数。

3) 检测并计算播放的延时，并选择性地丢弃音符事件。

在视觉方面，播放黑谱时需要展现其特定设计的可视化效果。使用专门设计的 HSN2RGB()函数，尽可能地在保证不同颜色饱和度最大的情况下，使亮度尽可能一致，如图 3.1 所示。除此之外，还可以通过在注册表专门定制音轨的颜色来呈现最佳的可视化效果。

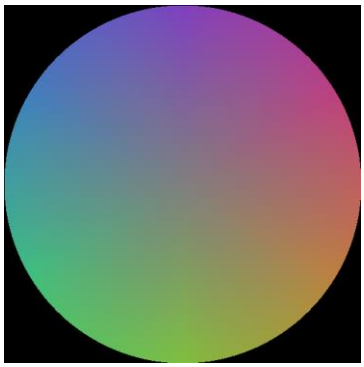

图 3.1 HSN 调色盘

在钢琴键盘上，除了基础的 88 键钢琴键盘以外，在播放黑乐谱时还可能会用到一般钢琴键盘上没有的全部 128 个按键，特殊情况下甚至还需要将所有 256 个按键映射到键盘上。因此，程序应当会根据不同情况选择不同大小的键盘。

总的来说，针对黑乐谱进行的优化和可视化效果的设计是 MIDI 黑乐谱播放器的核心功能之一，如图 3.2 所示，对于使用本软件的用户来说，这些优化和设计都能够提高他们的使用体验。

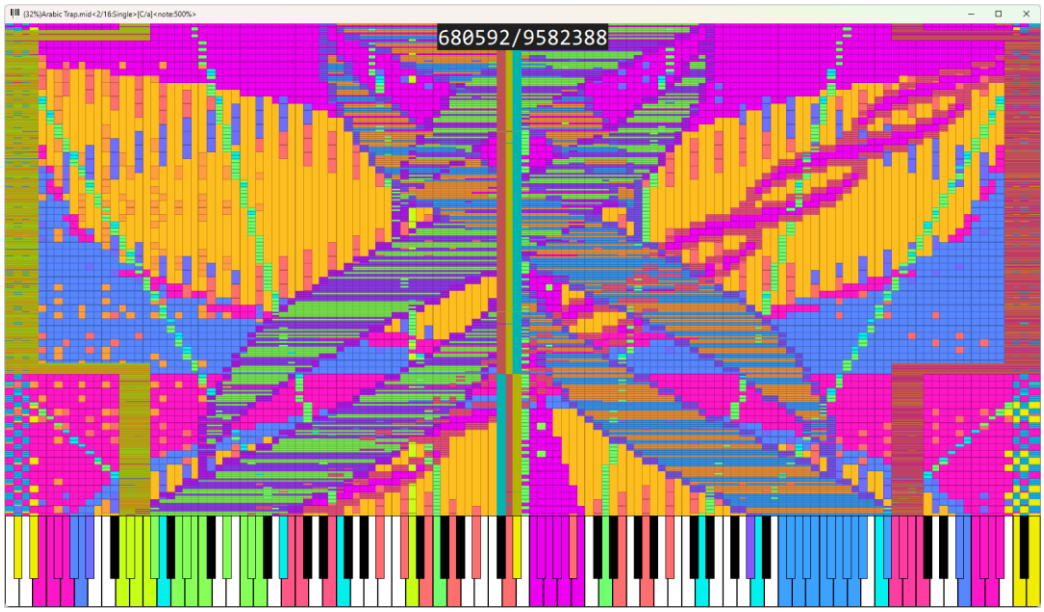

图 3.2 黑乐谱播放器的成品展示

3.2 Windows 窗体设计

3.2.1 Windows 窗体 API 的选择

在设计 Windows 窗体应用程序时，选择合适的 API 是至关重要的。在 Windows 平台上，用于 2D 应用程序开发的 API 主要包括 GDI/GDIPlus 和 DirectX（Direct3D）两种，这两种 API 已经有超过 20 年的历史，因此可以说非常成熟。

鉴于 DirectX 开发的复杂性以及其应用程序需要依赖部分特定的 DLL 组件，我选择采用 GDI/GDIPlus 系列的 Win32 API 来开发窗体应用程序。

3.2.2 窗体创建

在开发窗体应用程序时，考虑到 Win32 API 函数调用的复杂性，为了避免“重复造轮子”，特别针对 Free Pascal 设计了 Display 单元库，以简化窗体应用程序的开发过程。其中包括以下重要的窗体相关组件：

1) CreateWin()：该函数是创建窗体的核心函数，它调用 CreateWindowW 函数，根据窗体的坐标、大小和样式来创建窗体，并在创建之前使用 WinRegister()函数注册窗体

类。

2) WndProc()：该函数是回调函数，会在有消息发送到窗体线程队列时触发，根据不同类型的消息来执行不同的动作，例如 timeBeginPeriod()函数用于设定最小的时间间隔，DragAcceptFiles()函数用于允许拖拽文件等。

3) WinInit()函数：该函数是窗体初始化函数，用于处理创建窗体之前的初始化动作。同时，它还会建立程序内部的消息循环，将消息不停地使用 TranslateMessage()和 DispatchMessage()函数发送到窗体线程队列，并将消息存储到程序内部的数组缓冲区中，以备后续使用。

3.2.3 图形绘制

由于在屏幕上绘制新图形时会将旧图形覆盖，因此在每次重新绘制所有图形之前需要清空屏幕。为了防止在清空屏幕时产生闪烁效果，我采用与绝大多数游戏图像绘制相同的思路：将所有需要绘制的图形绘制在图片缓冲区中，然后再一次性将图片绘制到屏幕上。这样的一次绘制被称为一帧。

在 Display 单元库中，我设计了以下绘图函数：

1) Bar()/Line()/DrawText()：这些函数是绘图函数，可以将方形、线条和文字绘制到图片缓存或屏幕缓存中。

2) FreshWin()：该函数可以将屏幕缓存刷新到屏幕上，从而实现在窗体上进行绘制的效果。

3) LoadBMP()/CreateBMP()/DrawBMP()：这些函数用于从外部图片文件或屏幕读取图片，存储于图片缓存区中，或者将缓冲区内的图片绘制到外部图片文件或屏幕缓冲区中，如图 3.3 所示。

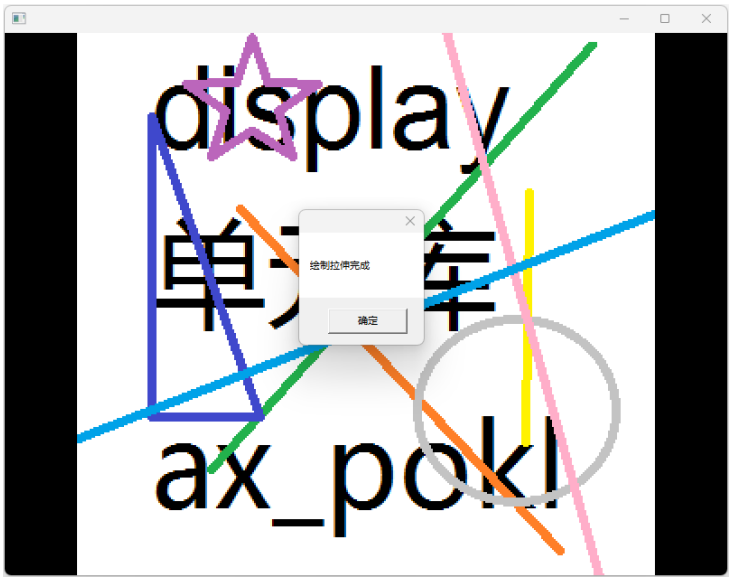

图 3.3 Display 单元库图像绘制测试

3.2.4 多线程

本程序采用了多线程设计。考虑到音频和图形可视化可能不同步，程序将拆分成 4 个主要线程：

1) 主线程：该线程处理程序主要事务，包括程序初始化、音乐文件的读取、时间控制以及消息处理。

2) 音频线程：该线程会将音符通过 MMSystem 系列 API 发送到音频合成器中。

3) 绘图线程：该线程会持续清空图片缓冲区和绘制图形，并将绘制好的图形显示在屏幕上。

4) 窗体线程：该线程是 Display 单元库的内部线程，主要用于分发和处理窗体消息，触发回调函数等。

3.2.5 窗体消息处理

为防止窗体失去响应，主线程会持续循环处理消息。为了避免 CPU 使用率过高产生自旋，当窗体没有消息时会，程序会进行 1 毫秒的等待。

为避免持续绘制图形造成 CPU 占用率过高的情况，程序会设定帧率，并只在需要绘制图像的时候进行绘制。同时，当绘制图像的时间不足时，程序会进行跳帧操作。

考虑到消息处理和窗体绘制并非同时进行，窗体线程会将回调函数无法处理的消息保存到消息缓冲区中，以便主线程进行处理。消息缓冲区中包括鼠标、键盘等常用消息，也包括窗体移动、拖拽、大小变更等特殊消息。此外，为了让不同窗体实例之间能够交互信息，进程之间的信息交互也是通过发送和接收自定义窗体消息进行的。

3.3 Windows 应用程序交互设计

3.3.1 应用程序配置与执行

为了方便用户使用，程序采用“程序名 文件名”的方式打开和播放音乐文件，并通过“-<键> <值>”的方式接收各种参数。这种情况下，如果用户将.mid 格式的音乐文件的打开方式设定为本程序，双击.mid 格式音乐文件将直接使用本程序进行播放。

在 Free Pascal 中的 ParamStr()函数提供了接收参数的方法，使得应用程序可以获取需要播放的音乐文件的路径。考虑到程序执行环境可能不是英文操作系统环境，需要特别注意文件路径可能包含非英文字符。因此，使用 GetCommandLineW()函数可以直接读取包含 Unicode 字符串的路径。

除了直接使用带参数的方式执行程序外，用户还可以将参数以“<键>=<值>”的形式保存在.ini 文件中，以便多次以一定参数执行本程序。在没有指定参数时，程序会读取该文本文件的内容并按照其中指定的参数执行。

在读取.ini 文件时，由于文件路径可能包含非英文字符，同样需要使用 UnicodeString

类型来保存路径。可以使用 Pascal 自带的 reset()和 readln()函数按行读取文件内容，并对读取的字符串进行分割处理。

对于大多数情况下，用户并不关心和了解如何使用配置文件和带参数执行，因此为了方便大多数用户，程序在退出时会把上一次播放的信息存储到注册表中，以便下一次执行时使用。

参数信息会以长整形数值或字符串类型表示，包括上一个播放文件的路径、播放时间、播放速度、音量以及菜单栏中可设定的所有音符样式、长度和颜色等设置。

在处理注册表时，程序使用 RegQueryValueExW()和 RegSetValueExW()函数来读取和写入注册表键值，并对 AniString 和 UnicodeString 进行特别处理，以支持非英文字符。在注册表中，参数会以 REG_DWORD 或 REG_SZ 类型表示。如果程序以带参数形式执行或存在配置文件，则注册表中的参数会被覆盖。

3.3.2 文件拖拽执行

为了提高用户的使用体验，程序采用了文件拖拽方式来播放音乐文件。用户只需将音乐文件直接拖拽到程序窗体中即可播放音乐，这对于绝大多数用户来说是最为方便的，提高了程序的易用性。

为了实现窗体接收拖拽文件的功能，在创建窗体时，需要调用 DragAcceptFiles()函数来允许文件被拖拽到窗体中。当用户将文件拖拽到窗体中时，程序会通过响应 WM_DROPFILES 事件来检测拖拽文件的动作，并使用 DragQueryFileW()函数来获取拖拽文件的路径。

3.3.3 应用程序实例互斥

作为一款音乐播放器，同时播放多个音乐文件通常是不必要的，因为这样会造成声音混乱。因此，当用户在播放一个音乐文件的同时打开另一个音乐文件时，应该考虑到用户其实是想要播放一个新的音乐文件。然而，一般情况下，双击打开音乐文件会让系统执行一个新的程序，从而导致两个程序同时播放不同音乐文件。

为了避免这种情况发生，可以将播放新文件的信息发送给已打开的程序，使其播放新的音乐文件。程序创建窗体时可以使用 RegisterClassW()函数以特定名称注册窗体类。新的程序创建时可以使用 FindWindow()函数检测系统中是否已存在该特定类的窗体，以判断是否已有程序实例在执行。如果检测到实例，则可以使用 SendToInstance()函数将特定的 WM_USER 类型的自定义消息发送给该窗体并退出程序。最后，旧窗体所在程序可以接收自定义消息并处理和播放新的音乐文件。

在某些特殊情况下，例如需要同时播放两个类似的音乐文件进行比对时，用户可以通过直接运行 MIDI 播放器程序文件来实现。由于播放参数中未包含音乐文件路径，新程序不会发送信息到旧窗体，同时也不会退出，而是会读取注册表中的音乐文件路径（通

常是另一个程序正在播放的文件）进行播放。

3.3.4 MIDI 文件遍历与加载

考虑到大多数音乐文件都是放在同一个文件夹内的，为了方便用户，程序包含了简易的文件列表播放功能，和其他大多数音乐播放软件一样。用户可以使用 PgUp、PgDn 等按钮来切换音乐文件。

由于不同类型的 MIDI 文件存在差异，程序不对文件的类型做出要求。所有在当前文件夹下的文件都会被视为 MIDI 文件处理。对于非音乐文件，程序在解析时会将其当做空白文件处理。

在遍历文件夹时，程序首先会使用 GetFileDirW()函数获得当前音乐文件所在的文件夹的路径，然后使用 FindFirstFileW()和 FindNextFileW()函数来获取该文件夹下所有文件的路径[13]。这是因为文件对象都是句柄，必须使用 FindFirstFileW()获取第一个文件的句柄，然后使用 FindNextFileW()依次获取后续文件的句柄和信息。由于音乐文件路径可能包含 Unicode 字符，程序使用的函数系列同样是包含“W”的，以确保正确处理文件路径。

在当前音乐文件播放完毕时，程序会根据设置的循环模式来决定是否重复播放当前音乐文件或切换到下一个音乐文件进行播放。此外，程序包含重新加载当前音乐文件的功能，以防止定位或暂停播放时产生混乱。

在加载 MIDI 文件时，考虑到多次读写文件会使得加载的时间显著变长，因此使用块状读取文件的方式进行处理，以提高加载速度。由于 Pascal 自带的文件函数不方便进行块状读取，程序先使用 CreateFileW()函数打开文件，然后使用 ReadFile()函数直接读取文件内容并放入缓冲区。这样做也方便后续解析文件时，可以按不同的数据类型读取内容。

3.4 MIDI 文件解析设计

3.4.1 MIDI 文件标准格式解析

MIDI 文件的标准格式由标头块和音轨块组成。标头块是唯一的，其中包含文件大小、文件类型、音轨数量和节拍单位等信息。而音轨块的数量则取决于文件类型。

MIDI 文件分为 0、1 和 2 三种类型，其中绝大多数都是类型 2 的多音轨文件。音轨块主要由音轨头、音符事件、元事件和系统信息事件构成。对于每个音轨，音轨头是唯一的，其中包含音轨的长度和事件数目等信息。

在 MIDI 文件标准格式中，表示整数的类型是可变长度值。这种类型的长度不是固定的，它的长度和值的大小有关，其中第一位决定了数值的高位或低位。

3.4.2 MIDI 文件事件提取与分类

MIDI 事件的长度并不是固定的，其中第一个字节的前 4 位代表事件类型，后 4 位代表音轨号。

在 MIDI 文件中，最重要的是音符事件。音符事件由 3 个字节构成，其中第二和第三字节分别代表对应事件的数据。例如，在音符开事件中，第二和第三字节分别代表按键号和音量[14]。

在音符事件中，最重要的是音符开和音符关事件。这两种事件的第一字节的前 4 位分别为 9 和 8。这两种事件构成了钢琴卷帘实际内容的全部部分，因为音符的开和关的时间决定了卷帘的位置。其余的所有事件都无需在卷帘中展现出来，只需要将对应的事件发送给 MIDI 合成器即可，例如音量修改、钢琴踏板的收放等。

元事件分为控制事件、程序事件、声道事件和滑轮事件等，由 2 或 3 个字节构成。对于以 12 或 13 开头的程序事件和声道事件，其数据只有一个字节（第二字节），需要特别处理。

系统信息事件是 15 开头的事件，其数据长度不固定。其内容对于 MIDI 合成器来说可能并不支持或者不重要，可以不发送给 MIDI 合成器，但是可能会对 MIDI 的播放有影响。例如，系统信息可能包含和弦信和节拍的信息，可能会影响到钢琴卷帘的颜色和小节线。

3.4.3 MIDI 文件兼容性支持与容错解析

为了考虑版本之间的兼容性以及与其他类型文件的兼容性，在解析文件时遇到无法解析的内容时应该继续解析，尝试解析越多的内容越好。

有些 MIDI 文件为 RMI 格式，即以 RIFF 格式将标准 MIDI 格式扩展的格式，其文件包含一个 RIFF 标头。因此，在解析时应该跳过 RIFF 标头的部分，而去查找 MIDI 标头。此外，一种还有 KAR 格式（卡拉 OK 扩展的 MIDI 格式），程序目前会将 KAR 格式中的卡拉 OK 信息忽略而当成普通 MIDI 格式解析，在未来会加入字幕以显示卡拉 OK 的内容。

考虑到一些 MIDI 文件的标头可能不标准或存在错误，因此在解析时不能依赖标头内的文件大小信息，而应遵循实际文件大小的信息，以防止解析时出现死循环。

在解析音符事件时，某些类型的黑乐谱为了显示效果可能会存在大于 127 号的按键，而标准 MIDI 只规定了 0-127 号的按键。对于这种情况，可以忽略按键的第一位，将 128-255 号按键当作 0-127 号按键处理。

3.5 MIDI 音频输出设计

3.5.1 MIDI 音频 API 的选择与交互

在 Windows 操作系统中，有三种函数可以将 MIDI 事件发送到合成器：midiOutShortMsg()、midiOutLongMsg() 和 midiStreamOut()。其中，midiOutShortMsg() 只支持单个事件的发送，而后两者支持将多个事件合并后一起发送。

在大多数情况下，为了防止事件遗漏并考虑程序的兼容性，我们应优先使用 midiOutShortMsg() 发送单个事件，尤其是非音符开关的事件。但是对于黑乐谱而言，需要发送的事件很多，如果使用 midiOutShortMsg() 一个一个地发送事件，则会导致音频播放速度跟不上实际速度。因此，对于黑乐谱这种情况，应该将事件打包好后通过 midiOutLongMsg() 或 midiStreamOut() 一起发送。

midiOutLongMsg() 是专门针对较长的系统信息事件而设计的，本身不具备发送打包事件的功能。尽管有些合成器支持该功能，但不应优先考虑使用这个函数发送多个事件。而 midiStreamOut() 则会将打包的事件放入缓冲区一并发送给 MIDI 合成器，发送速度更快，因此应该优先考虑使用这个函数。为了方便使用，程序也设计了在这两种发送方式之间切换的功能。

3.5.2 MIDI 合成器的选择与交互

由于微软默认的 GS 软件波表合成器性能较低，在复音数较高的情况下容易出现音符丢失。为了测试黑乐谱，可以选择使用 VirtualMIDISynth 或 OmniMIDI。其中，后者是一款开源软件，并且性能更优秀，应该优先考虑使用。

由于系统中可能安装了多款不同的 MIDI 合成器，在播放 MIDI 文件时需要进行选择。程序提供了合成器切换的功能，切换时会重置 MIDI 设备。如果 MIDI 设备播放不正常（例如音调或音量错误，复音过多导致音符无法结束，音符没有声音等），可以尝试强制重置 MIDI 设备。如果在切换 MIDI 文件后出现播放问题，也可以尝试重新加载 MIDI 文件来解决问题。

3.5.3 MIDI 音频控制功能设计

作为一款 MIDI 音乐播放器，暂停和定位功能是必不可少的。除此之外，由于 MIDI 文件由音符事件组成，因此设计功能来改变音量、速度和和弦也非常重要。

在暂停过程中，某些音符可能仍处于按下状态而发出声音（例如弦乐和管乐）。为了避免声音嘈杂，应发送事件关闭所有正在按下的音符。此时，画面仍可以显示音符为按下状态，以方便音乐爱好者进行研究。

在定位过程中，我们也需要关闭所有正在播放的音符。此外，由于播放状态可能已经发生改变（例如音量、播放乐器类型、音调等），还需要重新遍历 MIDI 文件，并将

所有非音符事件重新发送给合成器，以确保音频正确播放。

然而，当用户调整音量时，由于每个 MIDI 音符事件都包含音量信息，因此可以在原有音量的基础上根据调整的音量大小增加或减少音符的音量数据，从而控制音量。

对于速度的调整，可以通过修改程序内置流逝的时间来实现。在播放时，通过计算修改速度时的定位时间、实际时间和当前实际时间之间的差值，可以计算出当前位置的时间，从而加速或减速播放。

在调整和弦时，可以通过修改音符的按键号来改变音调。对于按键号过低或过高的音符，可以不进行处理。但是，在这种情况下，也需要处理好音量和非音符事件。

3.6 MIDI 图形可视化设计

3.6.1 钢琴卷帘

钢琴卷帘是 MIDI 图形可视化中最重要的部分。钢琴卷帘由一系列长方形的滑条构成，每个长方形的高度和位置由音符的起始时间和结束时间以及卷帘的长度设置决定，而宽度则和钢琴键盘的按键一致。长方形的颜色则取决于不同的颜色模式，有以下三种：

1) 和弦模式
2) 音轨/声道模式（黑白键）
3) 音轨/声道模式

在和弦模式下，音符的颜色取决于其按键号对应的音名和所在和弦。在选择颜色时，我们将 12 个音符的颜色平均分为 12 份，其中每 5 度相差一份颜色，如图 3.4 所示。这样可以保证完全和弦的音符颜色尽可能相近，方便进行和弦分析等研究。

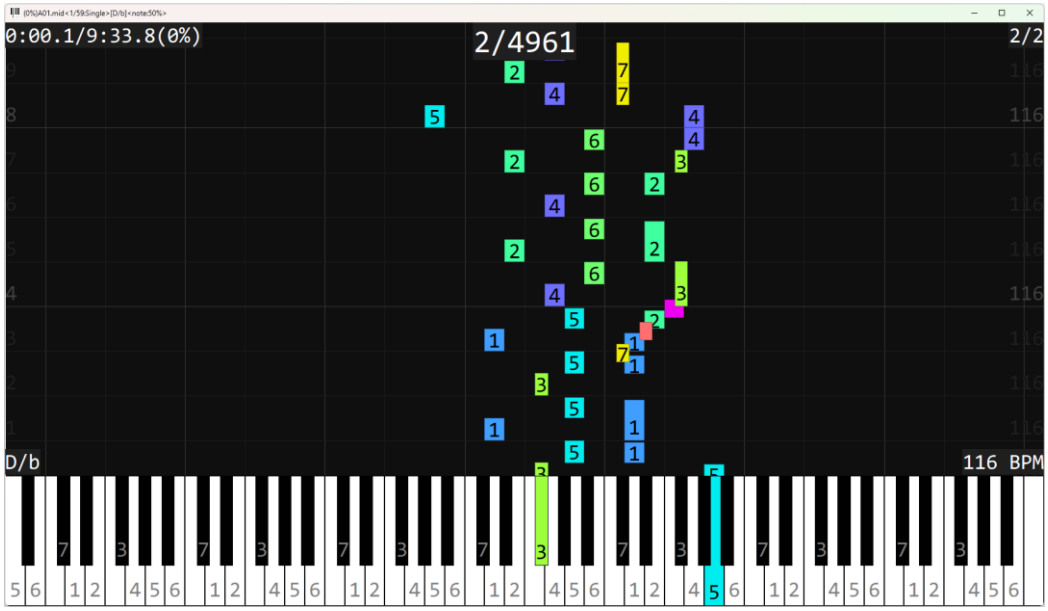

图 3.4 黑乐谱播放器的和弦模式

对于和弦的调整比较复杂。除了显示不同的和弦信息外，由于音符的位置已改变，还需要重新根据新的和弦重新绘制整个钢琴卷帘。这类似于调整钢琴卷帘的长度或样式时的情况。此时还需要考虑卷帘具体的颜色和和弦之间的关系。

在音轨/声道模式下，每个音轨/声道中的音符颜色是相同的，如图 3.5 所示。为了保证不同音轨/声道的颜色尽可能不同，先将颜色平均分为 3 份（红绿蓝），当音轨/声道颜色更多时再将剩余颜色分为 3 份（黄青紫），然后再将剩余颜色分为 6 份，以此类推。在选择颜色时，会根据音轨/声道的音符总数进行排序，并按照顺序分配颜色，以避免音符数量少的音轨/声道占用平均分配的颜色，导致音符数量多的音轨/声道颜色不够用，从而产生相近的颜色。当然，一些黑乐谱爱好者可能对音轨/声道的颜色有更多的要求，因此还可以通过添加注册表键值的方式对每个音轨/声道的颜色进行单独设置。

除了长方形滑条本身外，还可以在长方形上方标记音名或数字。为了保证字符的一致性，使用等宽字体 Consolas 来进行显示。由于卷帘是自上而下的，因此字符的位置位于长方形底端，大小不超过长方形的宽度，而高度则会根据长方形高度被截断或完全显示。对于黑键音符，字母会显示为小写，而数字则不会显示。为了区分多个不同的音符，当同一个音连续出现时，长方形会带有混合灰色的边框，新的（时间靠后的）音符则会覆盖旧的（时间靠前的）音符。

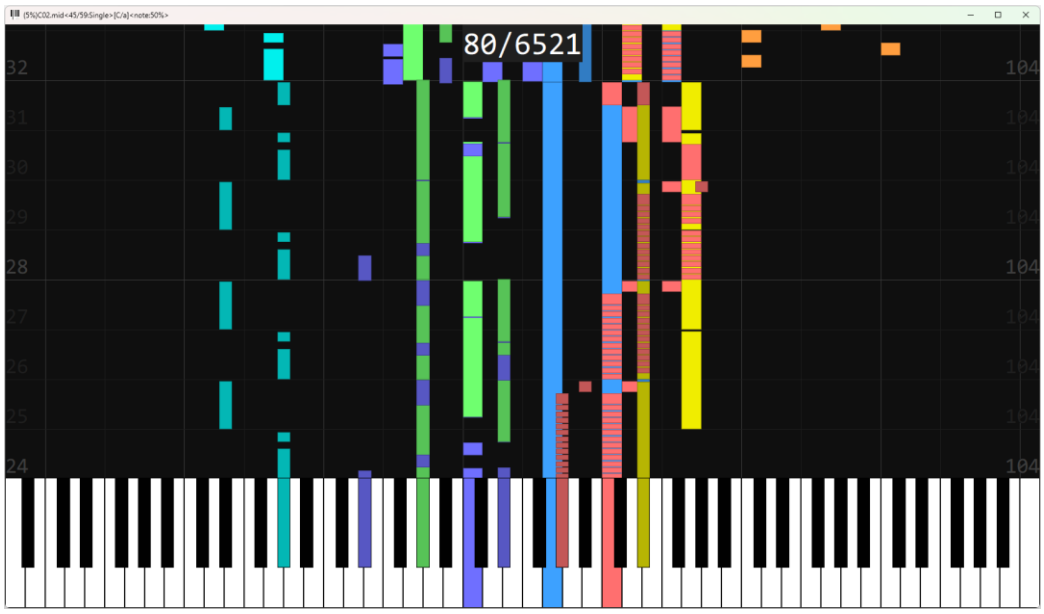

图 3.5　黑乐谱播放器的音轨/声道模式

3.6.2　钢琴键盘

作为一款拥有钢琴键盘的音乐软件，当用户在键盘上按下键时，程序会将对应音符发送到合成器以发出声音。为了避免混乱，在此时键盘上的颜色不会产生变化，其状态和未按键时一致。

钢琴键盘分为黑键和白键，其颜色、长度和位置与真实钢琴键盘类似。在按键按下

时，按键的颜色会发生变化，其颜色与钢琴卷帘一致。同时，为了强调按下的键，按钮上的字符也会以纯黑色“高亮”显示出来，一般情况下则会混有灰色。

在遇到音符重叠的情况下，尤其是不同音轨/声道同时按下同一个按键时，在延时不同的情况下遵循“谁先来，谁生效”的原则。在重复接收到音符开的事件时，保持音符按下，当收到一个音符关的事件时，立即放开。尽管在这种情况下，某些音符可能仍在播放且钢琴卷帘仍有音符显示，钢琴键盘也应该放下按键，以使按键的按下和弹起动作显得独立和连贯。

3.6.3 小节线与文字信息

为了让视觉效果更加丰富、音符更整齐，会在钢琴卷帘背后添加横线小节线。根据每小节的 3、4 或其他拍数信息，还将线细化为小节线和节拍线，并给节拍线使用较淡的颜色。同时，在线段两端分别添加节拍号和播放速度。

除此之外，在切换节拍号或调号（和弦）时，专门以“高亮”颜色显示分割线并标记。同时，按键也会根据音名从 C 到 B 分组，并在纵向添加分割线。这样可以将钢琴卷帘彻底“网格化”。

除了小节线以外，在屏幕左上方、正上方、右上方、左下方和右下方分别显示时间、音符数、复音数、调号和速度等文字信息，以便用户了解当前音乐的播放状态。在屏幕左侧还会显示每个音轨/声道的颜色和音符数量。

如图 3.6 所示，实际显示时，小节线、文字信息、钢琴卷帘以及钢琴键盘会分别绘制到不同的图形缓冲区中，最后再合并绘制到屏幕上。

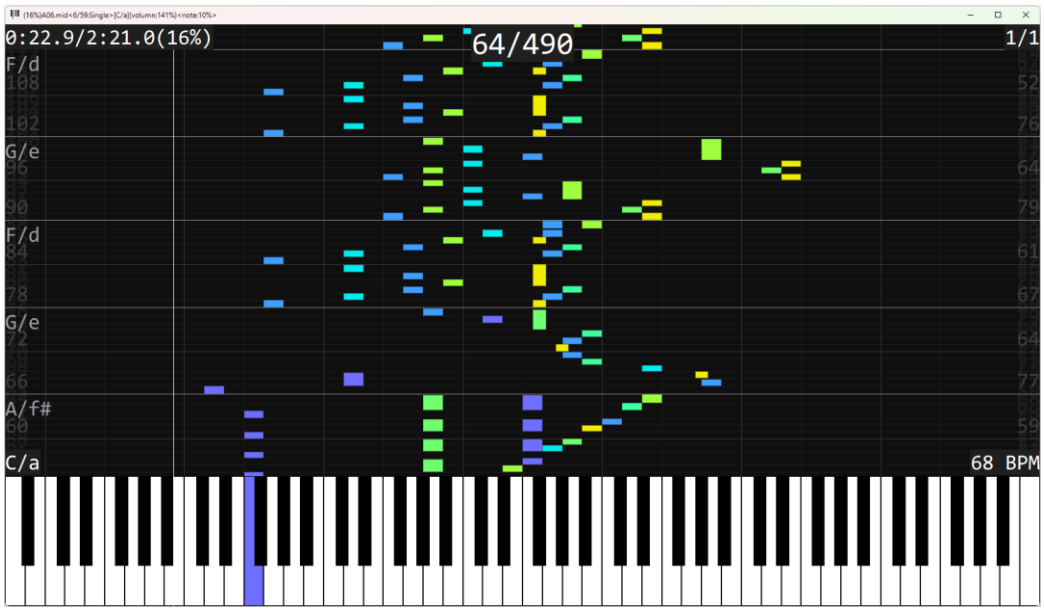

图 3.6 黑乐谱播放器中的小节线与和弦线

3.6.4 菜单系统

为了方便用户在播放音乐时实时进行调整和配置，除了使用传统的快捷键，本程序

还设计了菜单系统。如图 3.7 所示，使用菜单可以快速切换和加载曲目，并且在鼠标移动时会在屏幕上方显示出来。

在详细菜单中，用户可以对所有内容进行设置，包括：

1) 播放设置：调整乐曲的音量、播放速度、调号等。

2) MIDI 设备：切换 MIDI 设备和 MIDI 事件发送函数等。

3) 显示设置：修改音符长度、颜色和样式，修改小节线和文本信息样式等。

4) 优化设置：切换针对黑乐谱的设置，包括切换内存和文件，设置缓冲区，最小音量和帧率等。

5) 其它设置：录制视频，显示帮助和重置所有设置等。

更多详细信息可参考附录中的“MIDI 黑乐谱播放器说明书”。

通过以上设计，MIDI 黑乐谱播放器不仅在功能上具有丰富性和便利性，而且在操作上也更加方便和人性化。

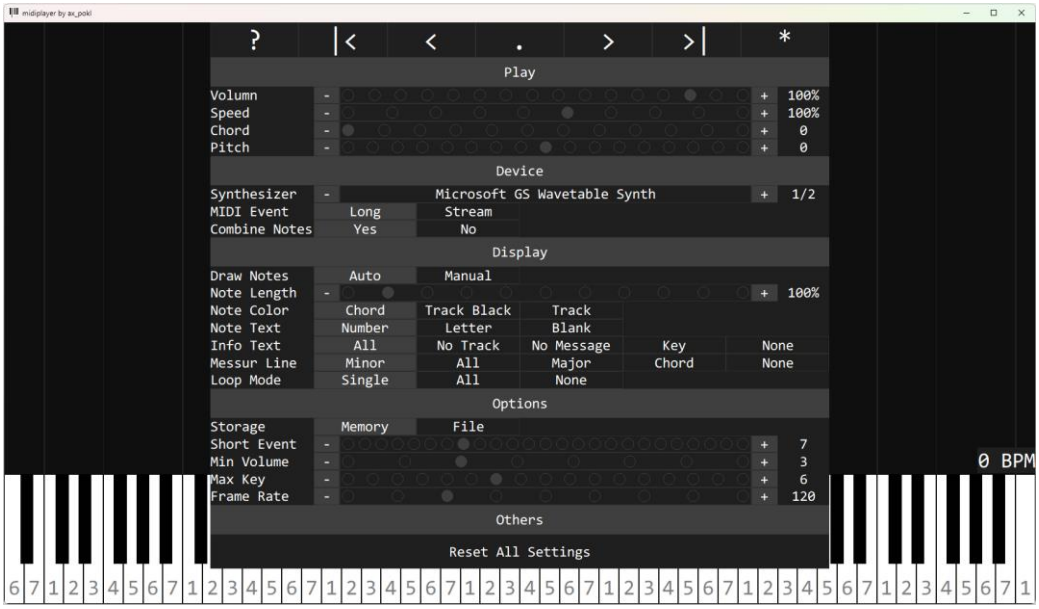

图 3.7 黑乐谱播放器的菜单系统

3.7 黑乐谱优化设计

3.7.1 MIDI 事件排序与优化

在实际编程和测试过程中，我发现对于黑乐谱的播放存在着很多问题。其中，最首要的问题就是需要对 MIDI 事件按照事件顺序进行排序。一般情况下，快速排序算法是所有比较交换类排序中速度最快的算法。然而，由于 MIDI 事件本身已经按照音轨/声道排列好了，因此在这种情况下使用快速排序算法并不是最优的选择。此外，由于排序文件内容是随机的，快速排序算法需要多次读写不同文件块，这会极大地降低排序的速度。

因此，我选择了多路归并排序算法来对 MIDI 事件进行排序处理。

在使用多路归并排序算法时，程序会针对每个音轨/声道添加一个指针，并将每个指针所指向的文件块放入缓冲区中。接着，程序会对缓冲区中指针所指向的元素进行排序，排序后的元素放入待写入的缓冲区中。当读取或写入的元素超出范围时，再进行真正的读写操作。在排序过程中，采用整块交换的方式，而非单个元素的交换方式。由于指针读写的元素都是连续的，因此读写文件块的顺序也是连续的，不会出现随机读写的情况，这进而大大加快了排序速度。

除此之外，在为音轨/声道选定颜色时，我们也需要对音轨/声道的颜色进行排序。由于颜色的选择是随机的，而音轨/声道的数量不会太多，因此可以在内存中完成排序。因此，我选择使用快速排序算法对音轨/声道的颜色进行排序处理。在排序的过程中，先根据音轨/声道的音符数量进行排序，然后为音轨/声道分配颜色，最后再将选好的颜色排序结果重新应用到原来的音轨/声道中。显示的效果如图 3.8 所示。

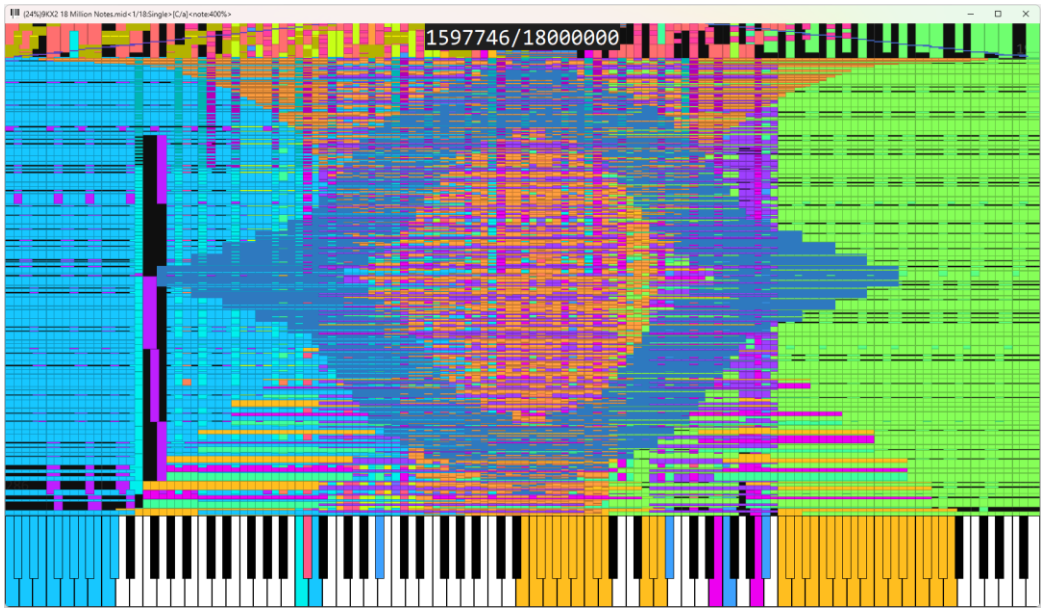

图 3.8　调整颜色后的黑乐谱展示

另外，在解析 MIDI 事件的过程中，我们还需要对小节线和调号线进行排序处理。由于小节线和调号线的数量不多且它们出现的时间完全随机，因此采用插入排序算法的方法将新读取的小节线和调号线插入到已有的数组中。在需要定位时，再使用二分法确定它们的位置。

因此，黑乐谱的整个读取准备过程可以分为以下四个步骤：

1) 读取 MIDI 文件并进行解析。
2) 对 MIDI 事件进行排序。
3) 计算 MIDI 事件的实际事件并生成小节线等信息。
4) 生成钢琴卷帘信息并对音轨/声道选定颜色进行排序。

在这个过程中，准备的进度会实时显示在屏幕上，以便用户可以清晰地了解整个读取准备过程的进度。

3.7.2 MIDI 事件播放的优化

在某些黑乐谱的音符过于密集的极端情况下，即便使用 midiStreamOut()函数进行播放，仍然可能出现音频无法跟上播放速度的情况。此时就需要处理或丢弃部分音符，可以采用以下几种方式：

1) 忽略音量低于一定数值的音符。
2) 合并按键号相同的音符。
3) 尽可能多地打包事件，以减少 midiOutShortMsg()的使用。
4) 强制丢弃部分音符。

在丢弃音符的过程中，程序也会放弃钢琴键盘的按键，以减少计算开销。

在定位和播放音符的过程中，由于黑乐谱的音符数量极多，而音符数据是按照时间顺序排列的，因此需要使用一种方法快速定位到对应时间的音符。我们可以使用二分法查找事件，大大缩短定位的时间。由于从头遍历 MIDI 事件花费的时间极多，因此我们可以设定一个缓冲值，只读取和发送定位时刻之前一段时间内的事件给合成器。由于对于黑乐谱播放者来说，视觉效果比音频效果更为重要，因此可以牺牲部分音频播放的准确性。

3.7.3 钢琴卷帘缓存与优化

在分配钢琴卷帘的图形缓冲区时，需要指定缓冲区的高度和宽度。由于这一数值是有限的，分配过大的缓冲区可能会失败，因此无法将整个钢琴卷帘放入一个缓冲区中，需要根据大小分割缓冲区。此时，在绘制长方形和文字时，需要分别绘制到不同的缓冲区中。当遇到长方形和文字跨越多个缓冲区时，还需要多次将其绘制到所有的缓冲区中。在需要显示钢琴卷帘时，从不同缓冲区中提取和拼接需要显示的部分，最后绘制到屏幕上。

由于音乐播放和图形绘制是同时进行的，而黑乐谱的完整绘制区需要很长时间，因此可以只绘制一部分钢琴卷帘图形（通常为两个屏幕），并优先将结果显示在屏幕上。尽管播放时视频可能会出现一些卡顿，但只要绘制的速度比播放的速度快，乐曲播放完毕时就能完整地绘制出来。第二次播放时，由于缓冲区已经绘制完毕，播放时视频就不会卡顿。然而，如果在绘制未完成的情况下进行跳转，旧的卷帘可能会覆盖新的卷帘，导致视觉上的不一致。此时需要重新完全绘制缓冲区。

对于部分音符长度超过两个屏幕的情况，可以先将所有超过两个屏幕的音符绘制出来，然后再绘制短的音符，以避免音符顺序进行时需要将缓冲区完整绘制出来。这一步通常在生成钢琴卷帘信息后进行。

当钢琴卷帘区过长时，需要分配许多图形缓冲区来绘制屏幕。但当缓冲区的数量超过一定限制时，Windows 创建图形缓冲区的函数可能会失败，从而导致后续的分配也失败，程序最终无法正常运行。

为了避免这种情况的发生，可以限制缓冲区的数量，并将多余的缓冲区以 BMP 文件的形式保存到临时文件夹中，如图 3.9 所示。当缓冲区需要被使用时，再从 BMP 文件中读取。可以将带有小节线和钢琴卷帘的内容保存到 BMP 文件中，并将黑键和白键分开处理，以确保黑键覆盖在白键之上。

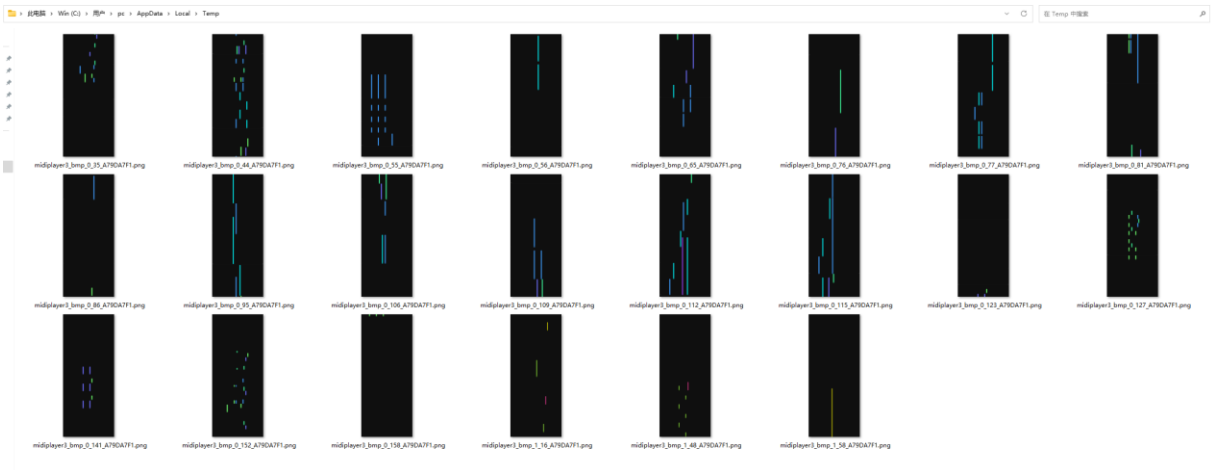

图 3.9　存储在临时文件夹内的 BMP 图像

3.7.4　存储系统优化

为了应对 MIDI 黑乐谱文件极大时而导致内存不足的情况，本程序采取了将事件保存到临时文件夹的方式。根据之前提到的 4 个步骤，事件被分为 3 个部分并保存到不同的文件中，分别是解析的事件、排序后的事件和钢琴卷帘信息。当不再需要使用这些文件时，程序会释放并清空它们，以减少磁盘占用空间。这样，程序的内存占用始终是有限的。

在排序事件的过程中，采用的是块状读写文件的方式。由于音频播放也是按照时间顺序播放，因此在播放时也采用了块状读取文件的方式。为此，程序专门开发了函数来实现这个功能。在定位时，程序也使用二分法以尽可能减少对文件的读取次数，以提高读取速度。

3.8　视频录制设计

3.8.1　视频录制架构与 API 的选择

为了呈现最佳的黑乐谱效果，许多黑乐谱爱好者会使用特定软件以及一定的配置来播放黑乐谱。同时，他们也会将播放过程录制为视频并上传到视频网站，与其他黑乐谱爱好者分享。考虑到这一需求，程序特别添加了视频录制功能。

为了实现视频录制，本程序使用了跨平台的开源视频录制解决方案 FFMPEG。市面上绝大多数视频播放、录制和转换软件都是基于 FFMPEG 来实现的。

本程序使用了 FFMPEG 的多个库，包括 libavcodec、libavformat、libavformat_avio、libavutil_frame、libavutil_mem、libavutil_pixfmt、libavutil_rational 和 libavutil_mathematics。

考虑到 MIDI 播放器是一个独立的程序，因此程序通过加载 DLL（动态链接库）的方式来实现视频的录制，从而可以直接将屏幕图像信息写入视频中。

3.8.2 视频录制编码器与格式的选择

考虑到视频录制的复杂性，本程序选择了当前最主流的编码器 H.264/MPEG-4 AVC 进行编码，并将编码的文件存储在 MP4 文件格式容器中。为了实现视频录制，程序专门创建了 EncodeVideo()、EncodeFrame()和 WriteFrame()等函数来录制和写入视频帧。在具体的实现过程中，使用了一些核心函数[15]，如：

av_guess_format()：设定输出格式类型

avformat_alloc_context()：分配格式上下文

avcodec_find_encoder()：查找编码器

avcodec_alloc_context3()：分配编码器上下文

av_make_q()：设定帧率

avformat_new_stream()：创建视频流

avcodec_open2()：打开编码器

avio_open2()：打开输出文件

avformat_write_header()：写入文件头

av_frame_alloc()：分配视频帧

avpicture_fill()：填充视频帧的图像信息

av_init_packet()：初始化视频数据包

avcodec_encode_video2()：对视频数据包进行编码

av_rescale_q()：设定帧间隔

av_write_frame()：写入视频帧

av_write_trailer()：写入文件结尾

4 结论与展望

4.1 MIDI 黑乐谱播放器总结

4.1.1 MIDI 黑乐谱播放器设计总结

本研究致力于设计并实现一款功能强大的 MIDI 黑乐谱播放器，经过不断地研究和开发，最终具备了播放 MIDI 音乐文件、以钢琴卷帘方式可视化展示音乐、播放黑乐谱、录制视频等多项功能。在设计中，采用了模块化的设计思想，将功能模块分离并独立开发，最终实现了整体的功能。通过不断的测试和用户反馈，该播放器的性能和用户体验得到了良好的验证。

4.1.2 MIDI 黑乐谱播放器技术总结

在技术实现方面，使用了 Pascal 语言实现了窗体显示、应用程序交互、MIDI 文件解析、MIDI 音频输出、图形可视化、菜单和控制等多项功能，并采用了多种技术优化黑乐谱的播放体验。例如，采用排序算法和二分法优化黑乐谱的读取和播放，采用多线程和 MIDI 事件缓冲切换优化播放效率，采用屏幕图像缓冲和内存和文件切换等技术减少了计算机资源占用，提高了软件的性能表现。

4.2 MIDI 黑乐谱播放器后续优化与维护

4.2.1 MIDI 黑乐谱播放器错误疑难排查与解决方案

MIDI 黑乐谱播放器是一款较为复杂的应用程序，因此在使用过程中难免会出现一些错误和问题。为了让用户能够顺利地使用该播放器，将不断地测试和完善该播放器，排查和解决软件错误。同时，记录每一次出错问题的原因和解决方案，以备将来查阅和总结经验教训。

4.2.2 MIDI 黑乐谱播放器后续新功能与维护

每一款优秀的软件都需要持续维护和更新。随着使用 MIDI 黑乐谱播放器的人数不断增加，我会不断收到来自用户的反馈和建议，也就会有越来越多的想法和需求出现。

为了更好地满足这些需求，我将持续维护 MIDI 黑乐谱播放器并添加新的功能。例如，增加更多的演奏效果展示、为教学者提供更多音乐分析功能、增加对更多文件类型（如 KAR 格式）的支持，以及使用其他的图像显示技术（如 Direct X）对软件性能进行优化等。相信这些新功能和优化将使 MIDI 黑乐谱播放器更加出色，也将更好地满足用户需求。

4.3 MIDI 黑乐谱播放器推广与合作

4.3.1 MIDI 黑乐谱播放器推广与传播

为了让更多人了解和使用 MIDI 黑乐谱播放器，我打算加强推广和传播工作。具体而言，我计划在各大社交媒体平台和视频门户网站上宣传该播放器，并积极鼓励用户分享他们的使用体验和反馈。

通过这种方式，我相信这些推广和传播的努力将能让更多的人了解和使用 MIDI 黑乐谱播放器，也相信这些努力将为 MIDI 黑乐谱播放器的发展带来更多的机遇和可能性。

4.3.2 MIDI 黑乐谱播放器创作与合作

为了探索 MIDI 黑乐谱播放器的应用和创新，我将与音乐创作者和其他相关领域的专业人士合作。例如，我将会通过与音乐创作者合作，创作更多的黑乐谱，同时寻找黑乐谱录制者录制和分享更多视频。

除了以上内容，我还计划通过参加相关的音乐展览、工作坊等活动，来展示 MIDI 黑乐谱播放器的应用和功能，从而吸引更多的用户和合作伙伴。

我会通过不断地与专业人士合作，推出更多的功能和应用，让更多的人知晓和使用 MIDI 黑乐谱播放器，从而为黑乐谱的创作和传播，以及音乐教育和娱乐等领域做出更多的贡献。

参考文献

[1] 王毅. 电子音乐发展史. 科学大众，2012，682：02，173

[2] 龚镇雄. 计算机音乐的进展. 中国音乐学，1995，04，88～95

[3] 邓紫芯，王芳. 试论动态可视化在音乐教学中的应用. 当代音乐，2022，665：08，56～58

[4] 甘冲萍. 音乐播放产品设计的发展设想及可能性概念的研究[D]. 上海：同济大学，2008

[5] 古钰. MIDI 音乐的基本应用分析与研究. 中国民族博览，2018，01，144～145

[6] 谭劲. MIDI 接口和 MIDI 文件格式. 计算机时代，1996，03，23～24

[7] 刘嘉欣. 嵌入式 MIDI 文件格式解析设计与实现. 微计算机信息，2006，32，66～68

[8] 韦岗，曹燕，王一歌，赵明剑. 计算机音乐可视化表征谱设计. 现代信息科技，2019，3：14，5～7

[9] 苏泉林. 基于“主旋律与情绪”的音频可视化产品设计研究[D]. 长沙：湖南大学，2019

[10]席文强. 基于 FFmpeg 的高清实时直播系统设计与实现[D]. 西安：长安大学，2017

[11]张磊. 从信息技术的发展看办公自动化系统发展趋势[J]. 武汉：中国地质大学，2018

[12]何树印. 从 BASIC 到 Scratch——浅谈教学用编程语言演变[J]. 山东：山东省淄博市电化教育馆，2015

[13]于麦燕. 基于 Web20 的企业信息平台设计与实现[D]. 成都：电子科技大学，2012

[14]程昭德. 基于双向 LSTM 的 MIDI 钢琴演奏评价方案设计与实现[D]. 广东：广东工业大学，2020

[15]熊华为. 基于 H264 和 DASH 视频传输系统的设计[D]. 重庆：重庆邮电大学，2017

致谢

我研究的领域是 MIDI 播放器和黑乐谱，这是一个非常冷门小众的领域，因此在完成这篇论文的过程中遇到了很多困难。在此，我要向所有支持和帮助我的人们表示感激之情。

首先，我要感谢我的导师王卫兵，他为我的研究提供了重要的指导和建议，对我的学术发展提供了宝贵的帮助。他的鼓励和支持使我在写论文的过程中坚持不懈地前行，不断探索并取得了一定的成果。

其次，我要感谢学习 Pascal 语言的同学和伙伴，他们在我遇到困难时给予了帮助和支持，使我能够克服问题，继续前进。我还要感谢我喜欢黑乐谱的好友，他们对我的研究软件提出了建议，帮助我不断改进和完善软件功能。

此外，我要感谢我的家人和同事，他们一直支持我、鼓励我、关心我，在我遇到挫折时给了我力量，让我能够坚定信心，继续前进。他们的支持和鼓励对我来说非常重要，让我感激不尽。

最后，我也要感谢我自己，因为是自己不断的努力和坚持，才得以完成这篇论文。我在研究中学到了很多新的知识和技能，同时也锻炼了自己的能力和耐心，这对我的学术生涯和职业发展都是非常宝贵的财富。

在这里，我要向所有帮助过我的人表示感激，特别是那些在我撰写论文时提供过帮助的人们。我也希望我的研究能够为这个小众领域的发展做出一些贡献，让更多的人了解和关注这个领域。