更新日志

2021-12-07

- macOS版本已完成，经过一些调整，我终于实现了让tkinter与pyglet一起工作而没有问题。然而，在macOS上播放时，pygame的mixer不能暂停MIDI文件，更确切地说，函数`pygame.mixer.music.pause()`不起作用，这个问题只发生在macOS上，对于Windows和Linux是可以的。我尝试了此时最新的pygame版本2.1.0，以及2.0.0，2.0.1，都不起作用。这不是pygame 与 pyglet 和 tkinter 一起工作时造成的问题，因为我在另一个只使用 pygame 的程序中测试了播放 MIDI 文件时的暂停功能，得到了同样的结果。

  所以在pygame的开发者修复这个bug之前，对于macOS版本，使用pygame的mixer的MIDI文件播放的默认设置不能在播放时暂停MIDI文件。如果你想在目前使用macOS版本的Ideal Piano中播放MIDI文件时暂停和取消暂停，你可以通过将设置文件中的`use_soundfont`改为True来切换到使用fluidsynth在Ideal Piano中播放MIDI文件，而这需要你在macOS上安装fluidsynth。这很简单，你可以用homebrew来安装fluidsynth，运行`brew install fluidsynth`并等待fluidsynth安装到你的电脑上。使用fluidsynth，你也可以使用不同的SoundFont文件播放MIDI文件。

2021-12-05

- 删除了keyboard模块，现在本软件中的电脑键盘检测由pyglet处理，此更新同时适用于Windows和Linux版本。这一改进有很多好处。

  首先，pyglet是创建本软件的主要的库，所以既然它支持读取键盘输入，就没有必要使用另一个python库来检测按键。

  其次，python的keyboard库有一个问题，它是全局检测按键的，这意味着当你进入电脑键盘演奏模式时，即使你不关注软件的屏幕，例如你最小化窗口并在另一个软件中打字，keyboard模块仍然会检测到按键并产生像你使用它时的钢琴音。使用pyglet检测按键就不会有这个问题，只有当你专注于软件的屏幕时才会检测到按键。

  第三，Linux版本需要使用sudo来打开，因为keyboard模块需要root权限，现在你可以双击可执行文件并打开软件。正在开发中的macOS版本，由于同样的原因需要sudo，现在至少在打开方面有了一点改进。

- 改进musicpy和browse的导入，这使得代码在函数和变量的命名空间方面更加安全。

2021-12-03

- 在sf2_loader包的更新之后，现在Ideal Piano本身能够在内部实现一个SoundFont播放器，它允许使用SoundFont文件播放MIDI文件，而不需要预先渲染成音频，并且可以实时暂停，取消暂停，停止。这使得使用SoundFont文件播放MIDI文件的启动速度快了很多，因为MIDI数据被直接发送到播放器上进行实时操作。

- **注意：这个更新目前只针对Windows版本**，因为在Linux上，通过`apt-get`安装的最新fluidsynth版本是1.9.1（我在Ubuntu上测试了Linux版本），这实际上是2018年的老版本，此时fluidsynth的最新版本是2.2.4，这个老版本没有许多在新版本fluidsynth中引入的重要功能，这些功能在sf2_loader以及Ideal Piano中使用得很多，例如`fluid_player_seek`。

  fluidsynth播放器的许多功能，如`fluid_player_stop`和`fluid_player_play`的行为也与新版本不同，例如，在2.2.4版本中，如果你停止播放器，声音几乎会立即暂停，但在Ubuntu上由`apt-get`安装的最新版本（也就是1.9.1），这将导致当前播放的音符永远持续播放，这是非常恼人的（这意味着在新版本中，开发者已经修复这个错误）。

好吧，我可以向所有的通道发送所有的声音关闭事件来停止声音，这很有效，但最恼人的问题来自于当你试图取消播放器的暂停时，在2.2.4版本中，你只需再次调用`fluid_player_play`，播放器将继续播放MIDI文件，几乎没有等待时间，但在1. 9.1版本的Ubuntu上，这将导致播放器在继续播放之前有很长的等待时间，而且每次都不同，从3秒到30秒不等，在等待时间内，音符条已经持续移动了很多，所以声音和音符条将完全不同步。这使得在Linux上使用fluidsynth的Ideal Piano播放MIDI文件的暂停和解除暂停功能完全无法操作。

我试着在Linux上使用不同的音频驱动，包括alsa、pulseaudio和其他一些可用的驱动，但它们在尝试暂停和取消暂停时没有任何区别。我努力寻找在Ubuntu上安装较新版本的fluidsynth的方法，如2.1.0或2.2.4，但目前还没有找到（也许我需要从源码构建）。对于Windows版本，我有最新的2.2.4版本的fluidsynth二进制文件在sf2_loader的包里，所以它工作得很好。

所以总而言之，目前Linux版本在选择SoundFont文件播放MIDI文件时，仍然会使用渲染的音频来播放，配置参数`use_soundfont`仍然存在。换句话说，对于Linux版本，你可以暂时忽略这个更新。在未来，我将尝试找出一个解决方案，以解决Linux上1.9.1版本的fluidsynth的暂停和取消暂停的问题（实际上这是唯一重要的问题，其他功能和在Windows上运行一样正常），也许我会尝试从源码构建，以获得在Linux上运行的最新版本的fluidsynth，或者从某个地方安装一个较新的版本。

2021-11-08

- 改进MIDI键盘演奏模式下钢琴踏板的处理，现在已经支持MIDI CC64, CC66, CC67三种不同的钢琴踏板。
- 增加踩下钢琴踏板时演奏的音符放开后显示另一种颜色的功能。
- 把音符播放结束的方式全部换为渐出，改进音符演奏时的听感。

2021-11-07

- 加入音符条和钢琴键盘的边框显示，演示效果更加美观，可以在配置文件里调整边框宽度和边框颜色。

2021-11-01

- 如果你使用SoundFont演奏, 你可以在演奏时使用按键组合来改变preset编号和bank编号来改变乐器 (当设置按键为默认的`Ctrl`):

  `Ctrl + 1` (上一个preset)

  `Ctrl + 2` (下一个preset)

  `Ctrl + 3` (上一个bank)

  `Ctrl + 4` (下一个bank)

2021-10-31

- 现在可以加载SoundFont文件用来播放MIDI文件，请将设置文件里的`use_soundfont`设置为True，`play_as_midi`设置为True，通过`sf2_path`设置SoundFont文件的路径即可。
  如果使用SoundFont文件，Ideal Piano会在内部使用sf2_loader渲染为音频信息，然后使用pygame播放。
  现在也可以用SoundFont文件进行电脑键盘或者MIDI键盘演奏，请将设置文件里的`play_use_soundfont`设置为True，可以通过`bank`, `preset`等一系列参数自定义选择SoundFont文件里的乐器以及演奏的音符的持续时间，音量等等。
- 使用`Ctrl + S`可以打开修改设置的页面，使用`Ctrl + R`可以重新加载修改过后的参数 (当设置按键是默认的左`Ctrl`)。如果你点击修改设置的页面上的`save`按钮或者按`Ctrl + S`保存设置，设置会在修改设置的页面关闭后自动重新加载。

2021-10-28

- 改进了对于Linux的兼容性，目前在虚拟机内的Ubuntu 18.04.5上面可以正常运行。
  在虚拟机内的macOS Catalina 10.15.5测试还是有非常多比较难以修复的python依赖库不兼容的问题，导致软件运行失败，我会在接下来这几个月尽量找时间对于macOS进行兼容性适配。

2020-12-27

- 新增了一个可以在midi键盘模式下查看当前的电脑的midi设备连接的情况，以及如果你的midi键盘当前设置的midi_device_id连接不上的时候可以查看哪个id是对应着你的乐器的名字。并且会显示错误信息（连接不上的具体原因）。在点击按钮进入midi键盘模式之后，按电脑键盘上的shift键可以显示当前电脑默认的midi设备的id，以及id为0到9分别对应的midi设备的名字，其中一个应该就会是你的乐器了。然后如果你的midi键盘没有连接上，显示的内容的最后也会列出具体的错误信息。按ctrl键可以关掉显示的内容。
- 目前我也已经将软件的midi钢琴读取的代码逻辑改进了很多，现在已经不需要先打开midi键盘再打开软件了，大家可以先打开软件再连接midi键盘并打开midi键盘然后再点击midi键盘模式的按钮就可以识别到midi键盘了（在midi_device_id对上了你的乐器的情况下），中途关掉midi键盘，之后再次打开也可以点击返回按钮然后再重新进入midi键盘模式，软件会重新识别midi键盘，然后你就可以继续演奏了（以前的话就需要重新开一次软件，现在不需要了）

2020-08-20

- 重新设计了钢琴结构，之前是一整张钢琴图片显示在屏幕上，现在加入了绘制钢琴的模式，在设置文件里的draw_piano_keys设置为True即可进入这个模式。在绘制钢琴模式下，midi键盘演奏或者电脑键盘演奏时对应的琴键会亮起，包括在下落音符模式播放midi文件时，音符落在琴键上也会亮起。绘制钢琴是采用直接按照钢琴88键的结构，根据可设定的参数绘制出黑白键，并且每个键可以改变颜色。
  在88键的绘制下面有一张黑色的背景图片，主要用于显示钢琴键之间的缝隙（填充用）。关于设定钢琴键盘绘制的参数的说明，我会写在设置参数说明书里面。可以关掉音符模式，（note_mode设置为不是dots, bars, bars drop的值即可）只打开绘制钢琴模式，演奏时对应的钢琴键位会亮起，播放midi文件时也会亮起当下的音符。也可以使用任何一个音符模式同时打开绘制钢琴模式。

2020-08-17

- 加入了直接以midi文件音源播放的模式，在设置文件里的play_as_midi设置为True即可进入这个模式。在选择合并所有音轨时，会合并所有音轨的音符进行演示，声音是直接在软件内部播放midi文件。如果选择的是其中一个音轨，则临时写入选中的音轨到一个midi文件然后播放。
- 修改设置的程序加入了直接点击输入布尔值（True, False）的按钮，方便修改布尔值的参数。

2020-08-13

- 加入了音符条（下落）模式，只有在播放midi文件的模式中可用，要使用音符条（下落）模式，只要把参数note_mode的值设置成'bars drop'即可。
- 加入了可以选择在和弦显示中不显示排序内容的参数sort_invisible，设置为True则在和弦判断结果为排序类型的时候不会显示和弦的排序内容，只显示和弦名。

2020-08-12

- 实时和弦分析算法加入了poly_chord_first参数，复合和弦模式，可以在和弦非常复杂，判断需要的时间会造成卡顿的时候，把和弦拆解为两个和弦，按照复合和弦的形式来判断（或者一个和弦下面加上一个最低音的形式），可以很大程度上地提高和弦判断的效率，尤其适合在爵士乐演奏中使用这个模式。
- 加入了播放midi文件模式中可以选择去除midi文件的鼓的轨道的参数get_off_drums，设置为True即可在读取midi文件时去掉鼓的轨道，避免在实时分析和弦时被鼓的音轨扰乱。

2020-08-11

- 加入了播放midi文件时可以选择合并所有音轨的功能。
- 修正了背景图片以及钢琴图片的文件路径有时候无法找到的bug。
- 加入了可以选择是否实时分析显示和弦的参数show_chord

2020-08-10

- 全新做了一个更改设置参数的程序，自带搜索栏功能，可以快速查找参数进行修改。
- 我写了一份软件参数说明书，讲解了每一个参数的含义，方便大家修改。
- 原来的默认字体Comic Sans MS显得不太正式，因此我现在把默认字体换成了Cambria，一个非常工整正式的英文字体，把默认的字体大小从20改成了23，并且把按钮的大小变大了（新加入了可设置参数button_resize_num，这个是按钮大小的缩放倍数），因为感觉原来的按钮大小有一些小。
- 新加入了播放midi文件时可以选择合并midi文件的所有音轨然后播放的功能。
- P.S. 我相信Cambria这个字体应该符合大多数人所认可的正式了吧，如果不是，可以到设置文件里自己修改为想要的字体

2020-08-09

- 加入了音符条模式，之前只有音符点显示在钢琴键位上的模式，这次加入的音符条模式相对来说更有欣赏价值一些。音符从键位出现并且上升，无论是自己演奏还是播放midi文件都是这样。自己演奏时如果按着不放，音符条就会一直拉长，放开之后音符条就会往上飘。播放midi文件则是直接根据每个音符的长度算出音符条的相对长度。而且音符的力度对应到了音符的透明度，（这个功能可以在设置文件里对应的参数设置开或关）然后还可以选择出现单色或者随机任意颜色，效果还是不错的。现在这个音符条的模式(bars)作为和之前的音符点(dots)一起存在的两种模式，可以在设置文件里自己设置想要哪种模式。音符的出现位置，音符的长度和宽度，音符每次上升走的步数（走多少个像素），音符的颜色，单色显示或者随机显示颜色，音符的默认透明度，音符的拉长速度等等都可以在config.py里进行设置。
- 我准备把config.py里的所有参数都做到change_settings.py里面，让这些参数全部都可以修改。

2020-08-08

- 默认的和弦显示对齐模式更新为左对齐，取代原来的居中显示，看着更舒服，避免了实时分析中和弦名长度不断变化导致眼睛跟不上的情况。
- 由于我这个软件是全英文的，考虑到很多小伙伴可能看中文的和弦类型名称比较亲切一些，因此我做了一个中文补丁包，变成中文的内容包括：界面的按钮， 显示的和弦种类名称，（原位和弦会保留其他的英文称呼，第一个是中文名称），单音和音程，选择midi文件的界面和每个设置项，更改设置的程序。中文补丁安装包的下载地址在README。