Here we explain the meaning of each parameter in the configuration file config.py, so that you can modify it more clearly.

We will try to cover all the current parameters, but we will focus on the more important ones.

screen_size: screen size (width, height)

background_image: background image file path

background_size: the size of the background image

piano_image: piano image file path

piano_size: piano image size

message_color: font color for chord display, formatted as (R, G, B, A)

fonts_size: font size

label1_place: text position of the currently played note name

label2_place: text position of the chord name

label3_place: text position of the status of the midi file being played

label_anchor_x: horizontal alignment of the text

label_anchor_y: vertical alignment of the text

fonts: font name

bold: whether to bold the text

notes_image: the path to the image of the notes in note point mode

notes_resize_num: the scaling size of the note points

go_back_image: Returns the button's image file path

go_back_place: Returns the image location of the button

self_play_image: the image file path of the computer keyboard's play button

self_play_place: The location of the computer keyboard play button image

self_midi_image: the image file path of the midi keyboard play button

self_midi_place: The location of the midi keyboard play button image

play_midi_image: the image file path of the play midi file button

play_midi_place: the image location of the play midi file button

key_settings: The dictionary for the 88 keys of your computer keyboard. Please note that all the files in the sound path must contain the keys you have set to

reverse_key_settings: dictionary of the 88 keys for the computer keyboard

mode: This is what I used to set before I wrote the UI of this software, whether it was a computer keyboard, a midi keyboard or a midi file, but now this parameter is not used 2333

self_device: It is also used to set the computer keyboard or midi keyboard before writing the UI, but now it is not used

midi_device_id: This parameter is to access the midi device (such as midi keyboard), midi device corresponding to the id, generally speaking, only access to the midi keyboard, not open the arranger software

It is 1 when only the midi keyboard is plugged in, 2 when the sequencer software and loopMIDI are on, and 1 when only the sequencer software and loopMIDI are on without the midi keyboard.

The following are the keyboard shortcut settings for playing midi files

pause_key: the key to pause

repeat_key: key to repeat playback

unpause_key: key to continue (while paused)

exit_key: key to exit the program

pause_key_clear_notes: whether to clear the display of all notes currently playing when paused

show_key: whether or not to show the names of the computer keyboard keys when playing on the computer keyboard

musicsheet: This parameter is also used when the UI is not written yet, when playing midi files, this parameter can put the musicpy language code, the program will play automatically, now

This parameter is also not used anymore

path: This is also the parameter used to set the path of the midi file, but now that the UI can select the file, this parameter is useless.

These two parameters are used to set the position of the track where the midi file is to be played

track_ind

track

bpm: the parameter used to set the speed of the track (BPM), but now that I have written the UI, this parameter is useless

play_interval: used to set a certain part of the track to play, now I wrote the UI and it's useless

#The next few are pygame's audio initialization parameters, which basically don't need to be moved. The larger the maxinum_channels, the less problems will occur when playing multiple notes at the same time

frequency = 44100

size = -16

channel = 1

buffer = 1024

maxinum_channels: the maximum number of channels to play notes

global_volume: the total volume, maximum is 1, minimum is 0

delay: whether to give a certain delay after playing a note

delay_time: the delay time (in seconds)

touch_interval: the interval in seconds between the end of a tone and its replay when the same tone is played continuously

delay_only_read_current: when the tone is played while delayed (the tone is not pressed)

The chord judgment does not include these tones that are still delayed, only the ones that are currently being pressed

sound_format: the file format of the sound source (file extension)

sound_path: the file path of the sound source

show_delay_time: the delay time of the notes when playing the midi file

These are the parameters of the musical logic algorithm for chord determination, and the default settings are the most widely applicable, if I want to explain what they mean.

If I were to explain what they mean, I'd probably need to understand my algorithm first, so I'll explain this part later when I introduce the algorithm

detect_mode = 'chord'

inv_num = False

rootpitch = 5

change_from_first = True

original_first = True

same_note_special = False

whole_detect = True

return_from_chord = False

two_show_interval = True

poly_chord_first = False (When this parameter is set to True, the

(When this parameter is set to True, it will be used as a compound chord for music theory in advance in case of very complex chords, and the chord judgment will be much faster)

show_change_pitch: Up or down modulation of the whole song when playing midi files (positive n is up by n semitones, negative n is down by n semitones)

show_modulation = [original_scale, transposed_scale] Transpose the whole song when playing a midi file

config_enable: whether to enable function keys when playing on computer keyboard

config_key: The key position of the function keys, which can be used with other keys for different functions

volume_up: The key that is used with the function keys to raise the total volume

volume_down: the key that is used with the function key to lower the volume

volume_change_unit: the volume of the total volume that changes each time

change_delay: The key used with the function key to change the delay or not

change_read_current: The key used with the function key to change whether only the currently pressed chord is judged

change_pause_key_clear_notes: key used with a function key to change whether the display of the currently played note is cleared when paused

note_place: The position of all keys on the piano from left to right in note point mode

load_sound: whether to load the sound source and play it when playing (set to False when using with the host)

show_chord: whether the chord is analyzed in real time by the music logic

These are the names of the intervals and the corresponding chromatic numbers

perfect_unison = 0

minor_second = 1

```
augmented_unison = 1

major_second = 2

diminished_third = 2

minor_third = 3

augmented_second = 3

major_third = 4

diminished_fourth = 4

perfect_fourth = 5

augmented_third = 5

diminished_fifth = 6

augmented_fourth = 6

perfect_fifth = 7

diminished_sixth = 7

minor_sixth = 8

augmented_fifth = 8

major_sixth = 9

diminished_seventh = 9

minor_seventh = 10

augmented_sixth = 10

major_seventh = 11

diminished_octave = 11

perfect_octave = 12
```

octave = 12

augmented_seventh = 12

These are the parameters of the algorithm I wrote to separate the major melodies of a tune

melody_tol = minor_seventh

chord_tol = major_sixth

These are some parameters in note bar mode

note_mode: selects the note display mode, currently there are three modes available: note point and note bar (ascending) and note bar (descending, only available in midi file mode).

The corresponding modes are 'dots' and 'bars' and 'bars drop' respectively.

bar_width: the width of the note bar

bar_height: the length of the note bar

bar_color: the color of the bar

bar_y: the vertical coordinate of the bar

bar_offset_x: the pixel value of the horizontal coordinate of the note bar that deviates from the note point position

bar_opacity: the transparency of the note bar, from 0 to 255, from fully transparent to fully opaque

opacity_change_by_velocity: if or not the transparency changes with keystroke force.

The lighter the keypress, the more transparent the note bar is, the heavier the keypress, the more opaque the note bar is

color_mode: the color mode of the note bar, currently there are two modes to choose from, monochrome and random.

These correspond to 'normal' and 'rainbow' respectively (in fact, you can

fill in other text that is not normal)

bar_steps: the number of pixels the note bar moves up each time

bar_unit: the length of the note bar in units for calculating the relative length when playing midi files

bar_hold_increase: The number of pixels that the note bar lengthens each time a key is held down (or a computer key is held down)

bars_drop_interval: in note bar (drop) mode, how long it takes for the bar to drop from the top of the screen to the specified position, in seconds

bars_drop_place: the specified position (height) that the note bar will drop to in note drop mode

adjust_ratio: A parameter that adjusts the accuracy of the bar drop to the specified position, generally not needed

Other parameters

get_off_drums: If True, in midi file playback mode, if you choose to merge all tracks, the drum tracks will be removed after the midi file is read, (if any) to avoid demo chords being scrambled by drum notes.

sort_invisible: if True, the sorting will not be shown in the demo chords (e.g. "Fmaj7 sort as [2,3,1,4]" will become "Fmaj7")

play_as_midi: Play the midi file without loading the source, and play the midi file directly inside the software (with the source that comes with the midi), the advantage is that midi files with more notes load much faster, and the playback will not lag when there are many notes playing at the same time and the chord type is complex. Set to True to enter this mode.

draw_piano_keys: set to True to enter the draw piano mode, (according to the parameters and the structure of the piano 88 keys to draw the piano keyboard, replacing the previous piano picture) In the draw piano mode, the corresponding keys will light up when the midi keyboard is played or the computer keyboard is played, including when the midi file is played in drop note mode, the notes will also light up when they land on the keys . The piano is drawn directly according to the structure of the piano's 88 keys, and the black and white keys are drawn according to settable parameters, and the color of each key can be changed. Underneath the drawing of the 88 keys there is a black background image, which is mainly used to show the

gaps between the piano keys (for filling). You can turn off note mode (note_mode can be set to a value other than dots, bars, bars drop) and just turn on draw piano mode, the corresponding piano key will be lit up when playing and the current note will be lit up when playing the midi file. It is also possible to use any of the note modes and turn on draw piano mode.

white_key_width: the width of the piano's white keys (horizontal length)
white_key_height: the height of the piano's white keys (vertical length)
white_key_interval: the distance between every two white keys of the piano
white_key_y: height position of the white keys of the piano
white_keys_number: the number of white keys of the piano
white_key_start_x: the horizontal position of the first white key of the piano
white_key_color: the color of the piano's white keys

black_key_width: the width (horizontal length) of the piano's black keys
black_key_height: the height of the piano's black key (vertical length)
black_key_y: the height position of the piano's black key
black_key_first_x: horizontal position of the first black key of the piano
black_key_start_x: horizontal position of the second black key of the piano
black_key_color: the color of the piano's black keys

black_keys_set: the relative interval between each black key in each group, except for the first black key, which is set individually, in groups of 5 (the first interval is usually 0, which means that the first black key starts from the leftmost relative position in the current group)
black_keys_set_interval: the interval between every two black keysets
black_keys_set_num: the number of black keysets

piano_background_image: the background image under the piano (for filling the gap)

Ideal Piano can read a text file with a composition analysis in a specific format, showing the transposition, subordinate chords, borrowed chords, etc. in the demo midi file mode according to the current bar, writing a composition analysis txt file and displaying the corresponding composition analysis in the current bar in real time. The default file is musical analysis.txt.
The format of the composition analysis file is
Number of bars1
Composition analysis content1

Number of bars 2
Composition analysis content 2

Number of bars 3
Composition Analysis 3
...
The number of bars here starts with bar 1, and the number of bars is a number, either an integer or a decimal. The composition analysis content is the content you want to display when you reach the specified number of bars. The software will parse the composition analysis file format and find the position of the first note up to the current number of bars, and then display the corresponding composition content when it reaches the corresponding note position during the demo.
Currently, in addition to this format, it also supports displaying the tonicity, so that you can display the current tonicity at any position, and if the tune has a transposition, you can write the tonicity statement before the beginning bar of the transposition. The syntax is (here is an example)
key: Tone 1 (you can write anything you want here, such as A major, A major, etc.)

Number of bars 1
Composition analysis content 1

Number of bars 2
Composition analysis content 2

Number of bars 3
Composition analysis 3

key: key 2 (you can write a new key when the tune is transposed)

Number of bars n
Composition analysis content n

Number of bars x
Composition analysis content x

Number of bars y
Composition analysis content y
...
(The phrase indicating the tonality must be separated from the measure statement by one line, and a measure statement must be adjacent to the corresponding composition analysis statement on the top and bottom of the line, together called a measure block. (Multiple lines can be written within a composition analysis statement, but there must not be a completely empty line in between)
(The number of bars supports both absolute bar position and relative bar

position syntax, absolute bar position is a number, which can be an integer or a decimal, relative bar position syntax is + relative bar length, for example, +1 means the position of the next bar relative to the previous position, +1/2 means the position of the next one-half bar relative to the previous position, relative bar position supports integers, decimals and fractions.)

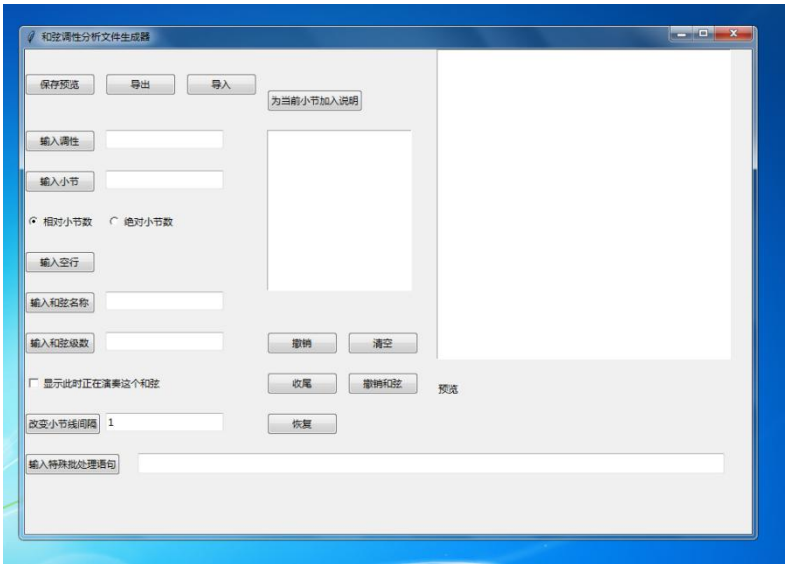
In order to be able to quickly enter a large number of compositional analysis statements, especially when analyzing a piece with complex chord progressions, (I myself generally write the latest 4-5 chords and divide them into 4-5 bar blocks, putting an arrow in front of the chord when each bar block is played to one of the chords, with different compositional techniques explained below according to the actual situation, such as +1)
Emaj9(omit 3) | D#m7 | DM7 | → C#11(omit 3)
IVM9 iii7 bIIIM7 V11 (F# major)
(Ready to transpose, the 2nd chord here is the 5th genus 11 chord in the new key)
I have written a chord tonal analysis file generator, which you can use to quickly enter the number of bars, chord names, chord functions and compositional techniques. You can open the file Chord Tone Analysis File Generator.exe and use it. This software provides automatic alignment of chord functions and chord names, which is very convenient. The software will automatically add all the blank lines needed for the syntax of my own composition analysis files. The generator also supports importing text files for further editing. When you have finished writing the composition analysis, click the "Export" button to export the composition analysis file. Since I personally still think that I can enter the composition analysis statements more quickly, I have designed a special batch input syntax for composition analysis statements, and I have written an algorithm that can parse this new syntax in the generator software, for example, if you enter 3 chords in a bar and the 3 chord function that goes with it, and the composition technique is explained, and the 2nd chord is being played at the moment.
bar number; chord name 1; ! Chord name 2; Chord name 3$ Chord function 1; Chord function 2; Chord function 3$ Composition technique explanation
(1) The number of bars can be chosen as relative or absolute bars, i.e. with or without the plus sign.
2、The analysis of compositional techniques can be written without the $ at the end when it is not written.
3、It is better not to have spaces around the separator; and $, but other places can have spaces.
4. If you need to have more than one line in your composition, you must use the \n line break to connect them, that is, you must write them in one line.
5、At this point, you can add a chord name in front of which chord is being played! marker, but if you don't want to show that any chord is being played,

you don't need to add the ! marker in front of any chord name if you don't want to show that any chord is being played.
6. The algorithm I designed to parse this particular batch statement is to look at the last chord with the ! marker as the chord being played at the moment and adding an arrow to that chord when converting, so you just need to make sure that the chord being played is the last chord with the ! marker, so you only need to make sure that the chord you are playing is the last chord with the ! You don't have to delete the previous chords if they have !
7. After successfully parsing the phrase, the generator software will automatically help you with the layout, aligning each chord name and chord function, and adding the necessary blank lines, which is very convenient.
8、Batch generate a paragraph with several chords syntax:
[n*];chord name1;chord name2;chord name3;... $ chord function 1; chord function 2; chord function 3;...

[n1,n2,n3,...] ;chord name1;chord name2;chord name3;... $ chord function 1;chord function 2;chord function 3;...
The n* in the brackets here sets all the bars of the current line to n. n can be both absolute and relative bars, and relative bars are usually used more often. You can also write n1,n2,n3 in parentheses to set the number of bars for each chord in the current line in turn.
9. Using this special batch syntax I have designed, I can input a large number of large analytic phrases very quickly and concisely, and the syntax itself is very comfortable for non-programmers. (You can also use this special batch statement as a small programming language to write chord function analysis, which I think is still very good :D)

The special batch syntax for entering a statement that displays the tonality is as follows:
k.tonality

Enter the special batch statement in the input field to the right of the "Enter Special Batch Statement" button in the Chord Temperament Analysis File Generator, then click the button and the generator software will parse the statement and enter the corresponding content into the preview on the right. The interface of the chord tonality analysis file generator is as follows:



Related parameters:
show_music_analysis: whether to turn on the display of the composition analysis content
music_analysis_file: the file path of the composition analysis file to be read
music_analysis_place: set the position of the composition content to be displayed
key_header: the beginning of the tune (this parameter shows the beginning of the tune, e.g. "current tune:")
music_analysis_width: the width of the music analysis text label
music_analysis_fonts_size: the font size of the music analysis text

The color of the note bar can also be assigned differently depending on the track and instrument.
Related parameters.
use_track_colors: whether to use different colors for different tracks and instruments
tracks_colors: A list of colors for different tracks and instruments, RGB parameter
use_default_tracks_colors: whether to use the colors of the set tracks, or use randomly generated colors for different tracks

clear_pitch_bend: Clear the bend information