Settings Manual

Here we give you an explanation of what each parameter in the configuration file config.py corresponds to, so that you can be clearer when you modify it.

Contents

Screen related parameters

screen_size: screen size (width, height)

background_image: background image file path

background_size: background image size

width_or_height_first: fill with width or height

background_fix_original_ratio: whether to adjust background image according to the original size ratio

background_opacity: the transparency of background image, from 0 to 255 (transparent to non-transparent)

background_place: position of background image on the screen

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

label_width: max with of the label

fonts: font name

fonts_file: the file path of the font file you want to specify

fonts_path: the directory of the fonts you want to specify

bold: whether to make the font bold

italic: whether to make the font italic

fonts_dpi: the dpi of the font

go_back_image: the image file path of go back button

go_back_place: the position of go back button

self_play_image: the image file path of play button

self_play_place: the position of play button

self_midi_image: the image file path of MIDI keyboard button

self_midi_place: the position of MIDI keyboard button

play_midi_image: the image file path of play MIDI button

play_midi_place: the position of play MIDI button

settings_image: the image file path of settings button

settings_place: the position of settings button

button_resize_num: the resize ratio of button size

button_opacity: the opacity of button

resize_screen: Whether to resize the components when changing the screen size while using the software

fps: frames per second of the screen

key_settings: The dictionary of the 88 keys of the computer keyboard, please note that all the files in the sound path must contain the keys you have set to

midi_input_port: This parameter is the input port id of the MIDI device after it is connected to the MIDI device (e.g. MIDI keyboard)

midi_output_port: the MIDI output port id when playing MIDI files

language: the display language of the software, including main window buttons, chord names, messages and the window to choose MIDI files, currently only 'English' and 'Chinese' are supported

pygame mixer initialization parameters

frequency, size, channel, buffer, max_num_channels, global_volume

(please refer to pygame's mixer module documentation online)

Keyboard playing and MIDI files playing parameters

pause_key: key to pause

repeat_key: key to repeat playback

unpause_key: key to continue (while paused)

move_progress_left_key: button to move the progress bar backward

move_progress_right_key: button to move the progress bar forward

move_progress_left_unit: seconds for each move progress bar backward

move_progress_right_unit: seconds for each move progress bar forward

pause_key_clear_notes: whether to clear the display of all currently played notes when pausing

delay: whether to give a certain delay to the tone after it is released

delay_time: the delay time (in seconds)

fadeout_ms: the time in milliseconds to fade out after the note is played

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

delay_only_read_current: When a note is delayed (not pressed), the chord judgment does not include those notes that are still delayed, only those that are currently pressed

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

sound_path: the file path of the sound source

play_midi_start_process_time: time in seconds to open the process that sends MIDI events when playing MIDI files

move_progress_adjust_time: the delay time in seconds when changing the progress bar

use_soundfont_delay_time: the delay time in seconds when use SoundFont files to play MIDI files under note mode `bars drop`

bars_mode_delay_time: delay time when note mode is `bars`

play_midi_reset_sounds: Whether to mute all sounds when pausing or changing progress while playing MIDI files

mute_all_sounds_cc_number: the MIDI CC number to use when muting all sounds

stop_all_sounds_cc_number: the MIDI CC number to use when stop playing MIDI files

midi_channels_number: the total number of channels in the MIDI file

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

config_key: the key position of the function keys, the function keys can be used with other keys to do different functions

volume_up: the key to use 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 function keys to change whether the display of the currently played note is cleared when paused

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

use_midi_output: whether to connect to MIDI output port when playing

show_key: whether to show the name of the keys of the computer keyboard when playing on it

show_chord: Whether or not the chord is analyzed in real time by musical logic

show_notes: whether to show the current notes in real time when playing

show_notes_delay: the delay of the display of currently playing notes and chord types relative to the notes played

get_off_drums: If True, in midi 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 the demo chords being scrambled by the drum notes.

sort_invisible: if True, the sorted content will not be shown in the display chords (e.g. `Fmaj7 sort as [2,3,1,4]` will become `Fmaj7`)

pitch_range: A tuple of two pitch strings, which filters out the notes between them when reading a MIDI file, to prevent the notes from exceeding the pitch range set by the piano.

soft_pedal_volume: The ratio of the reduction of the volume of the notes when the weak pedal is pressed in MIDI keyboard playing mode, a decimal number between 0 and 1

Chord analysis parameters

These are the parameters of the chord logic algorithm, the default settings are the most widely used, so if I want to explain what they mean, I may need to understand my algorithm first, so I will explain this part later when I introduce it.

change_from_first = True

original_first = True

same_note_special = False

whole_detect = True (changing this parameter to `False` will improve the speed of real-time analysis, but very complex chords may not be analyzed)

poly_chord_first = False

show_degree = True

Parameters of the algorithm for separating the main melody of a piece

melody_tol, chord_tol, get_off_overlap_notes, average_degree_length, melody_degree_tol

Note display related parameters

note_mode: Select the note display mode, currently there are 2 modes to choose from: note bars rising and note bars falling (only available in MIDI file mode), corresponding to '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, RGB tuple

sustain_bar_color: the color of the bar when the key is released while the piano pedal is pressed in MIDI keyboard play mode, RGB tuple

progress_bar_color: the color of progress bar

progress_bar_opacity: the opacity of progress bar

bar_y: the vertical coordinate of the bar's appearance

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

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 of the bar changes according to the keypress force, the lighter the keypress force, the more transparent the bar is, the heavier the keypress force, the more opaque the bar is

color_mode: the color mode of the note bar, currently there are two modes to choose from, monochrome and random, corresponding to 'normal' and 'rainbow' respectively (actually, you can fill in other text that is not normal)

bar_steps: the number of pixels the note bar moves each time it rises

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

bar_border: the width of the bar's border bar_border_color: the border color of the note bar, it is the RGB tuple, `(R, G, B)`.

use_track_colors: whether to use different colors on different tracks and instruments

tracks_colors: list of colors for different tracks and instruments, RGB parameter

use_default_tracks_colors: whether to use set track colors or use randomly generated colors for different tracks

show_chord_accidentals: show notes and chord types in sharp or flat accidentals, the values could be `'sharp'` or `'flat'`, the default value is `'sharp'`

show_chord_details: whether to show chord details of current chord you are playing

chord_details_label_place: text position of chord details

chord_details_label_anchor_x: horizontal alignment of chord details

chord_details_label_anchor_y: vertical alignment of chord details

chord_details_font_size: font size of chord details

chord_details_label_width: max width of chord details

show_current_detect_key: whether to analyze and show some of the most possible keys you are currently playing (this functionality is experimental and still under development)

current_detect_key_show_note_count: whether to show current notes counts

current_detect_key_label_place: text position of current key

current_detect_key_label_anchor_x: horizontal alignment of current key

current_detect_key_label_anchor_y: vertical alignment of current key

current_detect_key_font_size: font size of current key

current_detect_key_label_width: max width of current key

current_detect_key_algorithm: the key analysis algorithm to use, currently there are 3 algorithms to choose, corresponding to 0, 1, 2

current_detect_key_major_minor_preference: config parameter of current key analysis function, whether to put major and minor as the most possible result

current_detect_key_most_appear_num : config parameter of current key analysis function, the number of possible keys

current_detect_key_limit: the max number of notes to analyze when playing, if current number of notes exceed this limit, then clear the temp

current_detect_key_unit: the unit in bars to slice a piece in ranges, for the third key analysis algorithm (detect_scale3)

current_detect_key_key_accuracy_tol: the minimum accuracy in float of percentage of a part of piece to be detected as a key, for the third key analysis algorithm (detect_scale3)

Piano keyboard related parameters

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: the height position of the piano's white keys

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

white_key_opacity: the opacity of 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_key_opacity: the opacity of 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

show_note_name_on_piano_key: show note name on piano keys or not

show_only_start_note_name: show only note names with C or all white keys note names on piano keys

piano_key_note_name_font_size: font size of note names on piano keys

piano_key_note_name_bold: bold or not of note names on piano keys

piano_key_note_name_italic: italic or not of note names on piano keys

piano_key_note_name_color: color of note names on piano keys

piano_key_note_name_pad_x: x padding of note names on piano keys

piano_key_note_name_pad_y: y padding of note names on piano keys

piano_key_border: the width of the piano's keyboard border

piano_key_border_color: piano_key_border_color, RGB tuple, `(R, G, B)`

piano_background_image: the background image under the piano (to fill the gap)

piano_background_opacity: the opacity of piano background image

SoundFont file related parameters

use_soundfont: Whether to use SoundFont files to play MIDI files, when this is set to True, rendering the MIDI file to be played to audio with the loaded SoundFont file to play when playing the MIDI file

play_use_soundfont: Whether to use the SoundFont file when playing on a computer keyboard or a MIDI keyboard

sf2_path: The path of the SoundFont file

bank: the bank number of the SoundFont file

preset: the preset number of the SoundFont file

sf2_duration: the length of the notes generated by the SoundFont file, in seconds

sf2_decay: the fade-out time of the notes generated by the SoundFont file, in seconds

sf2_volume: the volume of the note generated by the SoundFont file, in units of MIDI note velocity

sf2_path: path to the SoundFont file

sf2_mode: on some computers you need to set this to 1 to play using SoundFont files without crashing

Show composition analysis

Ideal Piano can read a text file with a composition analysis in a specific format, and display the composition techniques such as modulation, subordinate chords, borrowed chords, etc. in the demo MIDI file mode according to the number of bars in the current composition. 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 piece has a modulation, you can write the tonicity statement before the beginning bar of the modulation. 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 piece is modulated)

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 input 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 reaches one of the chords, with different compositional techniques explained below, such as

+1

Emaj9(omit 3) | D#m7 | DM7 | → C#11(omit 3)

IVM9 iii7 bIIIM7 V11 (F# major)

You can use the editor of music_analysis_batch_language to generate music theory analysis statements according to the batch syntax of README .

Using this batch syntax I designed, you can input a large number of music analysis statements very concisely and quickly, and the syntax of the music analysis statements itself is very comfortable for non-programmers to read directly. So you can also use this special batch statement as a small programming language to write chord function analysis, I think it's still very good :D

Related parameters

show_music_analysis: whether to turn on the display of the composition analysis

music_analysis_file: the file path of the composition analysis file to read

music_analysis_place: set the position of the composition content to be displayed

key_header: the beginning of the key (this parameter shows the beginning of the key, e.g. "current key:")

music_analysis_width: the width of the music analysis text label

music_analysis_fonts_size: the font size of the music analysis text