Themera Help and Documentation

Version 2.1.0

Table of Contents

Introduction

What's Themera?

Themera is a theme code generator for PySimpleGUI.

It enables you to create themes based on any of the existing themes that comes built in with PySimpleGUI, edit any custom existing theme based on the dictionary containing its colors, or create a theme from an image.

After editing, simply copy your theme code, ready to use in your project without need for alteration.

It is – of course – built with PySimpleGUI, free and open source under the LGPL v3 license, and comes with a wide range of features from batch color manipulation, to 13 filters that simulate color blindness, auto-contrast, automatic dark and light modes for themes and more.

Getting Started

The Launcher

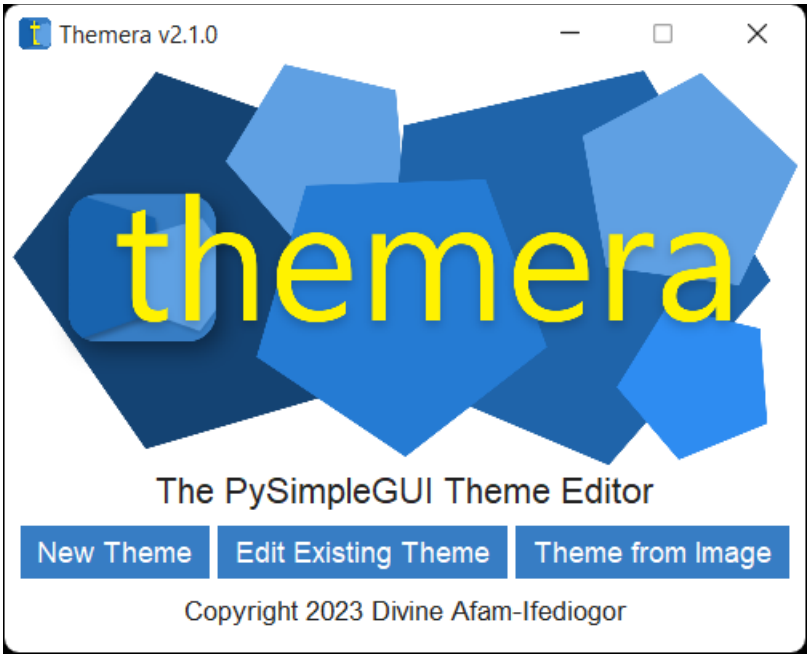


Figure 1: The Launcher in light theme

The launcher is the first window that comes up when Themera is opened.

There are three ways to get started:

- `New Theme`: This loads up a built-in theme of your choice as a base to craft your new theme from.
- `Edit Existing Theme`: This lets you input the theme dictionary of any existing theme you want to use that as a base for your theme.
- `Theme From Image`: This allows you to select a local image to base the colors of your theme on.

New Theme

Clicking `New Theme` switches to this view:

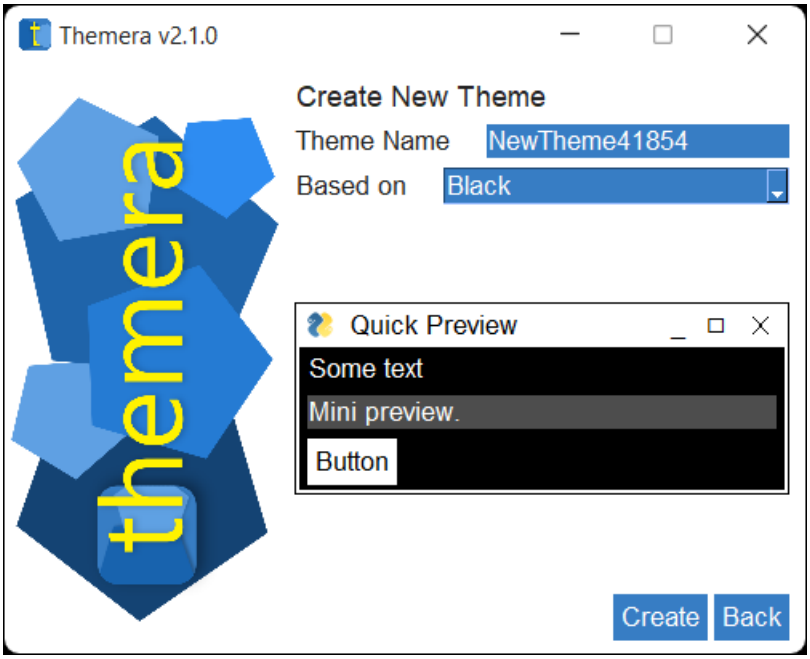


Figure 2: New Theme

- Enter your theme within the `Theme Name` slot. Feel free to name it whatever you want; you can always change it later.
- Select any theme from the `Based On` dropdown to serve as the base for your theme; Black is the default, but only because the list is in alphabetical order. The Quick Preview will update automatically as you select your desired theme.
- Click `Create` to continue.

Edit Existing Theme

Clicking `Edit Existing Theme` will switch to this view:

Figure 3: Edit Existing Theme

- Give your theme whatever name you want in the `Theme Name` slot. You can always change it later if you wish.
- Paste in the existing theme dictionary in the big textbox. It must be pasted exactly as it appears in your Python code, otherwise there will be errors. This is an example of a valid theme dictionary (the theme dictionary for the `PythonPlus` theme):

```
{
    'BACKGROUND': '#001d3c',
    'TEXT': '#ffffff',
    'INPUT': '#015bbb',
    'TEXT_INPUT': '#fed500',
    'SCROLL': '#015bbb',
    'BUTTON': ('#fed500', '#015bbb'),
    'PROGRESS': ('#015bbb', '#fed500'),
    'BORDER': 1,
    'SLIDER_DEPTH': 1,
    'PROGRESS_DEPTH': 0,
}
```

Figure 4: The Theme Dictionary for PythonPlus

- Click `Create` to continue.

Theme From Image

Clicking this option switches to this view:

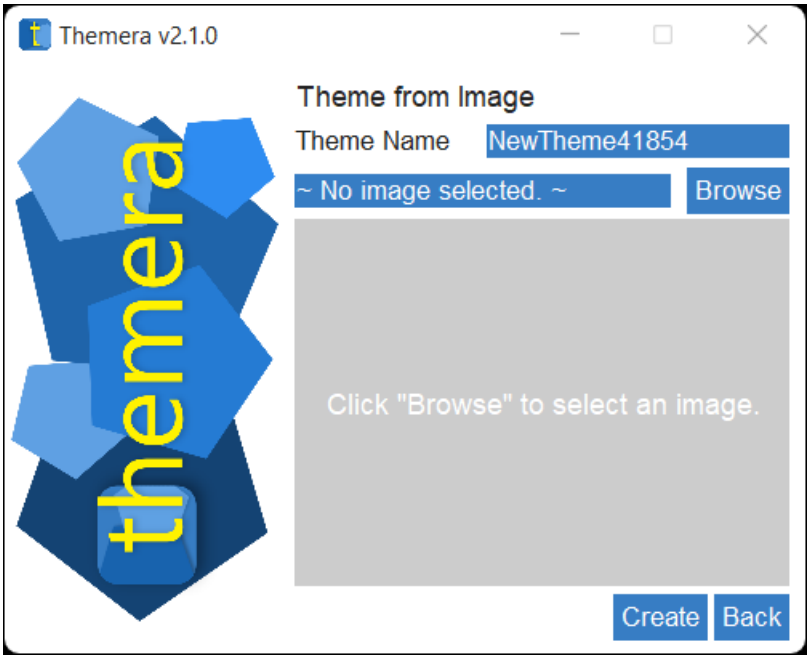


Figure 5: Theme From Image

- Enter whatever name you want for your theme; it can always be changed later.
- Type in a local filepath to your desired image, or click `Browse` to bring up a system file browser, from which you can navigate and select the image. The image will be loaded and a thumbnail of it will be displayed in the preview area (the check-pattern grid). Depending on the file size of your image and your computing power, the preview process may take a while. Also, not every image may be used successfully. If an error occurs because of an invalid image, try converting the image to a different file format and trying it again.
- Click create to continue.

The Editor

The Editor is where the bulk of the theming process takes place.

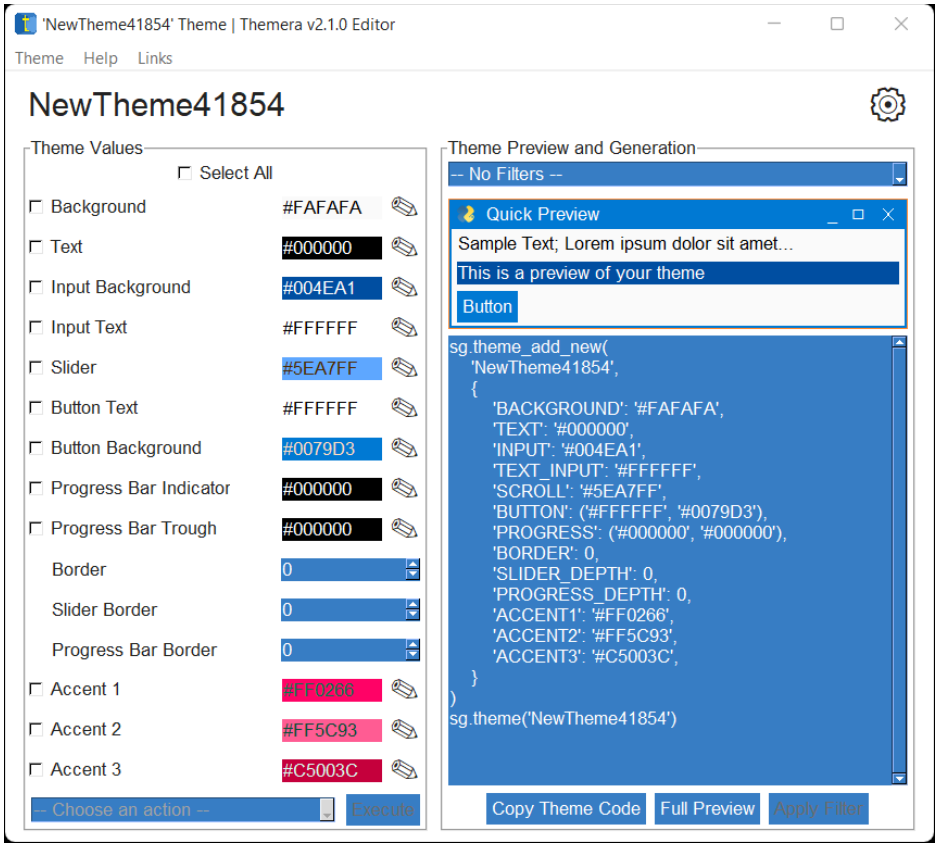


Figure 6: The Editor (New theme based on Material2)

It consists of four main sections:

- The menu bar (topmost),
- The name and settings section (beneath the menu bar),
- The `Theme Values` section (left), and
- The `Theme Preview and Generation` section (right).

The Menu Bar

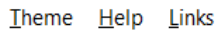


Figure 7: The Menu Bar

This bar contains some useful actions in the following order:

- Theme
  - Create New Theme (Ctrl/Cmd + N)
    - Reopens a launcher window with the Create New Theme view.
  - Edit Existing Theme (Ctrl/Cmd + Shift +N)
    - Reopens a launcher window with the Edit Existing Theme view.
  - Return to &Launcher (Ctrl/Cmd + Alt/Option + L)
    - Reopens a launcher window with the Default view.
  - Settings (Ctrl/Cmd + Shift + S)
    - Opens a Settings window
  - Revert to Beginning (Ctrl/Cmd + Alt/Option + R)

        - Returns the theme being edited to its initial, base state without modification.
- Help
    - Themera Help (F1)
        - Opens help (this document)
    - Report Issue on GitHub ⇗
        - Opens a new browser tab in the default browser and navigates to a `New Issue` page on Themera's GitHub.
    - PySimpleGUI Docs ⇗
        - Opens a new browser tab in the default browser and navigates to PySimpleGUI Docs.
    - View valid color names
        - Opens a window showcasing all valid, Tkinter-supported color names that can be used in Themera. Not to be confused with color shorthands.
- Links
    - Visit Themera's GitHub Page ⇗
        - Opens a new browser tab in the default browser and navigates to the Themera GitHub Repo.
    - Developer's GitHub Profile ⇗
        - Opens a new browser tab in the default browser and navigates to the GitHub profile of Themera's developer.

The Name and Settings section

Figure 8: The name and settings section.

This section houses mainly two elements:

- The name of the theme currently being edited, and
- A button leading to a `Settings` window.

Changing the Theme Name

To change the name of the theme, click on it.

Figure 9: The theme name shows a pencil icon when hovered over.

This will reveal an entry box for you to type in. Then, click on any other UI element or press Return to exit it.

Theme Values Section

This section is where the main editing takes place. Each entry represents a key-value pair in the final theme dictionary.

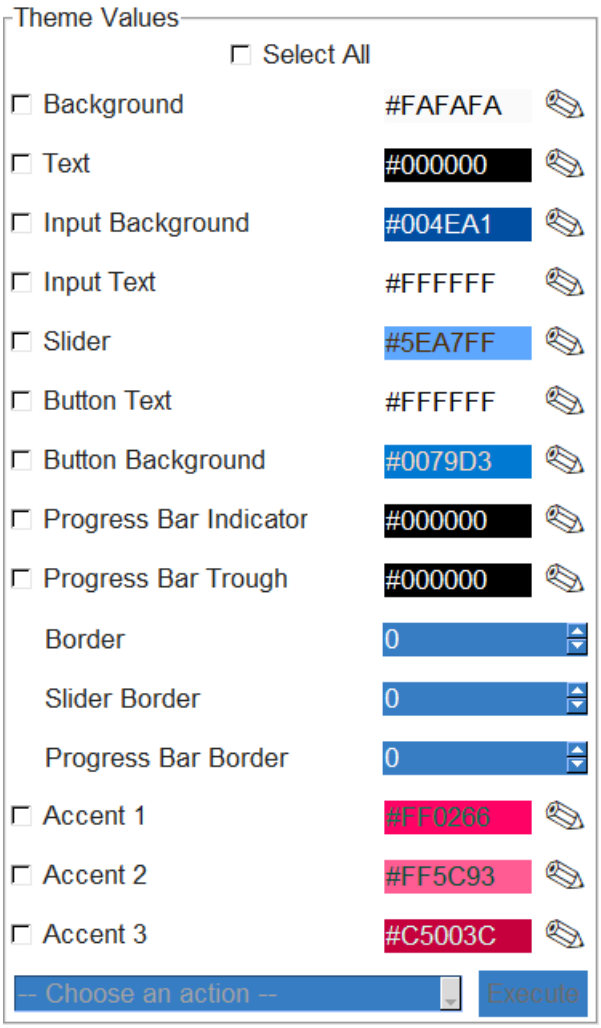


Figure 10: Theme Values Section

Color values

Figure 11: An example of a color value.

Color values are displayed with a checkbox denoting their name, an entry for their value and a pencil icon to choose colors. By default, the color values are displayed as the background of their entry. This behaviour is officially termed `Colorboxes`, and can be disabled in `Settings`.

The colors can be edited by any of three ways:

- Typing directly into the input box,
- clicking the pencil icon ( ) beside, or
- selecting multiple colors with their checkboxes and choosing `Random Color (All)` or `Random Color (Individual)` from the `-- Choose an action --`, then clicking `Execute`.

Numeric Values

Figure 12: An example of a numeric value.

Numeric values are shown with a Spin element. Nothing too fancy. Floating point numbers are treated similarly, but with increments of 0.1.

Non-Color, non-numeric Values

These are displayed with simple entry elements and are treated like ordinary Python objects – using quotes irresponsibly with strings for example will cause errors.

Errors

In this context, errors are not fatal to the program's execution, rather, it means a value is invalid. For all values in the Theme Values section, entering an invalid value will trigger a warning sign, and the theme and preview will not update until those errors are resolved.

Figure 13: An example of the warning sign for invalid entries.

Note that errors are not triggered only by color values; even numeric and non-color values can trigger warnings.

Select All

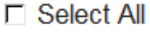


Figure 14: The Select All checkbox

This checkbox can be used to check all the boxes for the color values for a theme all at once. It can also be triggered by the keyboard shortcut Ctrl/Cmd + Shift + A.

Batch Actions

Figure 15: The Batch Actions dropdown

When *two or more* color values are selected, the Batch Actions (`-- Choose an action --`) dropdown gets activated. Upon selection of an action, the `Execute` button will become active, and to actually carry out a selected action, this button must be clicked, or its keyboard shortcut Ctrl/Cmd + Shift + E must be pressed.

This dropdown contains 7 actions:

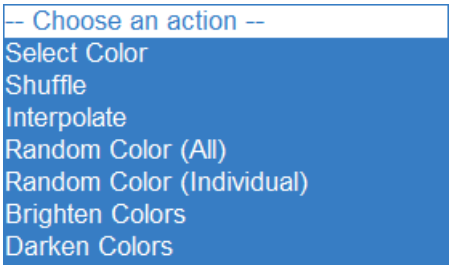


Figure 16: All 7 Batch Actions

- Select Color: A color picker will be invoked and all checked color values will have the selected color written as their value.
- Shuffle: Shuffles the colors around.
- Interpolate: Takes the colors at the extremes of the selection and fills in the rest with intermediate colors according to a linear gradient between those extremes.
- Random Color (All): Picks a random color and applies that to all selected colors.
- Random Color (Individual): Picks a random color for each one of the selected colors.
- Brighten Colors: Increases the luminance of all selected colors.
- Darken Colors: Decreases the luminance of all the selected colors.

Theme Preview and Generation Section

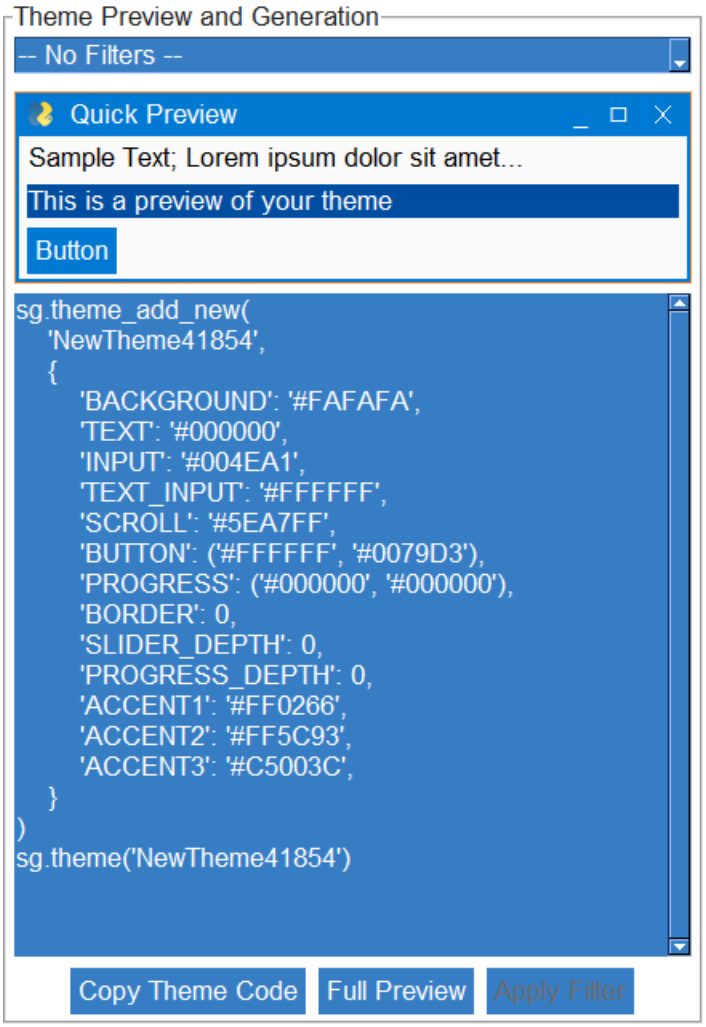


Figure 17: Theme Preiew and Generation.

This section has 4 main parts:

- The Filters
- The Quick Preview Panel
- Theme Code Preview
- Quick actions.

Filters

Figure 18: The Filters Dropdown

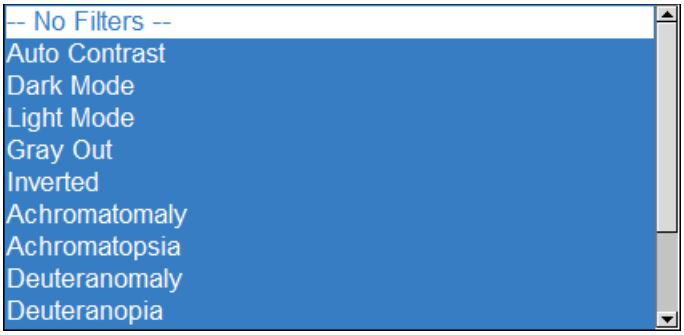


Figure 19: 9 filters.

Filters are an important part of Themera, which alter the entire look of a theme's color. There are 13 of them, 8 of which simulate color blindness:

- Auto-Contrast: Adjusts the luminance of each color to reach an appropriate level of contrast.
- Dark Mode: Rearranges the colors and adjusts their luminances to form a dark theme.
- Light Mode: Rearranges the colors and adjusts their luminances to form a light theme.
- Gray Out: Reduces the saturation of all colors to zero.
- Inverted: Inverts all colors.
- Achromatomaly, Achromatopsia, Deuteranomaly, Deuteranopia, Protanomaly, Protanopia, Tritanomaly, Tritanopia: Color blindness simulations.

Selecting a filter from the dropdown will immediately show its effects on the `Quick Preview` panel, however it won't change the color values themselves, or the theme code preview. To do so, click the `Apply Filter` (Apply Filter) button.

How do filters work?

This section is not essential to actually knowing how to use Themera to design themes, so if you do not wish to learn the inner workings of the filter system, you can safely skip this.

Themera filters are of three main types, from which all of the 13 filters are derived.

- Individual
- Index
- Transform

Individual Filters

Individual filters carry out a function on each color of all the color values of the theme. These could be as simple as inverting each color, the Gray-Out filter or brightening and darkening colors.

Index Filters

Index filters take a list of floating point numbers each ranging from 0 to 1, then rearranging the color values according to those numbers. They essentially rearrange the colors in a given theme according to positions (each position is floating point number ranging from 0 <start position; index 0> to 1 <end position; last index>) listed in an index list.

To give a simple example, given an `index` list `[0.2, 0.8, 0.6, 0.4, 0.5]` and a list of 5 colors (extracted from the theme) as `['black', 'red', 'yellow', 'cyan', 'white']`, an index filter will rearrange those colors to give: `['black', 'cyan', 'yellow', 'red', 'yellow']`. The second color in the re-arranged list is 'cyan',

because the second number in the index list is 0.8, and 0.8 multiplied by 5 is 4, and the 4/5th color is 'cyan'.

Filters of this type include:

- Auto Contrast
- Dark Mode
- Light Mode

…and each of the index lists behind their behaviour can be edited in the Settings.

Transform Filters

These filters perform an RGB color transform on each color according to a specified transformation matrix.

For example, given the transform matrix for Tritanopia $\begin{bmatrix} .95 & .05 & 0 \\ 0 & .43333 & .56667 \\ 0 & .475 & .525 \end{bmatrix}$, and an input RGB value of $\begin{bmatrix} .2 \\ .7 \\ .65 \end{bmatrix}$, (or (51, 179, 166) in RGB format and #33b3a6 in hex) the transform filter obtains the final RGB matrix of $\begin{bmatrix} (.2 \times .95)+(.7 \times .05)+(.65 \times 0) \\ (.2 \times 0)+(.7 \times .43333)+(.65 \times .56667) \\ (.2 \times 0)+(.7 \times .475)+(.65 \times .525) \end{bmatrix} = \begin{bmatrix} .225 \\ .40334 \\ .67375 \end{bmatrix}$, which is equivalent to (57, 103, 163) in RGB format or #3967a3 in hex format – a close enough approximation to simulate the effect of Tritanopia (blue-green confusion).

Quick Preview Panel

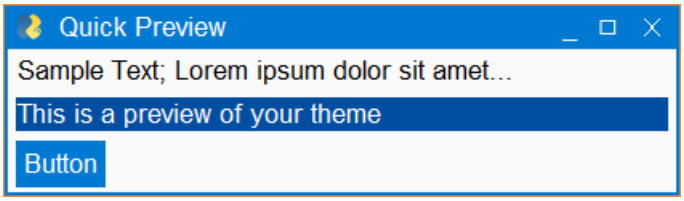


Figure 20: The Quick Preview Panel

This is essentially a simulated window with a PySimpleGUI Custom Titlebar that shows a real-time preview of the theme or any active filters. The titlebar buttons are non-interactive, the input element is interactive, and the regular button is clickable but does nothing.

It is meant to serve as quick visual feedback on the state of your theme, however if it is inadequate, clicking the `Full Preview` button at the bottom of the window will bring up one of 3 full-fledged previewer types. See the Full Preview section for more details.

Theme Code Preview

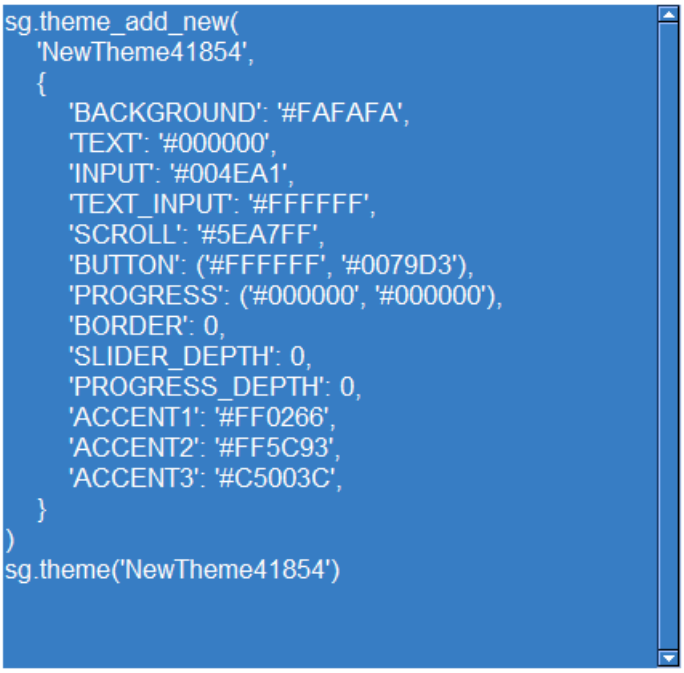


Figure 21: Theme Code Preview

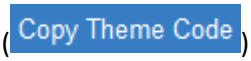
This is a read-only textbox that shows the actual code that results from the theme being edited. The code will always contain a `theme_add_new` call, with the name of the theme and an auto-generated theme dictionary, as well as a call to set the theme. The alias for PySimpleGUI is "sg" by default (just like the actual PySimpleGUI documentation), but it can be changed from Settings. The theme code will **not** reflect any un-applied filters.

Quick Actions

These are the set of buttons at the bottom of the Theme Preview and Generation Section.

Figure 22: The Quick Action buttons

Copy Theme Code

This button (

) copies the contents of the Theme Code Preview to the clipboard, to be pasted in your PySimpleGUI project. It is the primary means of "exporting" the finished product, due to the fact that themes aren't files that can be saved or exported in the traditional sense.

Full Preview

Clicking the 

button activates one of 3 larger previewers. The one that gets activated specifically is determined by the `Full Preview Mode` setting in Themera Settings.

- Default:

  Opens up a larger preview window with more widgets that displays the theme currently being edited along with any active filters.

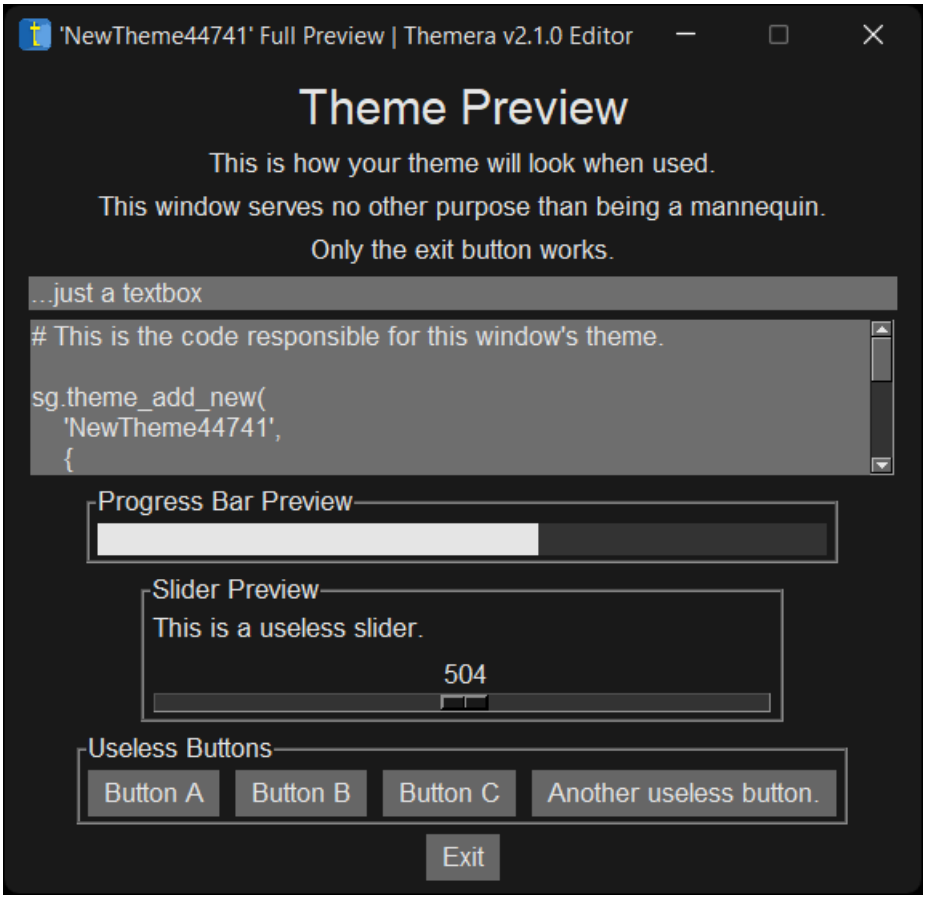


Figure 23: A Full Preview window (theme based on ThemeraDark with the Gray Out filter active.)

- Palette:

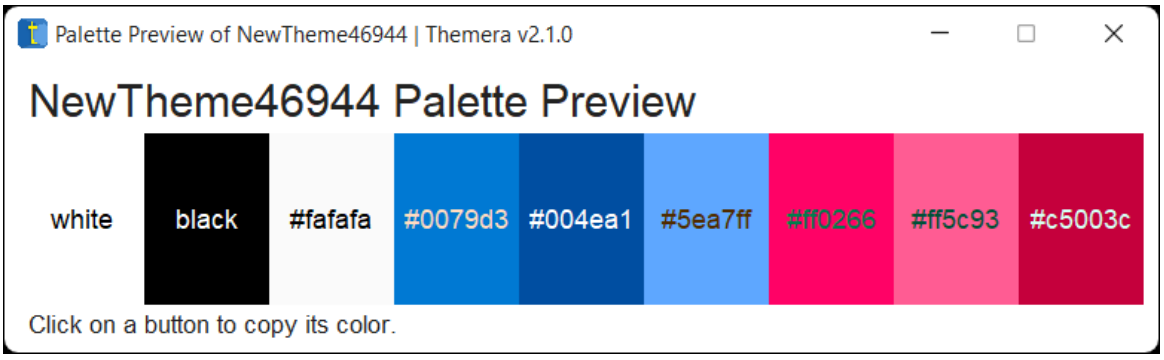


Figure 23: The Palette Preview window.

This preview mode shows all the colors in the theme (no filters applied) in a card-like fashion.

- Custom:

This preview mode allows for entry of a custom PySimpleGUI window layout to use as a preview window.

First, a window appears to take your desired layout as input

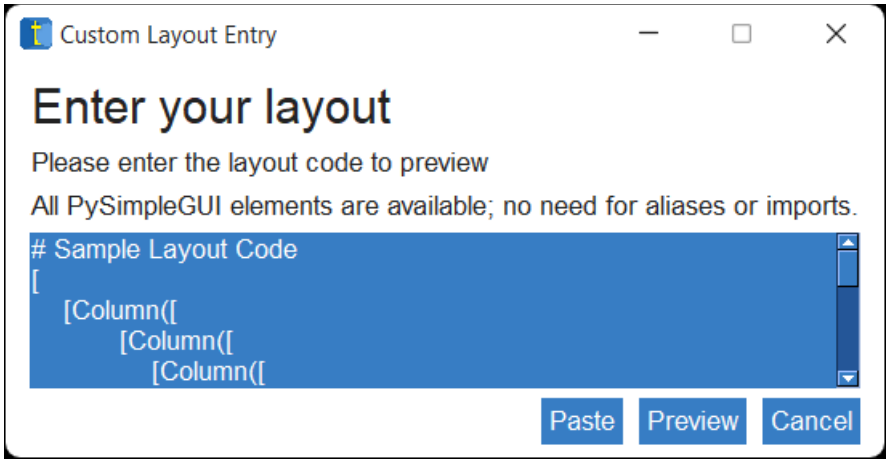


Figure 24: Layout entry

Make sure the layout data supplied is an ordinary layout – a list of lists, and use the names of the elements directly, without any aliases to PySimpleGUI. There are options to Paste from the clipboard, go ahead with the Preview or Cancel the operation.

When done, click Preview to continue.

The default layout results in this preview (for the Material2 theme):

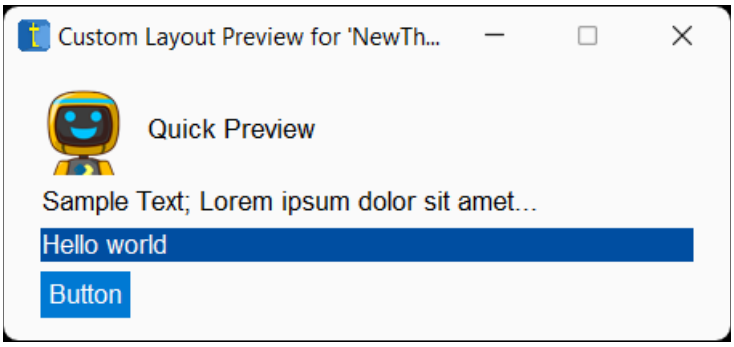


Figure 25: Custom preview for the default given layout.

Apply Filter

Clicking the Apply Filter button (when it is active) will apply the effects of the chosen filter to your theme. The color values will change to reflect this, and so will the Theme Code Preview.

Keyboard Shortcuts

This section details all the keyboard shortcuts usable in the Themera Editor.

Keyboard Shortcut	Action	Description
Ctrl/Cmd + Shift + A	Select All	Selects all color values in the editor window
Ctrl/Cmd + Shift + C	Copy Theme Code	Copies the current theme code to the clipboard
Ctrl/Cmd + Shift + E	Execute	Executes the currently selected batch action.
Ctrl/Cmd + P	Full Preview	Activates the full preview mode.
Ctrl/Cmd + N	Create New Theme	Opens the launcher to

		create a new theme.
Ctrl/Cmd + Shift + N	Edit Existing Theme	Opens the launcher to create a theme based on the theme dictionary of any existing theme.
Ctrl/Cmd + Alt/Option + N	Theme From Image	Opens the launcher to create a theme based on the colors in an image.
Ctrl/Cmd + Alt/Option + L	Reopen Launcher	Opens the launcher on the main screen.
Ctrl/Cmd + Shift + S	Open Settings	Opens a Settings window.
Ctrl/Cmd + Alt/Option + R	Revert to Beginning	Restores the values of the theme currently being edited to their original (source) form.
F1	Themera Help	Opens this document.

Color Shorthands

If you want a color, just type in its name first. There is likely a shorthand for it, if Tkinter doesn't natively support that color name.

The supported color names in Tkinter may not fully support the normal spectrum of color names that the user might require. For example, the color "burgundy" (#800020) is not officially supported. For this reason, Themera comes with 280 additional color shorthands from "heliotropepurple" to "wenge" to "persimmon". Simply type in the name of the color, all lowercase, no spaces.

If there is a valid shorthand for it, Themera will automatically replace the color name with the equivalent hex value (which Tkinter can handle).

Otherwise, you'll just have to supply the hex value yourself.

Settings

Themera Settings can be accessed by three ways:

- Clicking the Settings icon () in the Name and Settings section
- Using the keyboard shortcut Ctrl/Cmd + Shift + S.
- Selecting Theme > Settings from the menu bar.

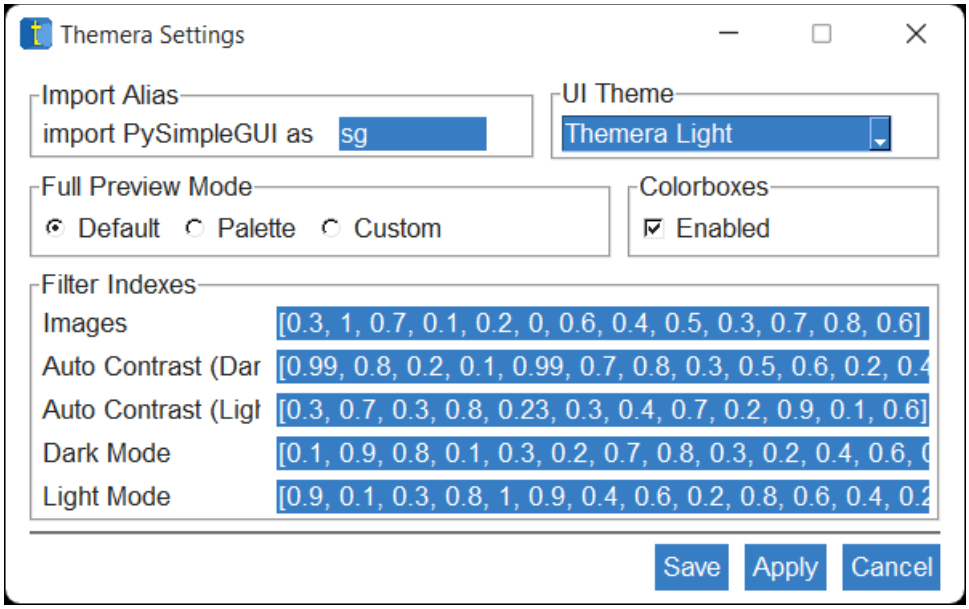


Figure 26: The Settings Window

Import Alias

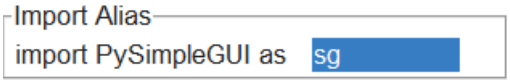


Figure 27: Import Alias Setting

This setting determines what the Theme Code Preview will refer to the PySimpleGUI import as. By default it is set to “sg”.

UI Theme

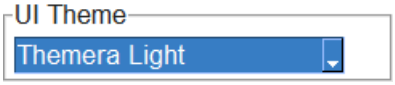


Figure 28: The UI Theme dropdown

This setting determines the theme that Themera itself will sport. Of course, its own dark and light themes are available choices, with the default trying to match the system’s theme where possible, thanks to the `darkdetect` package. If you wish though, the full array of built-in PySimpleGUI themes is available, and changing themes is as easy as selecting a new one from the dropdown and clicking `Apply` or `Save`. Changes take effect immediately thanks to the `reskinner` package.

Full Preview Mode

This setting determines which type of previewer gets activated by the Full Preview Quick Action Button.

Colorboxes

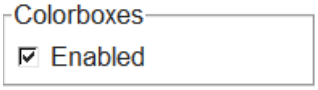


Figure 29: The Colorboxes checkbox

Themera has the ability to use the individual color values of the theme currently being edited as the background colors of their input elements. This behaviour is enabled by default and is officially known as Colorboxes. The text color of those inputs will also be calculated from the color value, so there may be text visibility issues or harsh contrasts with certain colors. If the effects are undesirable, the feature can be disabled entirely by unchecking the checkbox.

Filter Indexes

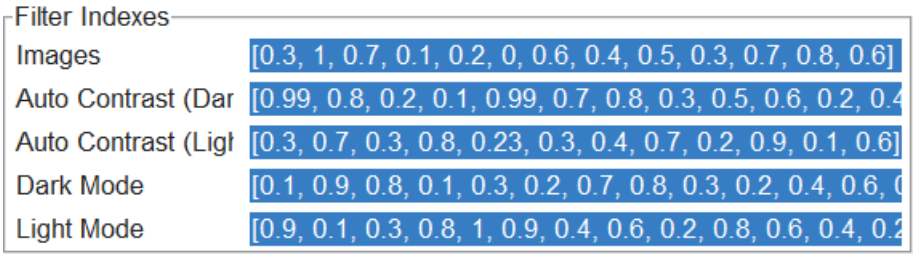


Figure 30: Filter Indexes

Altering this setting requires an understanding of how index filters work.

That said, each index list controls the behaviour of their respective filters.

Save And Apply

Saving

If any settings are changed from the defaults, saving settings will first Apply them, then save them to a persistent file on disk that Themera can reference in future runs. If the settings file gets deleted or Themera can't find it when it needs to, it will revert to the defaults.

Applying

Applying settings will apply their effects immediately to all open Themera windows part of the current execution, but those effects won't persist after Themera is closed completely. To persist them, use the Save button.

Coming from Themera v1.0

Themera v2.0 is a major, breaking leap from Themera v1.0. As such, this section is meant to help previous users adapt to the new changes.

Figure 31: Version 1's interface

Tabs

Figure 32: The Tabs, with the Specifier Tab active.

Firstly, the tabbed interface is gone. The functionality of each tab is still there in Version 2, but the approaches and usages are different.

The Specifier Tab

In v1, the Specifier Tab was where the main editing of theme values took place (see the figure here). Each value had a hard-coded counterpart in the UI, which proves inadequate when compared to modern PySimpleGUI themes such as LightGreen10:

```
sg.theme_add_new(
    'NewTheme12435',
    {
        'BACKGROUND': '#dde1de',
        'TEXT': '#dde1de',
        'INPUT': '#dde1de',
        'TEXT_INPUT': '#dde1de',
        'SCROLL': '#dde1de',
        'BUTTON': ('#dde1de', '#dde1de'),
        'PROGRESS': ('#dde1de', '#dde1de'),
        'BORDER': 1,
        'SLIDER_DEPTH': 0,
        'PROGRESS_DEPTH': 0,
        'COLOR_LIST': ['#205d67', '#639a67', '#d9bf77', '#d8ebb5'],
        'DESCRIPTION': ['Blue', 'Green', 'Brown', 'Vintage'],
    }
)
sg.theme('NewTheme12435
```

Figure 33: The theme code for LightGreen10

Hence, the v2 Editor structures the UI to match the theme it's working on, dynamically. The editor window also handles all the functionality of the Specifier tab directly.

Re-Use Theme Code

Figure 34: The "Re-Use Theme Code" Tab

This feature has been restructured into one of the three methods of creating a theme from the Launcher in v2: Edit Existing Theme.

Other Options

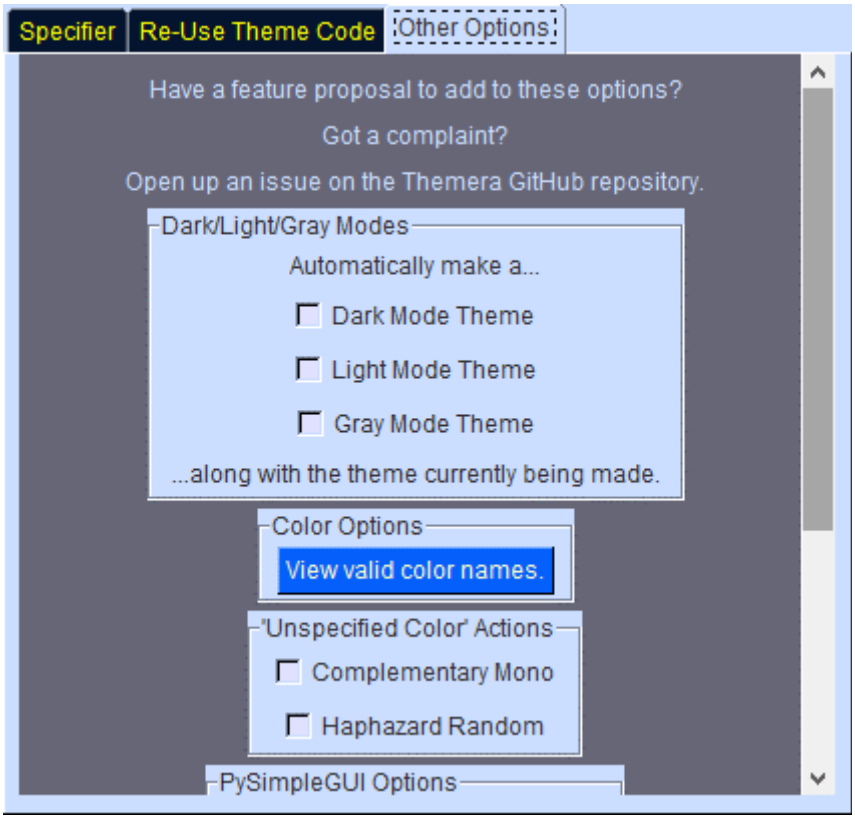


Figure 35: The "Other Options" tab

The options in this tab have been reworked into different forms or removed entirely in some cases:

Dark/Light/Gray Modes

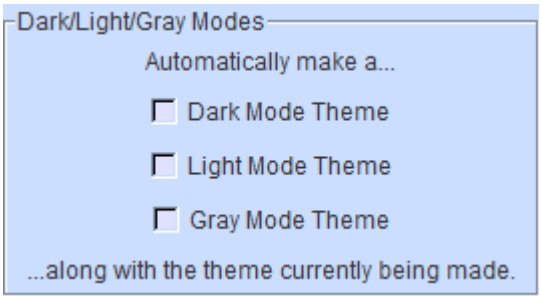


Figure 36: Dark/Gray/Light Modes

This option's effects can be obtained in v2 via Filters.

Color Options

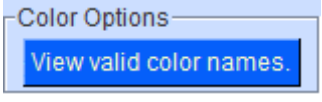


Figure 37: Color Options

The option to view all valid color names is accessible in v2 from the menu-bar.

‘Unspecified Color’ Actions

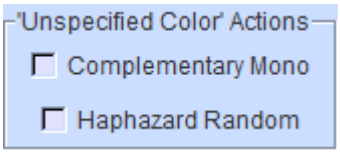


Figure 38: Unspecified Color actions

- The Complementary Mono option has been removed completely in v2, but it may be re-introduced in a future version of v2 if there is sufficient demand for it.
- The effects of the ‘Haphazard Random’ can be obtained by selecting two or more color values and using the `Random Color (Individual)` batch option.

PySimpleGUI Options

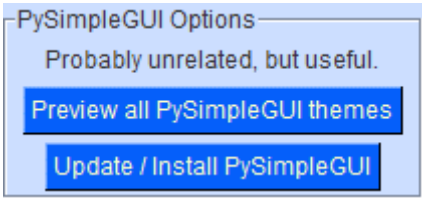


Figure 39: PySimpleGUI Options

- The previewer for all PySimpleGUI themes has been removed, mainly because any theme you wish to see in action can be set as Themera’s own theme or form the basis of a new theme directly.
- Updating and Installing PySimpleGUI itself directly from Themera was deemed to be out of the scope of the project, and has been removed with no alternative.

External Links

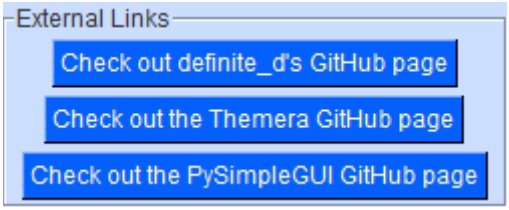


Figure 40: External Links

All external links have been relocated to the menu-bar under Links.

“Click this text for help”

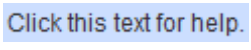


Figure 41: “Click this text for help”

The help text at the top of the window along with the logo have been removed. Help can be obtained in v2 by either clicking Help > Themera Help from the menu-bar or using the keyboard shortcut F1, both of which will open this document.

Current Theme Mode

Figure 42: Current Theme Mode

Previously in v1, the theme mode for Themera had to be set from the dropdown at the bottom of the window, and changes would take effect after a restart. This behaviour has been overhauled by leveraging the reskinner package which enables instantaneous theme changing, and it is now accessible from Settings.

Unspecified Colors

In version 1, trying to generate theme code with any color values not specified would result in a prompt like this:

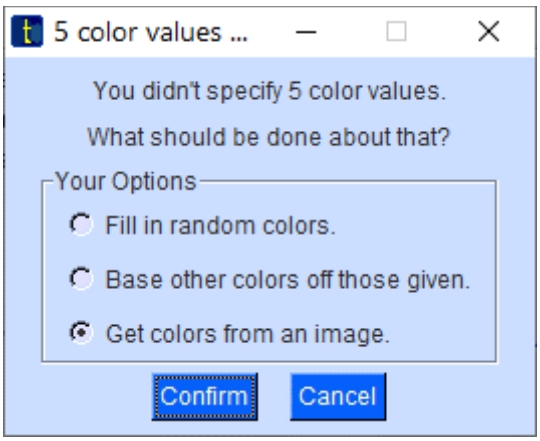


Figure 43: Unspecified Colors prompt

Fill in random colors

In v1, this option would fill in all unspecified color values with randomly chosen colors. If the `Haphazard Random` option was enabled, each color value would get its own random color. Otherwise, they would all receive the same random color.

Base other colors off those given

This option would consider the colors that were actually specified, and interpolate them to fill in any missing values. This behaviour can be found in v2 in form of the “Interpolate” batch action.

Get colors from an image

This option would bring up a popup to locate an image file, from which colors would be extracted and used to fill in the unspecified color values, a feature that was known as ImagePalette. In v2, this has been replaced by the ability to base a new theme off the colors in an image: Theme from Image

Generating themes – The Output Window

Generating themes in v1 would bring up this window:

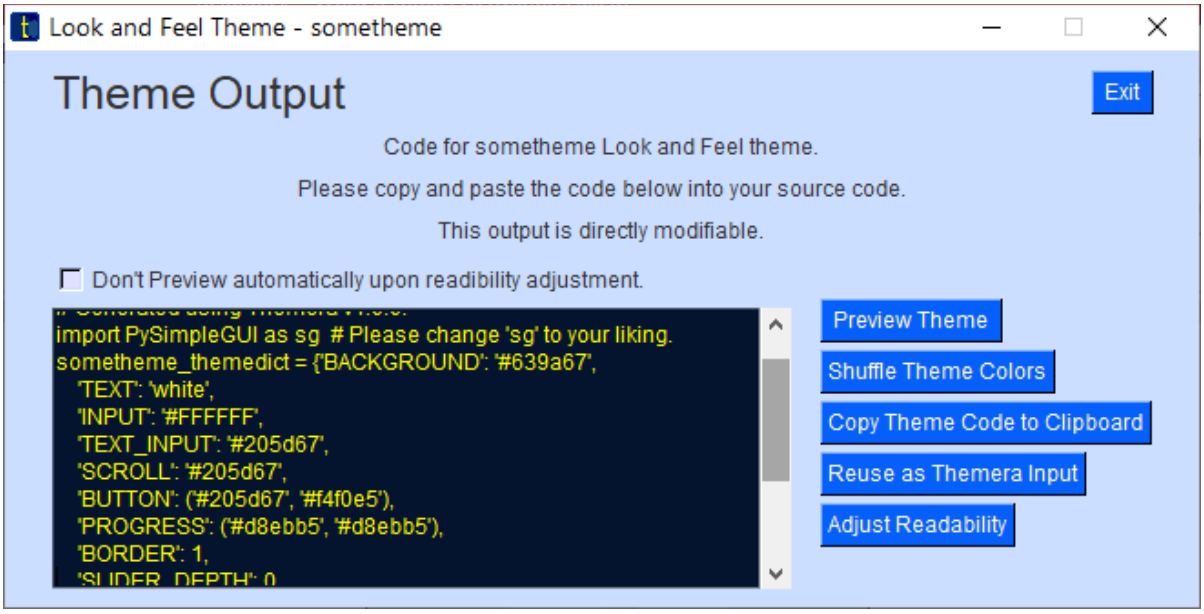


Figure 44: Version 1's Output Window

- The output textbox has been made redundant by the theme code preview in v2.
- The Preview Theme functionality is the same as version 2's `Full Preview`, with the added Quick Preview Panel feature as well. The main difference is, having multiple full preview windows open simultaneously is not yet supported.
- Shuffling theme colors is represented in v2 as a Batch Action.
- Copying theme code is still the same as in v2.
- Reusing as Themera Input was necessary in v1 because the output was fully detached from the Specifier and editable, so it removed the need to close the Output Window to make quick alterations. The structure of v2's editor reflects valid input changes in the theme code immediately without the need for an external output window, so this feature has been removed.
- Adjusting Readability of generated themes has been replaced by the Autocontrast Filter.