Programmatic Generation of Native Procreate Swatch Files

1. Introduction

Procreate has established itself as a leading digital art application for the iPad, widely adopted by artists and illustrators for its intuitive interface and powerful features. Color palettes are integral to the digital painting process within Procreate, facilitating consistent color usage, streamlining creative workflows, and enabling artists to express their unique visual styles. The ability to programmatically create native Procreate swatch files (.swatches) offers a pathway to automation, customization, and integration with external systems, addressing the needs of technically inclined users seeking enhanced control over their color workflows. This report will explore the landscape of programmatically generating these files, delving into the native file format, examining available software development kits (SDKs), libraries, and tools, and providing practical examples for implementation. The analysis will also consider advanced topics such as including color names and converting from other industry-standard color palette formats.

2. Understanding Procreate's Color Palette Ecosystem

2.1 Procreate's Native Swatch Format (.swatches)

The native file format utilized by Procreate for storing and sharing color palettes is the .swatches file [1]. This format allows users to save collections of colors for use within their artwork. In addition to its native format, Procreate also exhibits interoperability with other design applications by supporting the import of Adobe Swatch Exchange (.ASE) and Adobe Color files (.ACO) [1]. These Adobe formats are prevalent within the broader design ecosystem, particularly among users of the Adobe Creative Suite. Procreate's inclusion of support for these formats indicates a strategic approach to facilitate the exchange of color palettes across different software platforms, thereby increasing user flexibility in managing their color resources. While the application supports these external formats, the primary focus of this report will remain on the native .swatches format, directly addressing the user's query and providing a foundational understanding necessary for programmatic generation.

2.2 Manual Palette Creation and Export in Procreate

Within the Procreate application, users can manually construct custom color palettes through the color panel [1]. This process involves various methods for color selection, including utilizing the traditional color wheel, employing an eyedropper tool to sample colors directly from the canvas or imported images, creating palettes automatically from images sourced from the device's camera, photo library, or files, and saving the currently selected color as a new swatch. Once a palette is created, Procreate provides an intuitive method for exporting it as a .swatches file [1]. This is achieved through a simple drag-and-drop action within the Palettes tab of the color panel. By touching and holding a palette, users can drag it to a compatible location within the iPadOS file system, such as the Files app, effectively exporting it as a .swatches file. This user-friendly export mechanism suggests that the .swatches file is designed for ease of sharing and portability within the iPadOS environment, likely facilitating workflows that involve distributing palettes among users or utilizing them across multiple Procreate projects. While the official Procreate Handbook offers comprehensive guidance on how to manage and utilize color palettes within the application [1], it does not provide any technical specifications or details regarding the internal structure of the .swatches file format itself. This absence of official documentation necessitates an exploration of community-driven research, reverse engineering efforts, and the functionalities of third-party libraries to gain the necessary understanding for programmatic interaction with the .swatches format.

3. Deconstructing the .swatches File Format

3.1 The Zipped JSON Structure

A key finding in understanding the .swatches file format is that these files are essentially compressed archives containing JSON (JavaScript Object Notation) data [8]. This information has emerged from discussions within the Procreate community forums, indicating a level of reverse engineering and knowledge sharing among users. The use of JSON, a widely adopted and human-readable data format, as the underlying structure is a significant factor in enabling programmatic interaction. Numerous libraries across various programming languages are readily available for creating, reading, and manipulating JSON data. The structured nature of JSON, with its key-value pairs, is well-suited for representing the data associated with a color palette, such as the palette's name and the collection of color definitions. The compression of this JSON data into a zip archive likely serves to optimize the file for storage and sharing by reducing its size. This compression might also incorporate additional mechanisms for ensuring file integrity, such as metadata or checksums. It is crucial to note the disclaimer provided by Procreate regarding the .swatches file format [8]. They explicitly state that the internal structure may be subject to change in future updates and that any modification or manipulation of these files is undertaken entirely at the user's own risk, without any official support. This implies that developers who choose to programmatically generate .swatches files should be aware of the potential for future compatibility issues if Procreate decides to alter the underlying file format.

3.2 Likely Components of the JSON Structure

Based on the functionalities observed in Procreate and the information provided by community-developed libraries that interact with .swatches files, it is possible to infer the key components likely present within the unzipped JSON structure:

- **Palette Name:** A string value that represents the name assigned to the color palette as it appears within the Procreate application [1]. This name serves as a primary identifier, allowing users to easily locate and organize their color collections.
- **Color Data:** An array or list that contains the definitions for each individual color swatch included in the palette [9]. Each entry within this array likely corresponds to a single color and encompasses the specific color values along with information specifying the color space used to define those values.
- **Color Space Information:** An indication of the color model employed for each swatch. Strong evidence suggests that Procreate internally stores color information using the HSB (Hue, Saturation, Brightness) color model, which is functionally equivalent to the HSV (Hue, Saturation, Value) model [9]. The procreate-swatches JavaScript library explicitly confirms this by stating that Procreate stores swatches in HSB/HSV by default. Therefore, when programmatically creating .swatches files, it is essential to either represent the desired colors in the HSB/HSV color space or to utilize libraries that can handle the necessary conversion from other common color spaces such as RGB or hexadecimal.
- **Swatch Names (Potentially):** While earlier versions of Procreate might have displayed swatch names based on their corresponding hexadecimal values, more recent versions of the application allow users to assign custom names to individual color swatches [1], particularly when the palette is viewed in the "Cards" layout. The inclusion of custom swatch names enhances the user experience by enabling more descriptive labeling of colors within a palette. If the goal of programmatic creation is to achieve full compatibility with the latest Procreate features, incorporating the ability to include individual swatch names might be a desirable consideration.

The most direct method for gaining a definitive understanding of the .swatches file structure would involve manually exporting a color palette from Procreate, subsequently unzipping the resulting .swatches file, and examining the contents of the enclosed JSON file. However, it is imperative for users to exercise caution when undertaking such an endeavor and to remain mindful of Procreate's explicit warning against tampering with these files [8]. While this hands-on approach can yield valuable insights into the actual data structure utilized by Procreate, it should be pursued with the understanding that the format is subject to change without notice.

4. Leveraging Existing Libraries and Tools for Programmatic Creation

4.1 procreate-swatches JavaScript Library (npm)

The procreate-swatches JavaScript library, available through the Node Package Manager (npm), provides functionalities for both reading and creating Procreate .swatches palette files [9]. Installation is straightforward using the command npm install procreate-swatches. This library offers two primary functions:

- readSwatchesFile(data): This function is designed to parse the contents of an existing Procreate .swatches file. The data argument is mandatory and accepts the file content in various formats, including base64, text, binarystring, array, uint8array, arraybuffer, blob, and nodebuffer. The optional colorSpace argument allows developers to specify a target color space for the parsed colors, with 'hsv' set as the default. Given that Procreate internally utilizes the HSB (HSV) color model, leaving this argument blank will retain the colors in their native representation. Supported color spaces for conversion include rgb, hsl, hsv, hwb, xyz, lab, and lch. The function returns an object containing two properties: name, a string representing the palette's name, and colors, an array of color swatches. Each color in the colors array is represented as an array itself, following the format [[...colorValues], colorSpace] (e.g., ['rgb',]). It is also possible for elements within the colors array to be null, indicating an empty swatch position within the palette. The function may throw a ProcreateSwatchesError if the provided file is not a valid .swatches file or if the requested target color space is not supported. This function is particularly useful for developers who need to analyze or manipulate existing .swatches files, enabling tasks such as extracting color information for use in other applications or converting palettes to different formats.
- createSwatchesFile(name, colors, format = 'uint8array'): This function enables the programmatic creation of new Procreate .swatches files. It requires two primary arguments: name, a string specifying the desired name for the palette, and colors, an array containing the color definitions to be included in the palette. The colors array should follow the format] (e.g., ['rgb',]). It is important to note that while various color spaces can be provided as input, the library will internally convert them to HSB for storage within the .swatches file. The colors array can also

include null values to represent empty slots in the palette. A Procreate palette has a limitation of 30 colors; any colors beyond this limit will be ignored by the library. The optional format argument specifies the desired format for the returned file content, with 'uint8array' being the default. Supported formats include base64, text, binarystring, array, uint8array, arraybuffer, blob, and nodebuffer. The function returns the raw content of the newly created .swatches file as an arraybuffer. It may throw a TypeError if the provided name is not a string or if the colors array does not adhere to the expected format. Additionally, a ProcreateSwatchesError may be thrown if an unsupported color space is provided or if the color values are deemed invalid. This function provides a relatively simple interface for generating .swatches files, abstracting away the intricacies of the underlying file format structure and the necessary color space conversions.

The procreate-swatches library can be utilized in both Node.js and browser environments, offering versatility for various programmatic palette creation scenarios. For instance, in a Node.js environment, one might read an existing palette file using the promises API from the fs (file system) module, as demonstrated in the following snippet [9]:

JavaScript

```javascript
import { readSwatchesFile } from 'procreate-swatches';
import { promises as fs } from 'fs';

(async () => {
 const data = await fs.readFile('path/to/palette.swatches');
 const swatches = readSwatchesFile(data);
 console.log(swatches);
})();
```

Similarly, for creating a new .swatches file in Node.js, the createSwatchesFile function can be used [9]:

JavaScript

```
import { createSwatchesFile } from 'procreate-swatches';
import { promises as fs } from 'fs';

(async () => {
 const swatchesFile = createSwatchesFile('My Palette', ['rgb', ],
   ['rgb', ],
   ['rgb', ]);
 await fs.writeFile('my_awesome_palette.swatches', Buffer.from(swatchesFile));
 console.log('Palette created successfully!');
})();
```

These examples illustrate the basic usage of the library in a server-side context. The library also provides similar functionality for browser-based applications, enabling web-based tools for palette generation.

4.2 procreate-swatch-generator JavaScript Library (npm)

Another JavaScript library available on npm is procreate-swatch-generator, which focuses specifically on generating Procreate swatches files from a list of colors [10]. This library can be installed either as a project dependency using $ npm install --save procreate-swatch-generator or globally via $ npm install --global procreate-swatch-generator for command-line access.

The primary API function is procreateSwatchGenerator(colors, [options]). The colors argument is mandatory and expects an array of valid color strings. These color strings can be in various formats, including hexadecimal (e.g., '#000'), RGB (e.g., 'rgb(128, 128, 128)'), RGBA, HSL, or HSLA. It is important to note that any alpha channel information (from RGBA or HSLA) will be discarded, as Procreate swatches do not account for color opacity. The optional options argument is an object that can contain configuration parameters. The -f or --filename option allows specifying the filename for the generated .swatches file (without the extension). If this option is not provided, a filename will be automatically generated. The -o or --output option can be used to define the directory where the .swatches file should be written. If not specified, the library will attempt to infer the output path from the filename or default to the current working directory. The procreateSwatchGenerator function returns a PassThrough stream object, which is a type of stream in Node.js that can be piped to another destination, such as a file using fs.createWriteStream. This stream-based approach is efficient for handling color data and allows for flexible integration with other Node.js stream operations.

The procreate-swatch-generator library also offers a command-line interface (CLI). Once installed globally, users can generate .swatches files directly from the terminal. For example, the command $ procreate-swatch-generator '#000' 'rgb(128, 128, 128)' will create a file named My Awesome Swatch.swatches in the current directory containing black and gray colors. Users can also specify the filename and output directory using the -f and -o options, as in $ procreate-swatch-generator '#000' 'rgb(128, 128, 128)' -f "$HOME/Dropbox/Goth Drab". This command will generate a file named Goth Drab.swatches in the user's Dropbox folder. Furthermore, the output can be piped to another process or destination using standard shell redirection, such as $ procreate-swatch-generator '#000' 'rgb(128, 128, 128)' -f 'Goth Drab' > './Goth Drab.swatches'. Similar to the procreate-swatches library, this tool also enforces the 30-color limit per Procreate palette, ignoring any colors beyond the thirtieth in the input array. The CLI provides a convenient and quick way to generate .swatches files, making it suitable for scripting and rapid palette creation.

4.3 procreate-swatches Ruby Gem

For developers working within the Ruby ecosystem, the procreate-swatches gem offers similar capabilities for interacting with Procreate .swatches files [11]. This gem allows users to parse existing .swatches color palettes into Ruby objects, extract the colors in various formats, and generate new Procreate color palettes from an array of colors, exporting them to .swatches files. The gem can be installed by adding gem 'procreate-swatches' to the project's Gemfile and running the $ bundle command, or by installing it directly using $ gem install procreate-swatches.

A key aspect of this Ruby gem is its utilization of the Chroma gem [11] for color manipulation. The Chroma gem is a powerful Ruby library that provides extensive support for various color formats and color space conversions. This integration ensures robust and accurate color handling within the procreate-swatches gem. The primary functionalities are accessed through the Procreate::Swatches module. To use the gem, the main file needs to be required using require 'procreate/swatches'.

The gem provides several methods for interacting with .swatches files. Existing files can be parsed using Procreate::Swatches::Parser.call(file_path), which takes the path to a .swatches file as an argument and returns a Procreate::Swatches::Wrapper instance. This wrapper object provides access to the palette's name via wrapper.name and the array of colors via wrapper.colors. The wrapper.colors method can optionally take a format parameter (e.g., :hex, :rgb, :hsl) to retrieve the colors in a specific representation. By default, each color in the returned array is an instance of the Chroma::Color class, providing access to a wide range of color manipulation methods. For generating new .swatches files, the gem offers the methods Procreate::Swatches.export(name, colors) and its alias Procreate::Swatches.to_file(name, colors). Both of these methods take a palette name (string) and an array of colors as arguments. The colors array can contain color values in any format supported by the Chroma gem (e.g., hex codes, color names, RGB values). The methods then generate a .swatches file with the specified name and colors. A basic example of creating a .swatches file using the Ruby gem is as follows [12]:

Ruby

```ruby
require 'procreate/swatches'

palette_name = 'My Ruby Palette'
colors = ['#FF0000', '#00FF00', '#0000FF', '#FFA500']

output_path = Procreate::Swatches.export(palette_name, colors)
puts "Palette created at: #{output_path}"
```

This Ruby gem offers a similar level of functionality to the JavaScript libraries discussed, providing developers in the Ruby ecosystem with the necessary tools to programmatically interact with Procreate color palettes.

Table 1: Comparison of Programmatic Libraries

Library Name	Programming Language	Installation Method	Key Features	Ease of Use	Link to Repository
procreate-swatches	JavaScript	npm install procreate-swatches	Read/Write .swatches, Color space conversion	Moderate	https://github.com/szydlovski/procreate-swatches

			(various), Node.js & Browser		
procreate-swatch-generator	JavaScript	npm install procreate-swatch-generator	Create .swatches, Accepts various color formats, CLI support	Easy	https://github.com/natecavanaugh/procreate-swatch-generator
procreate-swatches	Ruby	gem install procreate-swatches	Read/Write .swatches, Color manipulation via Chroma gem, Various formats	Moderate	https://github.com/laurentzziu/procreate-swatches

5. Illustrative Code Examples

5.1 JavaScript Example (using procreate-swatches)

This example demonstrates creating a Procreate .swatches file named "My Awesome Palette" containing four colors defined in different formats (RGB, Hex, HSL) using the procreate-swatches library in a Node.js environment. It includes basic error handling for file writing.

JavaScript

```javascript
import { createSwatchesFile } from 'procreate-swatches';
import { promises as fs } from 'fs';

async function createPalette() {
  const paletteName = 'My Awesome Palette';
  const colors = ['rgb', ],        // Red
    ['hex', ['00FF00']],        // Green (represented as a hex string array)
    ['hsl', ],      // Blue
    ['rgb', ]        // Orange
  ];

  try {
    const swatchesFile = createSwatchesFile(paletteName, colors);
    await fs.writeFile('my_awesome_palette.swatches', Buffer.from(swatchesFile));
    console.log('Palette created successfully!');
  } catch (error) {
    console.error('Error creating palette:', error);
  }
}

createPalette();
```

In this example, the colors array contains color definitions using the 'rgb', 'hex', and 'hsl' color spaces. The createSwatchesFile function will handle the conversion of these colors to the HSB format used by Procreate. The resulting swatchesFile is an ArrayBuffer which is then converted to a Buffer before being written to a file.

5.2 Ruby Example (using procreate-swatches)

This Ruby example utilizes the procreate-swatches gem to create a .swatches file named "My Ruby Palette" with four colors specified as hexadecimal color codes.

Ruby

```ruby
require 'procreate/swatches'

palette_name = 'My Ruby Palette'
colors = ['#FF0000', '#00FF00', '#0000FF', '#FFA500']

output_path = Procreate::Swatches.export(palette_name, colors)
puts "Palette created at: #{output_path}"
```

Here, the Procreate::Swatches.export method simplifies the process of creating the .swatches file. The gem, leveraging the Chroma library, can handle the hexadecimal color codes provided in the colors array and generate the appropriate .swatches file.

6. Advanced Considerations and Potential Extensions

6.1 Including Color Names

While Procreate allows users to rename individual swatches within a palette, especially in the "Cards" view [1], the programmatic inclusion of these individual color names during the creation of a .swatches file through the explored libraries appears to have limited direct support based on the readily available documentation. Some tools, like the swatches-to-ase converter [14], mention handling color names during conversion to the Adobe Swatch Exchange format, suggesting that the concept of associating names with colors in a palette file is relevant within the broader ecosystem. Further investigation into the source code of the procreate-swatches JavaScript [15] and Ruby [12] libraries might reveal undocumented functionalities or specific data structures for including swatch names. If direct support is lacking, potential workarounds could involve creating the base palette programmatically and then manually adding the desired color names within the Procreate application itself. Future enhancements to these libraries might include explicit support for associating names with individual color swatches during programmatic creation, which would further enhance their utility.

6.2 Converting from Other Formats (.ASE, .ACO)

Procreate's native ability to import Adobe Swatch Exchange (.ASE) and Adobe Color files (.ACO) directly [1] provides a degree of interoperability. However, the programmatic conversion of these formats to the native .swatches format using the discussed libraries is not explicitly detailed in their primary documentation. The existence of projects like joanroig/swatches-to-ase [14], which focuses on converting .swatches files to .ASE, indicates a community interest in format conversion. This suggests that the reverse conversion, from .ASE or .ACO to .swatches, might be feasible, potentially through other specialized tools or by gaining a deeper understanding of the internal structures of both the Adobe formats and Procreate's native format. If direct conversion capabilities are not currently offered by the procreate-swatches libraries, a possible approach for developers would be to find or develop parsers for the .ASE and .ACO file formats (if specifications are publicly available) and then utilize the createSwatchesFile function of the JavaScript or Ruby libraries to generate the corresponding .swatches file based on the extracted color data. Programmatic conversion between these formats would significantly enhance the flexibility of managing color palettes across different design applications.

6.3 Limitations: Maximum 30 Colors

Both the procreate-swatches JavaScript library [9] and the procreate-swatch-generator [10] explicitly mention the limitation of a Procreate palette to a maximum of 30 colors. When using these libraries to create .swatches files, any colors provided beyond this limit will be ignored. For scenarios requiring a larger set of colors, a programmatic strategy could involve automatically splitting the color set into multiple .swatches files. These files could be named descriptively to indicate their relationship, such as "My Palette Part 1.swatches", "My Palette Part 2.swatches", and so on. This approach allows developers to manage and utilize extensive color schemes within Procreate's constraints.

6.4 Exploring Other Potential Tools or APIs

While the procreate-swatches and procreate-swatch-generator JavaScript libraries and the procreate-swatches Ruby gem are prominent examples of community-developed tools for programmatic interaction with Procreate color palettes, it is possible that other smaller or more specialized tools or scripts exist within the broader digital art and development communities. Users seeking alternative solutions are encouraged to explore online repositories such as GitHub and community forums dedicated to Procreate and digital art for additional resources. It is important to reiterate that Procreate does not currently offer an official SDK or API for direct interaction with its file formats [1]. Therefore, the programmatic methods discussed in this report rely on community-driven reverse engineering and library development. This also implies that the long-term stability and compatibility of these methods are subject to the ongoing efforts of the community and any potential future changes that Procreate might implement in its file formats.

7. Conclusion

The programmatic creation of native Procreate swatch files (.swatches) is achievable through the utilization of community-developed libraries, primarily within the JavaScript and Ruby ecosystems. Libraries such as procreate-swatches and procreate-swatch-generator for JavaScript, and the procreate-swatches gem for Ruby, provide developers with the necessary tools to generate .swatches files based on defined color data. These libraries abstract away the complexities of the underlying file format, which is essentially a zipped JSON structure, and handle color space conversions to ensure compatibility with Procreate's internal HSB color model. The benefits of programmatic palette creation include the potential for automation, customization based on specific algorithms or data sources, and integration with other tools and workflows. However, it is crucial for users to acknowledge the unofficial nature of these methods and the explicit warning from Procreate regarding tampering with its file formats. As such, any programmatic manipulation of .swatches files is undertaken at the user's own risk, and awareness of potential future compatibility issues is advised. Despite these considerations, the availability of these community-driven resources empowers developers and technically inclined artists to enhance their digital art workflows and explore the possibilities of automated color palette generation for Procreate.

Works cited

1. Palettes — Procreate Handbook, accessed March 20, 2025, https://help.procreate.com/procreate/handbook/colors/colors-palettes
2. help.procreate.com, accessed March 20, 2025, https://help.procreate.com/procreate/handbook/colors/colors-palettes#:~:text=Import%20Adobe%C2%AE%20color%20palettes&text=Procreate%20imports%20Adobe%C2%AE%20Swatch,in%20both%20of%20these%20formats.
3. Importing Color Palettes — Procreate for iPad Help Center, accessed March 20, 2025, https://help.procreate.com/articles/zqbpae-importing-color-palettes
4. Color Palettes in Procreate | Importing .aco and .ase Files Tutorial - YouTube, accessed March 20, 2025, https://www.youtube.com/watch?v=jWWjjiCVmMA
5. How To: EASILY MAKE CUSTOM COLOR PALETTES IN PROCREATE - YouTube, accessed March 20, 2025, https://www.youtube.com/watch?v=C4jeC5Gtk34
6. How To Make Color Palette In Procreate - Tutorial For Beginners - YouTube, accessed March 20, 2025, https://www.youtube.com/watch?v=GKZrSgVg17o
7. How To Import And Export Color Palettes In Procreate - Tutorial For Beginners - YouTube, accessed March 20, 2025, https://www.youtube.com/watch?v=f9O1NIfltDg
8. convert ASE to .swatch for Procreate?, accessed March 20, 2025, https://folio.procreate.com/discussions/7/25/23551
9. read and create Procreate .swatches files - GitHub, accessed March 20, 2025, https://github.com/szydlovski/procreate-swatches
10. Generate a Procreate swatches file from a list of colors - GitHub, accessed March 20, 2025, https://github.com/natecavanaugh/procreate-swatch-generator
11. File: README — Documentation for procreate-swatches (0.1.4), accessed March 20, 2025, https://www.rubydoc.info/gems/procreate-swatches
12. laurentzziu/procreate-swatches: A gem to interact with ... - GitHub, accessed March 20, 2025, https://github.com/laurentzziu/procreate-swatches
13. jfairbank/chroma: Ruby gem for color manipulation and palette generation - GitHub, accessed March 20, 2025, https://github.com/jfairbank/chroma
14. joanroig/swatches-to-ase: Convert Procreate Swatches files to ASE format (Adobe Swatch Exchange). Can be used for importing Procreate palettes into Photoshop or other programs like Illustrator or Affinity Designer. - GitHub, accessed March 20, 2025, https://github.com/joanroig/swatches-to-ase

15. accessed December 31, 1969, https://github.com/szydlovski/procreate-swatches/tree/main/src
16. accessed December 31, 1969, https://github.com/laurentzziu/procreate-swatches/tree/main/lib/procreate