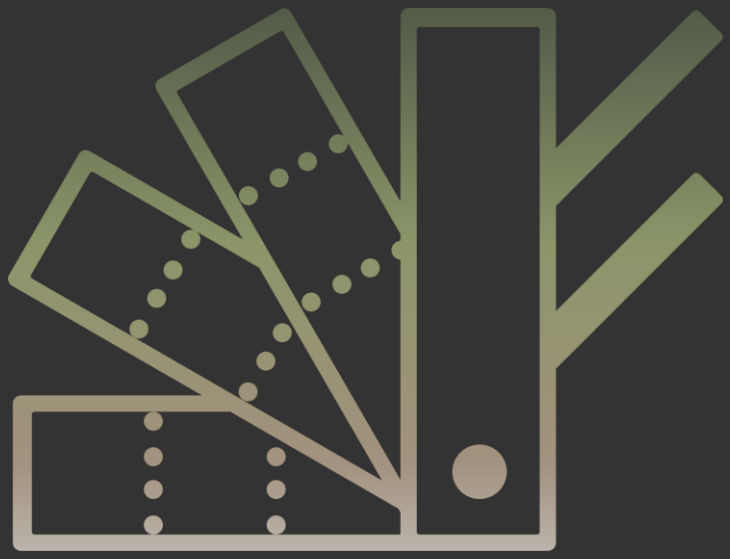


COLOR SCHEME GENERATOR

DOCUMENTATION

VERSION 5.0.0

DOCUMENTATION VERSION 3.0

TABLE OF CONTENTS

PREFACE

On a wonderful day, the 9th of August 2022 to be exact, during a wonderful holiday in Rostock, Germany to clear my mind from that summers extreme stress and all things uni, I decided that taking a break is for losers, only the strong shall survive and how better to practice my tolerance to constant exposure to stress than to start a coding project during my four days of holiday for the whole year?

So I stood there in the ocean, gently being rocked back and forth by the waves, intensely staring at some water insect that had been trying to swim to me for several minutes at this point, when I decided to look back at the beach and noticed the white houses, green grass, blue ocean and golden sand. And I thought “How nice would it be, if I could paint this scene with exactly those colours?”, well knowing, that I hadn't touched paint in ages and that landscape painting was absolutely not my thing.

But that did not matter. My mind had already started racing.

There were a few problems though. The Wi-Fi of the place we were staying had died and so did my high speed, leaving me with no quick way to read up on anything related to creating a colour scheme from an image.

So there I was, armed with only my phone, my tablet I used for drawing, OneNote and my mind going at 100 km/h. I started thinking and writing wild concepts and ideas, until some sort of idea had formed. And because planning is for people who don't know where they are going, I started writing code, since I definitely absolutely knew what I was doing and where I was going.

I remembered my professor's words, that we needed to know how to write code on paper as we wouldn't always have a computer and an IDE available and I cursed my arrogance as I had thought, that if I needed to code, I would use my PC or an online IDE and if I had neither available, I would just not code. So I had to fall back onto my very limited memory of my first programming classes. Lucky for me, due to my bad memory, I write everything down, so I had at least some code snippets I could use as reference for my first lines of code.

I spent my holiday reading up on K-Means Clustering, three-dimensional spaces in Java and different forms of visualizations of colour spaces. And if I am honest, those holidays were amazing. There is a difference between sitting at home and researching and sitting in the sun at the beach in a different city and researching.

When I had come home on the 12th of August, I transferred my code into my IDE and to my surprise and joy, I had only very few issues to fix until the version 1.0 of my Color Scheme Generator was working.

Since then, I incorporated the iTextPDF library to be able to save the resulting colour scheme, I became friends or at least decided to call a truce with gradle to better manage my dependencies and recently even cautiously approached maths to add a colour wheel to the file.

I have I have also learned a ton about Java, gained a lot of experience with managing a project, Code Conventions and most importantly patience and determination and can finally say: I am incredibly proud of my project. It might not be a revolutionary tool that people are in dire need of, but it is my first large project that I have kept up with and improved as I continued my studies and that is the result of my determination, patience with myself and my love for coding, which I discovered a few months into my first semester at the University of Applied Sciences in Wernigerode, Germany.

HOW TO GET **STARTED**

To get a local copy up and running, follow these steps.

PREREQUISITES

This program requires Java 8 to run properly. No action should be necessary as Java 8 is pre-installed on most devices.

INSTALLATION

USING THE COLOR SCHEME GENERATOR

1. Download the latest release file from the 🔗 releases page
2. Open the file by double clicking it

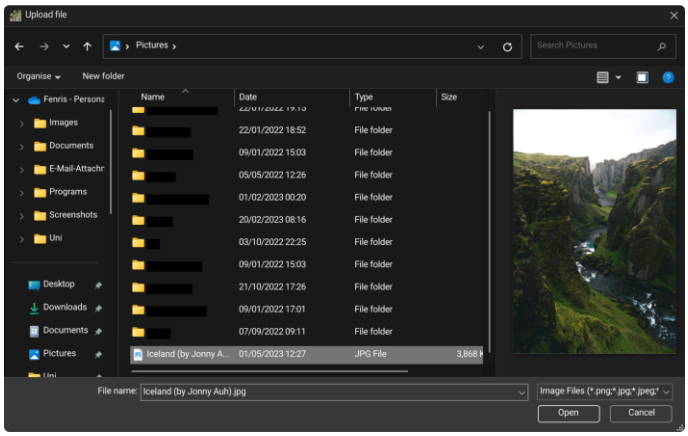
HOW TO USE
THE PROGRAM

This program can be used to generate a colour scheme from an image. The number of colours in the colour scheme can be adjusted. In addition, several more details about the image will be shown in the output file. These details include the average colour, saturation, and brightness of the resulting colour scheme as well as some metadata of the image.

GENERAL USE

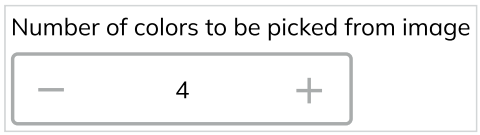
1. Click on the Upload panel.

2. Choose your image and click on “Open”.



3. Select, whether you want the colour scheme to be downloaded automatically (the file will be saved in your Downloads folder with the name “ColorScheme_[Filename].pdf”)

4. Choose the number of colours you want to be extracted.



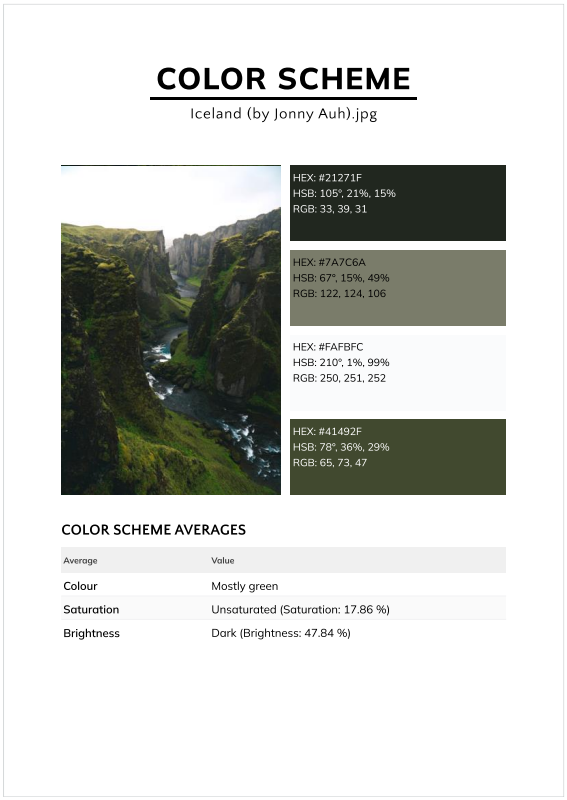
5. Choose, whether you want to add harmonic colours for each extracted colour to the file.

If you decided to add harmonic colours, the dropdown will open where you will be able to select one or more types of colour harmonies. See here for an explanation of the different types.

6. Click on “Start” or click the button with the upload symbol to choose a different image.

7. When the process has finished, the download button will be enabled, and you’ll be able to download your colour scheme.

OUTPUT FILE OVERVIEW



COLOR SCHEME

Iceland (by Jonny Auh).jpg

HEX: #21271F
HSB: 105°, 21%, 15%
RGB: 33, 39, 31

HEX: #7A7C6A
HSB: 67°, 15%, 49%
RGB: 122, 124, 106

HEX: #FAFBFC
HSB: 210°, 1%, 99%
RGB: 250, 251, 252

HEX: #41492F
HSB: 78°, 36%, 29%
RGB: 65, 73, 47

COLOR SCHEME AVERAGES

Average	Value
Colour	Mostly green
Saturation	Unsaturated (Saturation: 17.86 %)
Brightness	Dark (Brightness: 47.84 %)

Page 1

Page 1 shows the file name right below the header. Right next to the selected image (or below, depending on the image size), the chosen number of main colours are listed. Their colour values are specified in hexadecimal format, as an HSB triplet and as an RGB triplet.

Below the colours are the colour scheme averages, listing the average colour (more precisely, the average hue), the average saturation and the average brightness. Note, that this table might be on page 2 if the image is in landscape format.

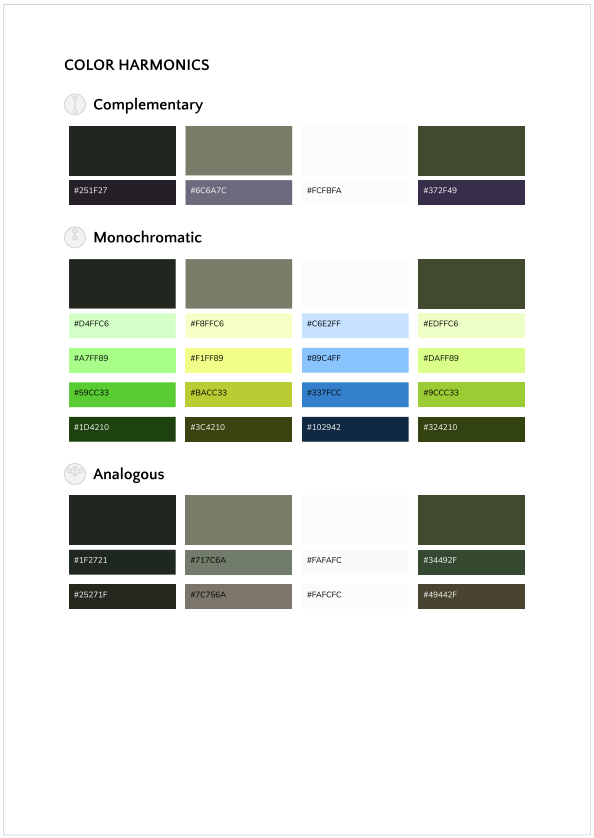
IMAGE META DATA

Meta Data	Value
File name:	Iceland (by Jonny Auh)
File type:	JPG
Size:	3.00 MB
Creation date:	01/05/2023 12:27:56
Last modified date:	01/05/2023 12:27:56
Last accessed date:	13/05/2024 20:54:11
Height:	6000 px
Width:	4000 px
Image type:	8-bit RGB integer pixels in BGR color model without alpha (TYPE_3BYTE_BGR)
Color components:	3
Bit depth:	24
Transparency:	Completely opaque
Alpha:	Transparency not supported by color model
Alpha type:	Non-premultiplied alpha

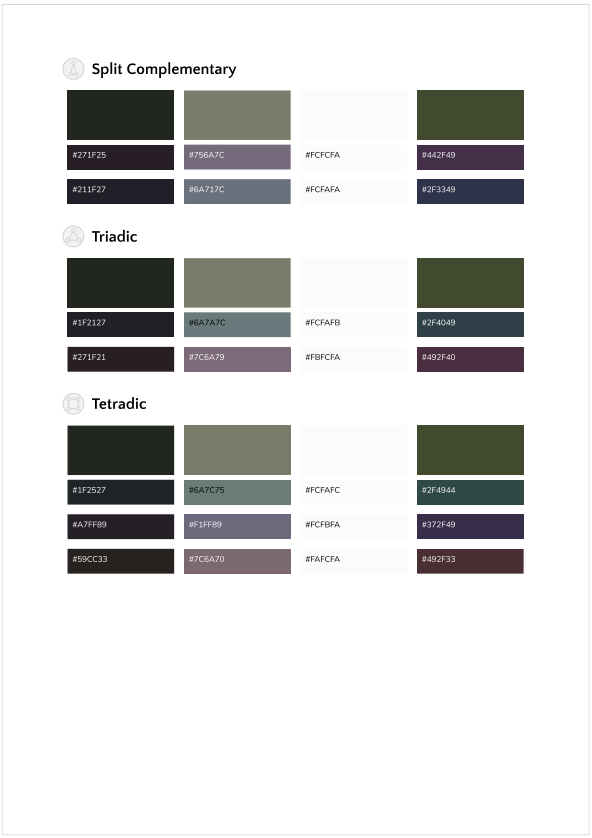
Page 2

Page 2 shows the extracted main colours on a colour wheel. Additionally, the selected images meta data is listed.

Pages 3 and 4 list the selected colour harmonics for the main colours.



Page 3



Page 4

META DATA

1 – **File Name**

The file name and extension of the chosen image as displayed in the file explorer.

2 – **Image**

The image chosen by the user that is used to generate the colour scheme.

3 – **Colour Scheme**

The colour scheme generated from the image along with the colour codes in Hexadecimal, RGB and HSB.

4 – **Colour Wheel**

Shows the main colours of the colour scheme in a colour wheel.

5 – **Averages of colour scheme**

This part shows the average colour, whether the colours are saturated or unsaturated on average along the average saturation and whether the colours are bright or dark on average along the average brightness.

6 – **Metadata of the image**

Shows the metadata of the image, if available.

6.1 – **File name**

The display name of the file, which is also displayed in the file explorer. The file title set in the details tab of the file properties may differ.

6.2 – **File type**

Shows you the file type of your file. This is either a “JPG”/ “JPEG” (no difference) or a “PNG” since the program only accepts files of that type.

6.3 – **File size**

Shows the file size in the appropriate unit, meaning it is within the 1 – 999.99 range of the unit, except for MB, which is in the 0 – 999.99 range.

6.4 – **Creation date**

Shows the date and time of the creation of the file according to the time zone of the user.

6.5 – **Last modified**

Shows the date and time of the last modification of the file according to the time zone of the user. This includes modifications that change the files' metadata, such as editing the name, owner, or permissions.

6.6 – **Last accessed**

Shows the date and time of the last time the file has been accessed, meaning it has been opened. This can mean the file was opened by the user or a program.

6.7 – **Height**

Shows the height of the image in pixels. This value might change according to the rotation of the image.

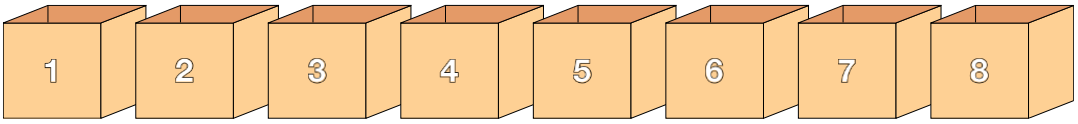
6.8 – **Width**

Shows the width of the image in pixels. This value might change according to the rotation of the image.

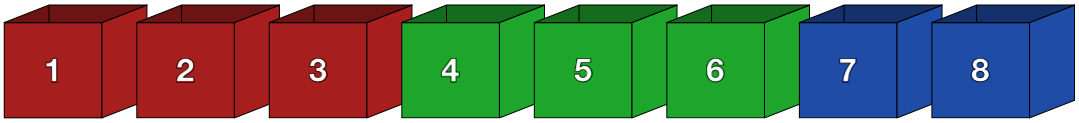
6.9 – **Colour Space**

A colour space is a range of colours on a spectrum that can be displayed visually. There are several ways do display a colour range. See here for some examples.

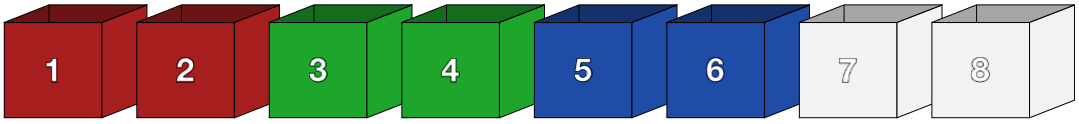
R stands for red, G stands for green and B stands for blue. When the image supports transparency, the A stands for alpha (transparency value).
These colours must be stored in a format readable for a computer. Usually, colour spaces are 8-bit large, which can be imagined as 8 boxes that can hold a single colour value.



3 bits store red, 3 bits store green, and 2 bits store blue, since the human eye is less sensitive to blue light.



If the image supports transparency, each colour only gets two bits, and the alpha value is stored in the 7th and 8th box. Typically, the colour space name will then contain an A somewhere, like in RGBA.



The boxes holding the colours can be moved around to create different colour spaces like the BGR colour space. This only means that the order of the boxes holding the colours is reversed.

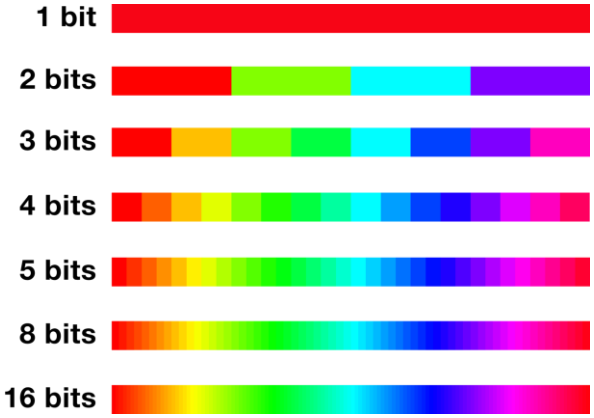
6.10 – **Colour components**

This number shows the amount of colour components of the image. This includes the red, green, and blue, making for three colour components. If the

image supports transparency, the alpha channel is counted as the fourth colour component.

6.11 – **Bit depth**

This shows the number of bits used to indicate the colour of a pixel. The higher the bit depth, the more bits (boxes) are used to store the colour. With more bits, gradual changes in colour can be displayed more accurately. Bit depth is most notable when working with gradients as gradients will start showing clear steps in colour as the bit depth goes down.



6.12 – **Transparency**

Shows, how the image supports transparency. If the image does not support transparency, this will be displayed as “completely opaque”. If transparency is supported, the image is either completely transparent or allows gradual steps in transparency (value between 0 and 1).

6.13 – **Alpha**

Shows, whether the image supports transparency. If the image does not support transparency, this will be displayed as “Transparency not supported by colour model”. If transparency is supported, this will be displayed as “Transparency supported by colour model”.

6.14 – **Alpha type**

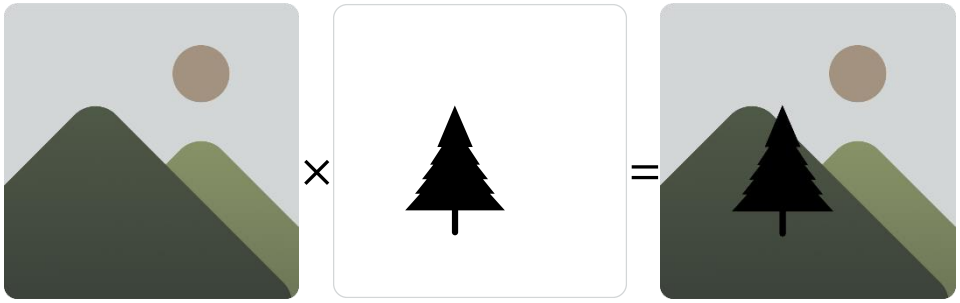
Shows, whether the alpha is premultiplied. This can easily be explained by this quick example, where we want to add a tree to an image of a landscape.

Image 1

Image 2

Now, the background of the image with the tree is not transparent so if we were to add that image onto image 2, it would cover image 2 completely. To fix this, we need to create an alpha mask. The alpha value controls the transparency of an area, so an alpha mask specifies, which areas should remain opaque and which ones should be transparent. Transparent areas are usually shown as black in the mask, while opaque areas are white. These masks are applied to the image by multiplying the mask with the image.

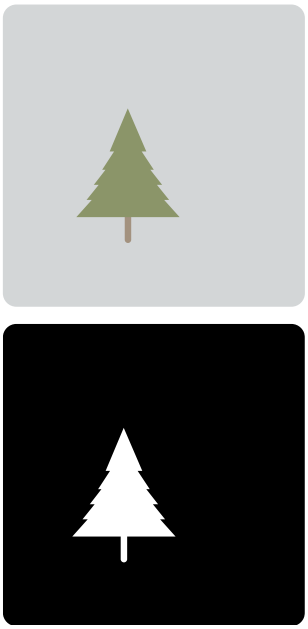
Then, we need to cut out space for the tree on the new background by using the same, but inverted mask. Now everything but the area of the tree is white, meaning only the space, where the tree will be added is transparent.



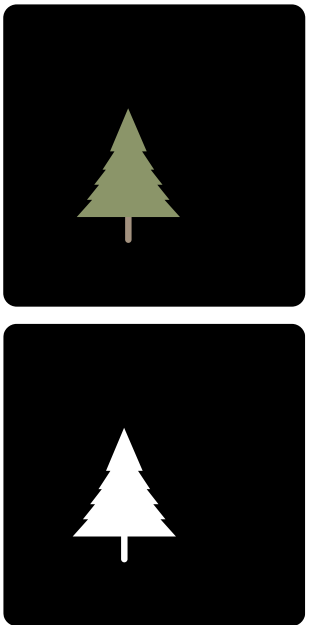
Finally, we can add the masked first image into the masked second image, where it will fill the transparent area.

For the first two steps I used a multiplication symbol to show the mask being applied. This is because masking means multiplied. Therefore, premultiplied is pre-masked. An image with premultiplied alpha already has information about the individual image components. This could look like this:

These components will be added to create the final image in a process called compositing.



Not masked/not pre-multiplied



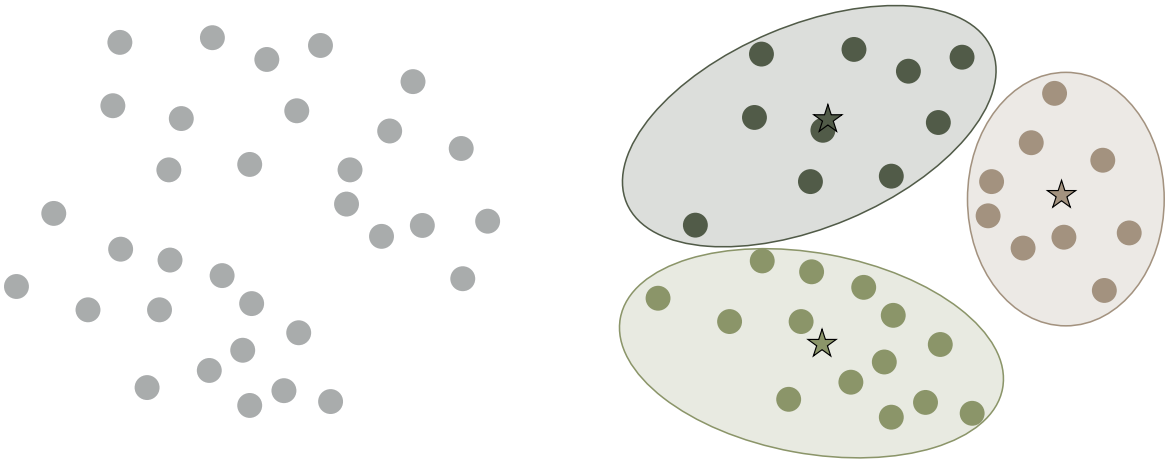
Pre-masked/premultiplied

STRUCTURES

K-MEANS CLUSTERING

K-Means Clustering is a method for grouping several values into a set number of groups. The number of values must be smaller or equal to the number of groups. Each value belongs to the group with the mean value that's closest to the value. This mean is called a centroid.

K-Means aims to minimize the difference of the values within the groups to the mean value of the group to end up with groups of values that are like each other.



Before K-Means

After K-Means

The algorithm repeats until the minimum variance has been reached and works as follows:

1. Define a number of groups.
2. Select random values as the initial centroids.
3. Each observation is assigned to the cluster with the nearest mean value, in this case, this is determined by calculating the Euclidean distance. This is a method to

determine the distance between two points in a two- or three-dimensional space. An observation can only be assigned to one cluster at once.

1. Now, the centroids will be recalculated, as this value will have changed after the assignment of the observations. The new centroid is the mean of all observations that are assigned to the cluster.
2. Repeat step 3 and 4, until the assignments no longer change, or the minimum variance has been reached.

K-MEANS++

K-Means++ is an algorithm that is used to determine the initial centroids. This algorithm is used to prevent the initial centroids from being chosen poorly, meaning they might be close to each other, which can result in a bad clustering. The algorithm works as follows:

1. Choose one centroid at random from the values.
2. For each value that is not a centroid, calculate the distance between the point and the nearest centroid.
3. The value that's furthest away from the centroids is picked as the next centroid. This is determined by calculating the fitness of each value using a method called Roulette Wheel Selection.
4. Repeat Steps 2 and 3 until the chosen number of centroids has been selected.
5. Now that the initial centroids have been chosen, proceed using standard K-Means.

EUCLIDEAN DISTANCE

The Euclidean distance describes, how long the distance between two points is. For one-dimensional calculations, this can be calculated using the formula $d(p,q) = |p - q|$ where p_1 and p_2 are two points, $d(p,q)$ is the distance between those points and $|p - q|$ describes the absolute value of $p - q$, meaning, that if the result should be negative, the

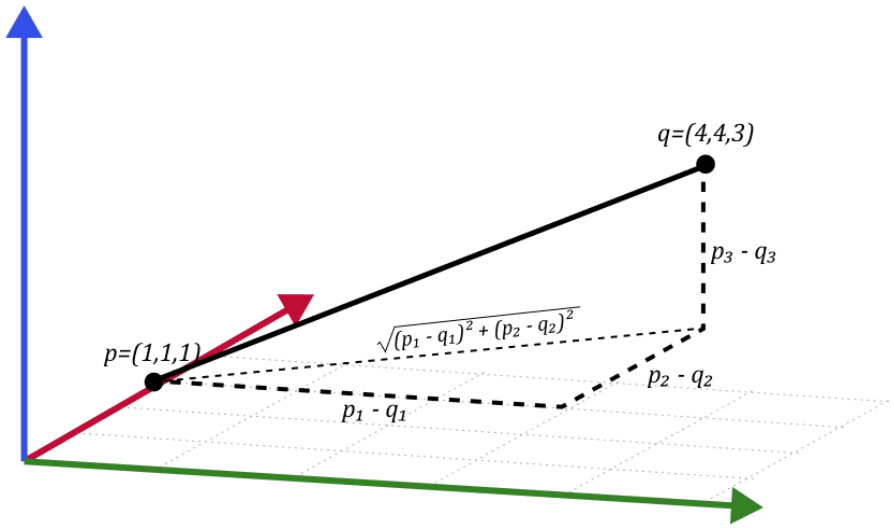
leading minus will be omitted. This is the same as $d(p,q) = \sqrt{(\mathrm{p}-\mathrm{q})^2}$, as squaring and then taking the square root will leave a positive result.

In this case, we need three dimensions, as we will be moving within a three-dimensional colour space. This just means, that we will have an x-, y- and z-Axis that represent the R(ed), G(reen) and B(lue) value of a pixel. To calculate the distance between two three-dimensional points p and q,

$$p = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}, q = \begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix}$$

the following formula will be used:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}$$

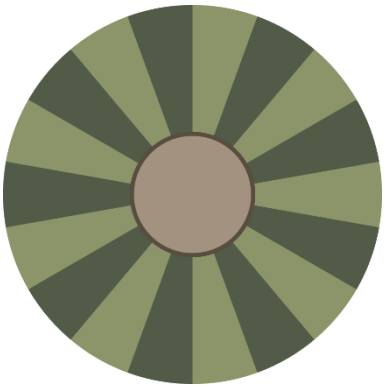


FITNESS

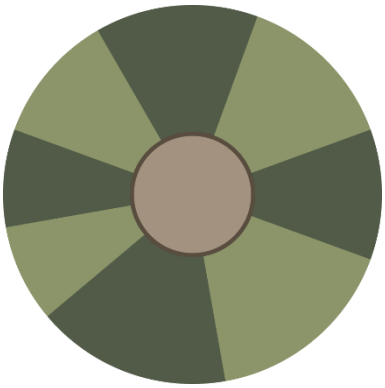
The fitness of a value is a value that determines how likely it is to be chosen. The higher the fitness, the more likely it is to be selected. The fitness is calculated using a method called Roulette Wheel Selection. It is inspired by a roulette wheel but has some important differences.

A regular roulette wheel has slots of the same size, meaning each slot has the same chance of being picked.

In Roulette Wheel Selection, the size of the slots is determined by the fitness of the values. The higher the fitness, the bigger the slot. This means that values with a higher fitness are more likely to be selected.



Regular roulette wheel

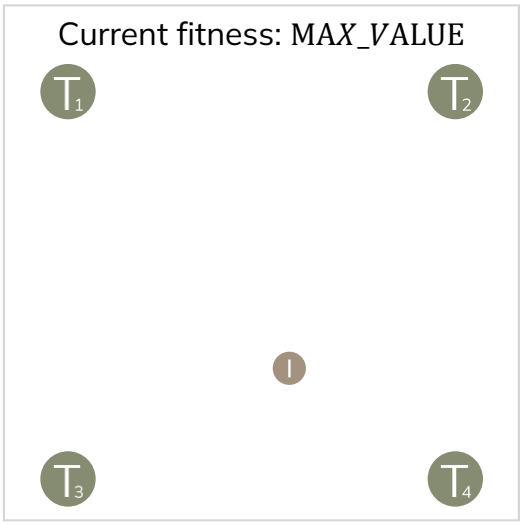


Roulette wheel selection wheel

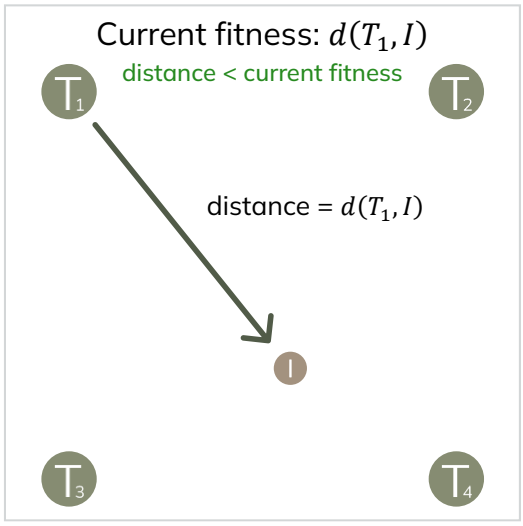
This fitness will be calculated by setting the fitness of the currently inspected value to the highest number possible. If there are no other values, this value will be picked as there are no other values to pick from.

Should there be other values, the distance between the currently inspected value and the centroids will be calculated and set as the values' fitness. If another centroid is closer to the value, the fitness will be set to the distance between the value and this centroid. This ensures, that a values' fitness depends on the distance to all centroids.

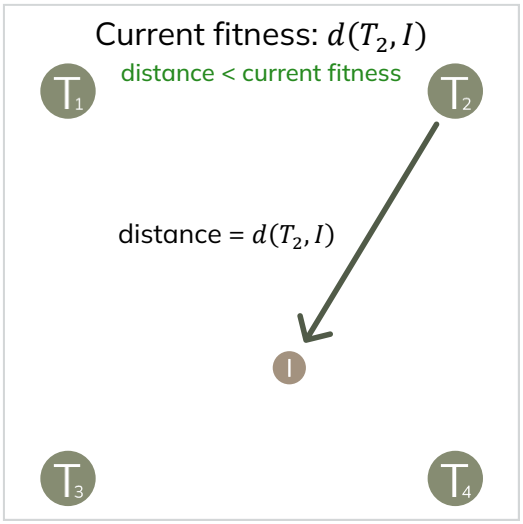
To determine the new centroid, the fitness of each value, that is not a centroid will be summed up. Then, a random number between 0 and this sum will be chosen. This number is now the threshold. Now, all values will be summed up again, one after another. If adding a values' fitness to the sum would surpass the threshold, this centroid will be selected as the new centroid.



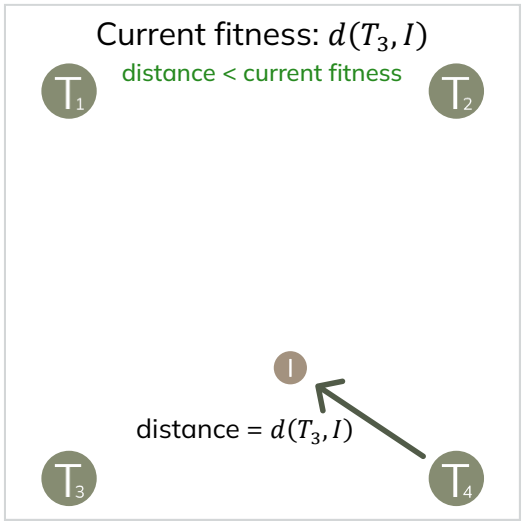
1. Start of the clustering process



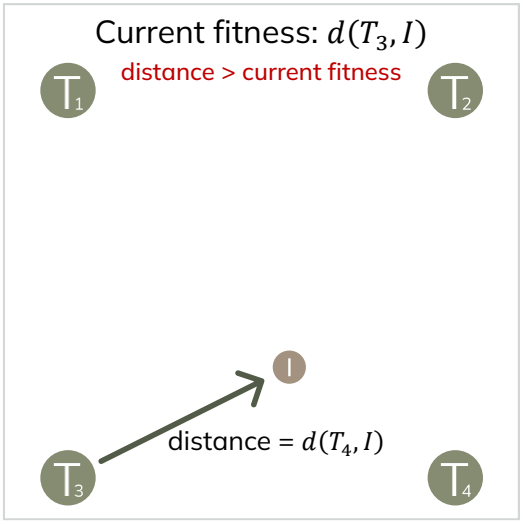
2. Calculate distance d to centroid 1 and if d < fitness, set fitness to d



3. Repeat for centroid 2



4. Repeat for centroid 3



5. Repeat for centroid 4

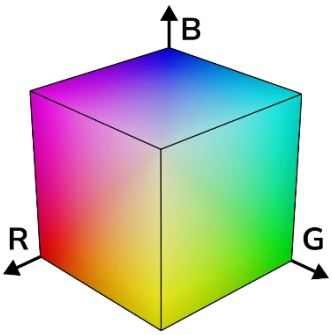
Fitness of value:
$d = (T_3, I)$

3D COLOUR SPACES

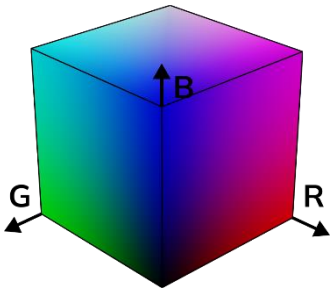
A colour space is a conceptual tool to help with understanding the colour capacities of a file or a screen device. The three-dimensional visualisation of such a colour space contains all possible colour combinations, meaning that each colour has a unique position in the colour space. Colour spaces are often represented by using two-dimensional slices from the three-dimensional shape. Since the Color Scheme Generator mainly uses the RGB and the HSL colour model, I will only go into detail for these two.

RGB

This colour model uses the three primary colours red, green, and blue (thus RGB) to create up to 16.8 million different colours. To represent a colour using the RGB model, the colours proportion of red, green, and blue must be set, which can each have a value between 0 and 255.



The front view of the 3D-RGB colour model



The back view of the 3D-RGB colour model

Examples for colours using the RGB model:

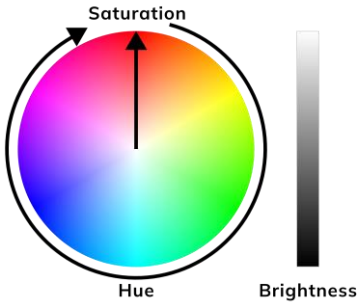
Name	Colour	Red	Green	Blue
Red		255	0	0
Green		0	255	0
Blue		0	0	255
Black		0	0	0
White		255	255	255
Yellow		255	255	0
Cyan		0	255	255
Magenta		255	0	255

HSB

A colour consists of three parts: The hue, the saturation, and the brightness.

Hue		The hue can have a value between 0° and 360°, as this value determines the location on the colour wheel.
Saturation		The saturation can have a value between 0 and 100.
Brightness		The brightness can have a value between 0 and 100 as well.

Examples for colours using the HSB model:



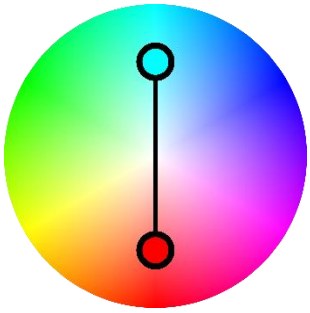
Name	Colour	Hue	Saturation	Brightness
Red		0°	100	100
Green		120°	100	100
Blue		240°	100	100
Black		Any	Any	0
White		Any	Any	100
Lavender		278°	41	91
Tomato		12°	72	94
Emerald		148°	52	86

COLOUR HARMONIES

Colour harmonies refer to aesthetically pleasing combinations of colours. These harmonies may consist of several different colour types.

COMPLEMENTARY

Complementary colours exist opposite each other on the colour wheel. Their saturation and brightness value remain unchanged.

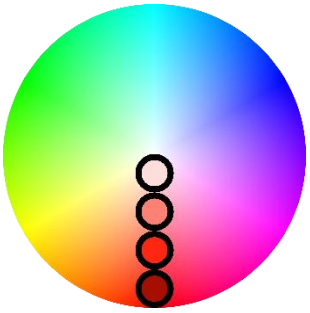


Colour 1

Colour 2

MONOCHROMATIC

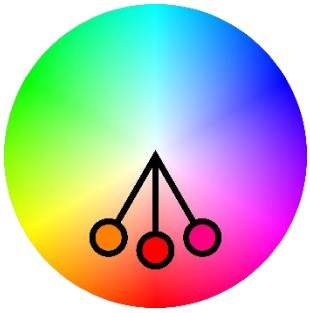
Monochromatic colours all have the same hue but different saturation and brightness values.



Colour 1 Colour 2 Colour 3 Colour 4

ANALOGOUS

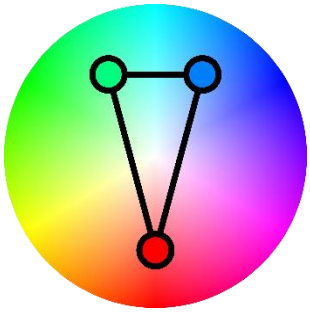
Analogous colour harmonies consist of colours, that are next to each other on the colour wheel.



Colour 1 Colour 2 Colour 3

SPLIT COMPLEMENTARY

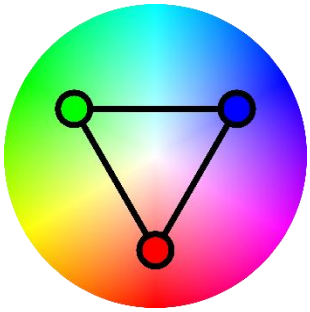
Split complementary colour harmonies are a mix between complementary and analogous colour harmonies. The two harmonic colours are the two colours next to the main colours complementary colour.



Colour 1 Colour 2 Colour 3

TRIADIC

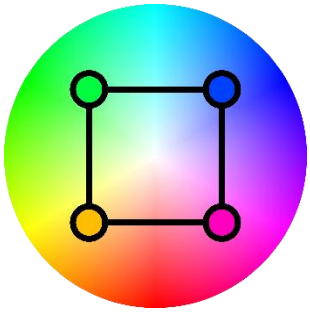
A triadic colour harmony uses three hues, that are spaced evenly on the colour wheel. The saturation and the brightness remain unchanged.



Colour 1 Colour 2 Colour 3

TETRADIC

A tetradic colour harmony uses four hues, that are spaced evenly on the colour wheel. The saturation and the brightness remain unchanged.



Colour 1 Colour 2 Colour 3 Colour 4

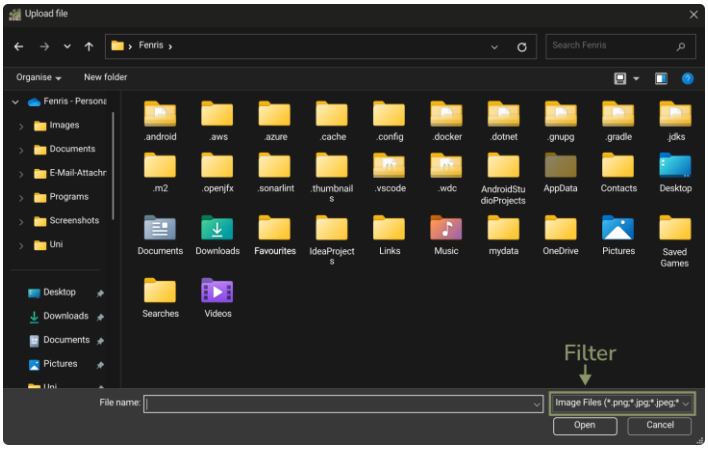
CODE

STATISTICS

Packages: 5 Files: 14 Methods: 144 Lines: 3978

OVERVIEW

If the user clicks on the upload panel, a file chooser is opened at the home location of the file explorer. Additionally, a filter is applied, showing only images and directories. The user interface will also be reset in case the user has uploaded a different image before.

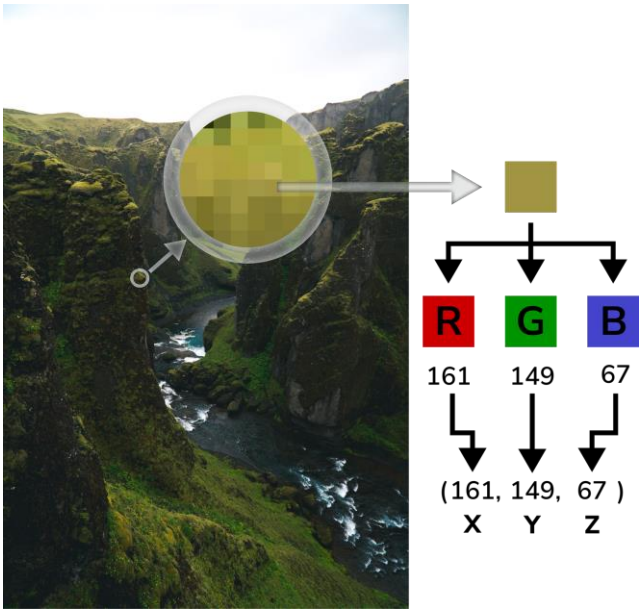


Now, the user can select an image to extract the colour scheme from. Unless the user cancels the choosing process or closes the file chooser, the path to the selected file is saved. The upload panel will now be replaced by the Start Button and a button to select a different image. Additionally, the name of the selected file will be displayed. The user can now either choose to keep the default settings or adjust them to their liking. They can choose to have their file downloaded, as soon as the process is finished, adjust the number of extracted colours to a number between one and six or add different colour harmonics for each extracted colour. Per default, the file will not be downloaded automatically, four colours will be extracted, and the file will not contain any harmonic colours. If the user clicks on the Start Button, the process to extract the colour scheme is started. If the image is larger than 150 pixels in either width or height, it is resized to measure 150 pixels on its longest side while retaining its aspect ratio. This is done, to reduce the

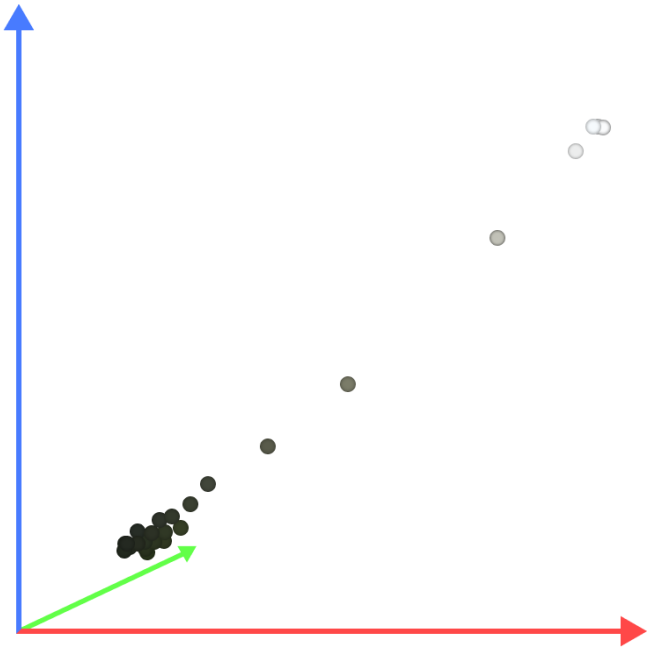
workload and required time for the program to finish, as each pixel will need to be inspected and this amount will grow exponentially with increasing image size.

Now that the image has been found, it will be analysed pixel by pixel.

Each pixel holds the colour information for that specific point in the image. This colour data can be extracted as an RGB value and will be saved as a three-dimensional point in a list. To convert the colour into a three-dimensional point, the red, green, and blue values will be saved as this points x, y, and z value respectively.



When every pixel has been analysed and saved to a list containing all pixels as three-dimensional points, the process to determine the main colours starts. The number of colours to be generated that can be set by the user will be used to initialize the clustering process by utilizing KMeans++.



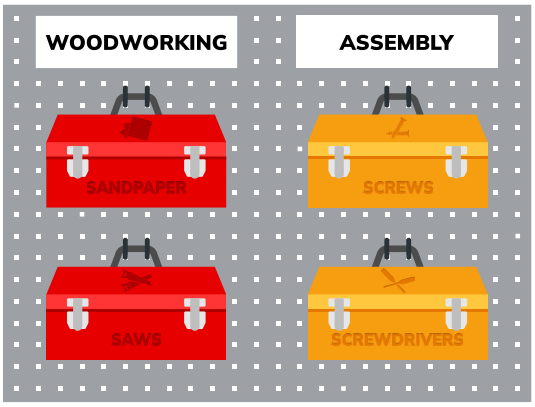
Example of 30 RGB-Points in a coordinate system

If this process is successful, the actual KMeans clustering starts. All the pixels that were saved into the list, will be added to a three-dimensional space, ranging from 0 to 255 on each axis. This range is always the same, as black, the darkest colour possible, is RGB values of (0, 0, 0) and the brightest colour, white, has an RGB value of (255, 255, 255). Therefore, any values outside this range do not exist.

When all pixels have been divided into the number of clusters, specified as the number of colours to be extracted, the coordinates of the centre points of those clusters represent the main colours of the chosen image. These colours will then be added to the output file, together with the dominant colour hue, average saturation, and average brightness of the image.

The output file is created from top to bottom. At first, the title and the file name are added. Below, the image will be displayed. If the image is in landscape format or exceeds a certain width, the extracted colours will be displayed below the image. Otherwise, they will be added to the right. Then, the colour scheme averages, consisting of the average colour, saturation and brightness are displayed in a table below. To visualise the extracted colours, a colour wheel is added, displaying where on this colour wheel the extracted colours are located. This colour wheel disregards the colours brightness and only places them using their colour and saturation. Below this, the images meta data is added. If the user decided to add colour harmonies, they will be added to the following pages.

INTRODUCTION TO CODE



Imagine a workshop with several areas, where each area has several toolboxes with unique tools. One area might be for woodworking, so one toolbox might contain several different types of sandpaper and another one might contain different saws. Another area might be used for assembling different parts, so one toolbox might contain different screws and another one has a variety of screwdrivers. But all those areas are required to build a chair.

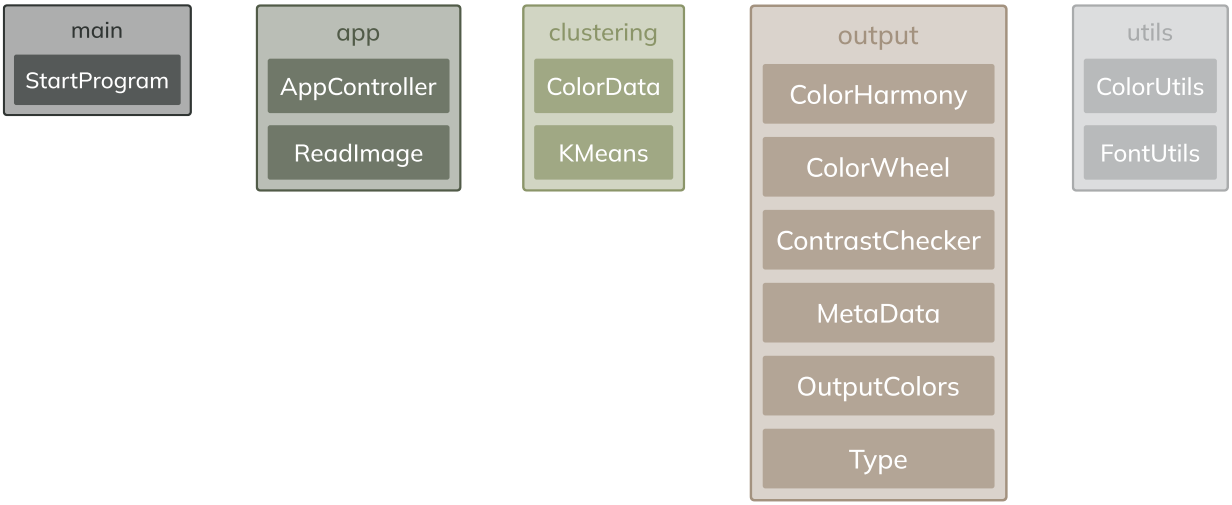
Now, how does this relate to code? Well, the structure of code is quite similar. Code can be separated into so called “packages”. Each package contains code for a different part of the program, for example the output. This code is divided into classes. Each class contains the tools required for a specific step or a functionality of this part. For example, the output-package has the class ColourHarmony, where the harmonic colours are

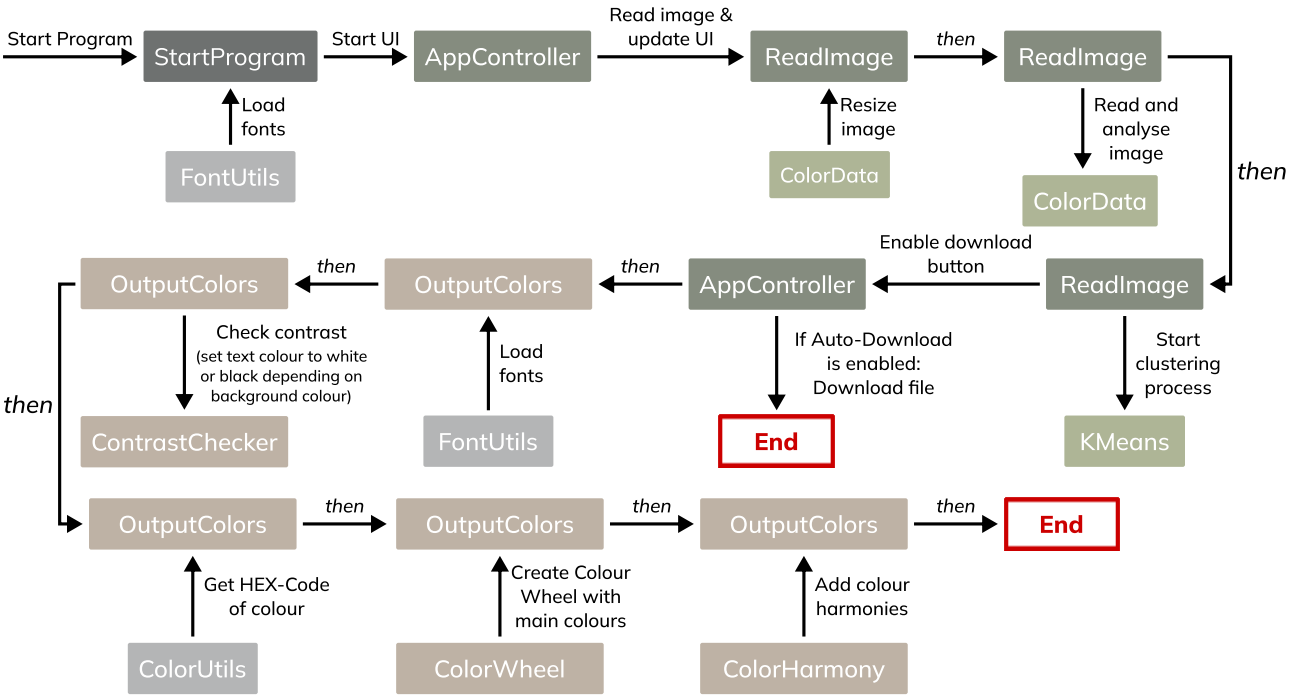
determined. Those classes contain methods, which are simply tools to do certain tasks. This is where the magic happens. While packages and classes simply separate and organise code, methods contain the code that will be used for the program.

The following chapter will go into further details. You can skip this part, if you are not interested in code or a more detailed description of how the program works.

HOW THE PROGRAM WORKS

We start out with our 5 packages and their classes.





CLUSTERING

Files: 2 Methods: 26 Lines: 797

COLORDATA.JAVA

Type: Class Methods: 23 Lines: 618

This class is responsible for the most important part of the program: reading the image and determining the main colours. Here, the image will be read pixel by pixel. Each pixel is stored to later be used to calculate the colour scheme. Additionally, the class contains an inner storage class called Pixels, that stores information about an image's pixel, such as the colour and its cluster number. The class also contains a custom exception that will be thrown, when the number of recorded pixels differs from the number of pixels that are calculated by multiplying the images width with its height.

KMEANS.JAVA

Type: Class Methods: 3 Lines: 179

This class provides the methods for the K-Means clustering process that is used to determine the main colours.

INTERFACE

Files: 2 Methods: 27 Lines: 770

APPCONTROLLER.JAVA

Type: Class Methods: 24 Lines: 633

This class controls the user interface and computes the user input. Here, the values required for the program will be collected and handed over to the part of the program, where the main colours will be calculated.

READIMAGE.JAVA

Type: Class Methods: 3 Lines: 137

This class structures the process of the colour scheme generation. As the different parts of the program are doing their work, this class sends updates to the controller, to keep the user updated about the status of the process.

MAIN

Files: 2 Methods: 3 Lines: 81

LAUNCHER.JAVA

Type: Class Methods: 1 Lines: 8

Launches StartProgram.java and is required to execute the program.

STARTPROGRAM.JAVA

Type: Class Methods: 3 Lines: 73

Starts the application and opens the user interface.

OUTPUT

Files: 6 Methods: 67 Lines: 2029

COLORHARMONY.JAVA

Type: Enum Methods: 0 Lines: 18

This enum simply contains the available colour harmonies that can be chosen. The available colour harmonies are complementary, split complementary, analogous, triadic, tetradic and monochromatic.

COLORWHEEL.JAVA

Type: Class Methods: 14 Lines: 366

This class creates the colour wheel displaying the calculated main colours and provides the methods required for calculating the colour harmonies.

CONTRASTCHECKER.JAVA

Type: Class Methods: 7 Lines: 112

This class is used to determine the contrast between two colours. It provides methods to determine, whether to use black or white text on a certain background colour to ensure legibility.

METADATA.JAVA

Type: Interface Methods: 5 Lines: 68

This interface represents the meta data of an image file and allows values to be assigned to the meta data types that will be displayed in the output file.

METATYPE.JAVA

Type: Enum Methods: 3 Lines: 123

This class contains the types of meta data that will be listed in the output file. As the program and the output is available in both English and German, the titles are not set but rather are pulled from the file containing the text snippets in the language chosen by the user.

OUTPUTCOLORS.JAVA

Type: Class Methods: 38 Lines: 1342

This class creates the output file. It takes the calculated colours and the user input and uses them to display the image, colour scheme, colour scheme averages, meta data and optionally the colour harmonies.

UTILS

Files: 3 Methods: 21 Lines: 301

COLORUTILS.JAVA

Type: Class Methods: 3 Lines: 52

This utility class provides colour related utility methods that provide functions such as converting a hexadecimal colour to an RGB colour and vice versa.

FONTUTILS.JAVA

Type: Class Methods: 10 Lines: 126

This utility class provides font related utility methods that for example return a font of a certain type, with a specified weight or colour. Additionally, the class contains an enum that provides the possible font weights.

PATHUTILS.JAVA

Type: Class Methods: 8 Lines: 123

This utility class provides file paths to required assets. Since the way file paths are evaluated, changes depending on whether the program is executed from a JAR file or from an IDE, the appropriate file paths are set when starting the program.

DEFINITIONS

ONENOTE

Microsoft OneNote is a software for organized notetaking, designed for free-form information gathering and multi-user collaboration. It can be used for gathering notes, drawings, screen clippings, and audio commentaries, and synchronizes between devices, provided, the devices have a connection to the internet.

IDE

An IDE (short for “integrated development environment”) is a software used for programming and software development. It usually consists of a code editor, tools for compiling code into binary, packaging binary, and running automated tests (building) and debugging (error finding and fixing) tools.

K-MEANS CLUSTERING

An algorithm, that aims to partition several values into a set number of groups. Each group has a mean value, called a centroid, and the algorithm ends, when each value is assigned to the group with the mean value, that is closest to its own value.

ITEXTPDF

iTextPDF is a library for creating and manipulating PDF files, originally released in 2000. Until version 5.0.0, the source code was licensed as open source, meaning everybody may contribute to expanding the library.

GRADLE

Gradle is a tool, that automatizes the building process of a program and controls the compilation, testing, deployment, and publishing of code. This means, that rather than downloading an entire library, which can get quite extensive, a link can be added to a special file and gradle will download the library and all libraries this library might depend on automatically. Should the user not need that library any more or require a newer version, the link can easily be changed, and all unused remnants of the library will be deleted.

CODE CONVENTIONS

Coding conventions are guidelines for a programming language that recommend programming style, practices, and methods for each aspect of a program written in that language. These conventions span from file organization, indentation, comments, naming conventions and programming practices to the general structure of the code. Every language has their own code conventions.

OS

OS is the abbreviation for “operating system” and describes the system software, that a computer is running on. The most common operating systems are Windows, macOS (Apple) and Linux.