Osman Warsi September 12, 2020

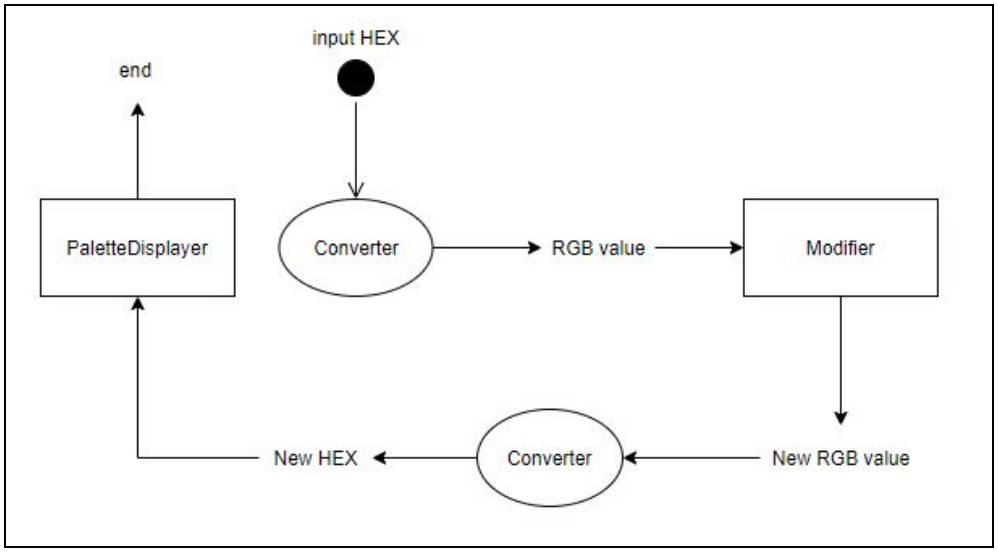
Wisual
https://wisual.netlify.app

Wisual generates different color schemes for designers/users based on their color input. It uses color theory principles and generates the best possible color palettes, provides with UI templates, code, and the ability to copy the hex values.

Color type input from HTML5 is only available on desktops and certain android browsers, but for mobile use, the input must be in hex format (#rrggbb).

General Overview

- User input in hex format needs to be converted to RGB since dealing with RGB values is significantly easier for me to manipulate.
- A converter is needed to change the hex value to the corresponding RGB value.
- The RGB value is modified to generate the 3 primary colors (main, complementary, tertiary) and 2 secondary colors (modifiedUp, modifiedDown).
- The new RGB value is now processed in the Displayer which uses it to generate the final palettes.



Converter

- HextoRGBvalues(hex_input):
  - The hex input characters are stored into separate r, g, and b variables.
  - After converting them to decimal and multiplying by 16, the total value of each r, g, and b is obtained in decimal form.
  - The final value is concatenated in a form of a string and returned.
- rgbTohex(r, g, b):
  - Using the individual decimal values, convert it to hex and concatenate with '#'
- convertHextoRed(hex):
  - This is used to only return the total R-value in decimal form.

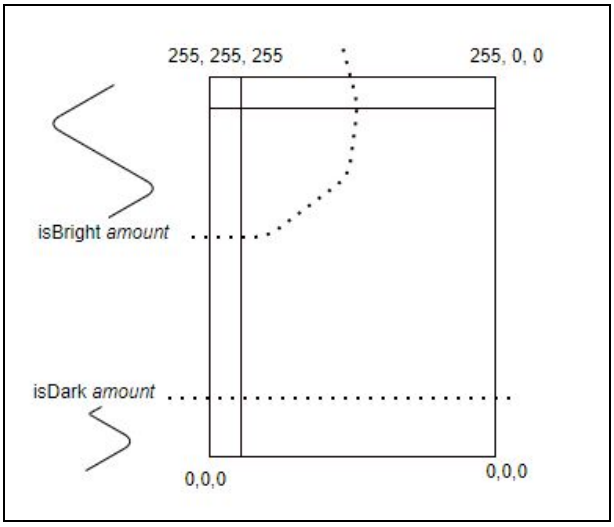
- convertHextoGreen(hex):
    - This is used to only return the total G value in decimal form.
- convertHextoBlue(hex):
    - This is used to only return the total B value in decimal form.

These separate values are needed to compare the user input and modify each value separately as needed.

- checkHighestVal(hex):
    - Store each R, G, and B values using convertHextoRed/Blue/Green functions above.
    - Compare all and return the max value

Color Testers

- **isDark (hex, amount)**
    - Using the converters, check the individual values of R, G, and B
    - Check for each value is it is *less* than the amount
    - If they are, the color is considered dark
- **isBright(hex, amount)**
    - Using the converters, check the individual values of R, G, and B
    - Check for each value is it is *greater* than the amount
    - If they are, the color is considered bright



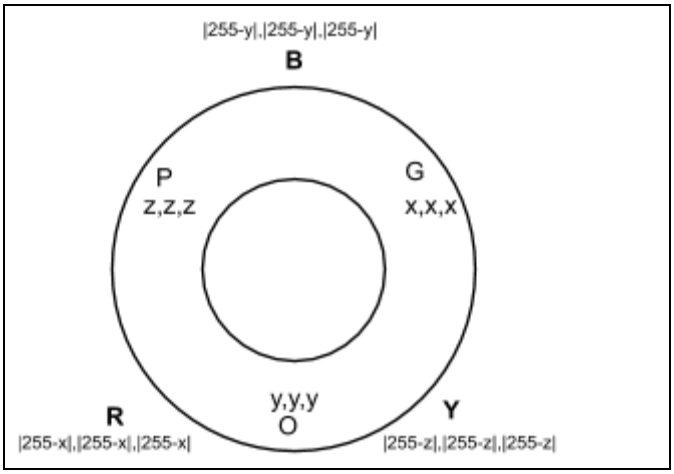
Using an RGB 3-vector space here is a description of how I come up with the amount values. The current values I find good for isDark and isBright are 60 and 200 respectively.

Modifiers

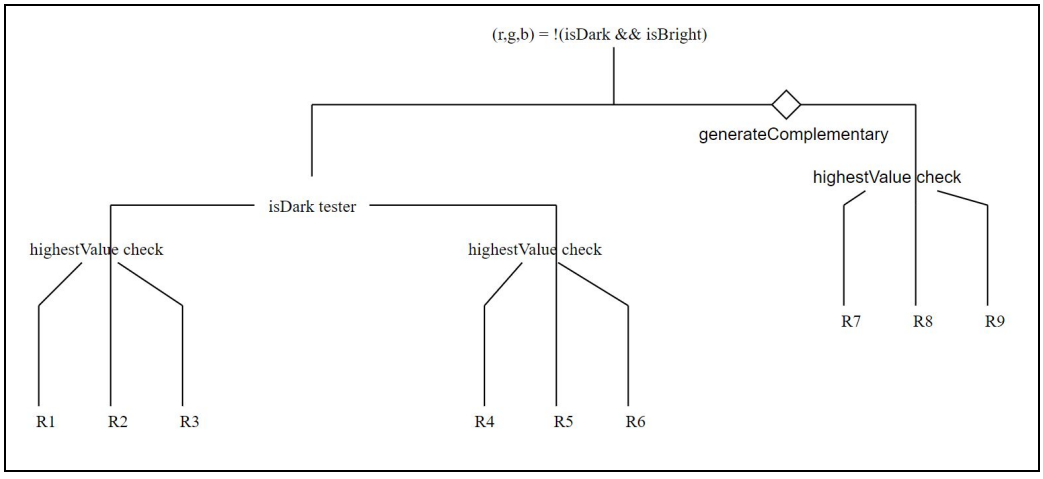
- hexAddModifier(hex, amountR, amountG, amountB)
    - Converts hex to RGB and adds specific amounts to RGB values and returns the new RGB
- hexSubtractModifier(hex, amountR, amountG, amountB)

- Converts hex to RGB and subtracts specific amounts to RGB values and returns the new RGB

➢ generateComplementaryColor(hex)
- Extracts RGB values and returns the modulus subtracted by 255 for each RGB
- Returns the new RGB value



➢ generateTertiaryColor(hex)
- Checks if the hex isDark or isBright or neither
- Based on the check, use second check: checkHighestVal
- Based on second check, use hexModifiers suitably



Decision Tree for tertiary generator

Displayers

➢ DisplayColor
- Component which styles a block using a single prop value

    - Extract prop's R, G, and B value to generate a color bar
    - Returns a table with image block and color bar
- ➢ secondaryColor
    - Uses the modifiers to generate secondary colors using the prop value
- ➢ UIGenerator
    - Component accepting 3 prop values
    - Based on the values, styles the corresponding css sections
    - Returns the canvas concept by embedding seperate sections
- ➢ ColorGenerator
    - Using the specific modifiers and components, generate a single palette of colours
    - Uses the parameters to inject them to the default CSS code

Aesthetic Color choices

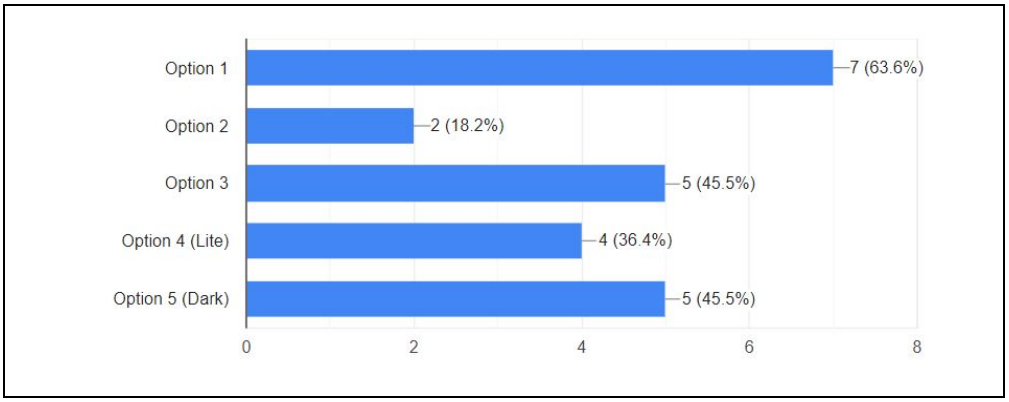
- ➢ Complementary Color
    - Basic version: apply generateComplementary without any modifications
    - Some Darker colors and colors with a high R, B, and low G value do not blend well together
    - Updated version: Due to this, using converters measure to test if color is within these parameters and modify the generated complementary colors with more modifiers to obtain a better color
- ➢ Tertiary Color
    - After application of 3 levels of checks, return the color by the displayer
    - Updated Version: Some tertiary colors do not blend well if they have low G and isDark(hex, ~60) value. To counter this, use appropriate modifiers for further modification
- ➢ Secondary color
    - Generate analogous colors based on the main color value
    - Using modifiers, generate an analogous palette with some tertiary components in RGB

Empirical observation

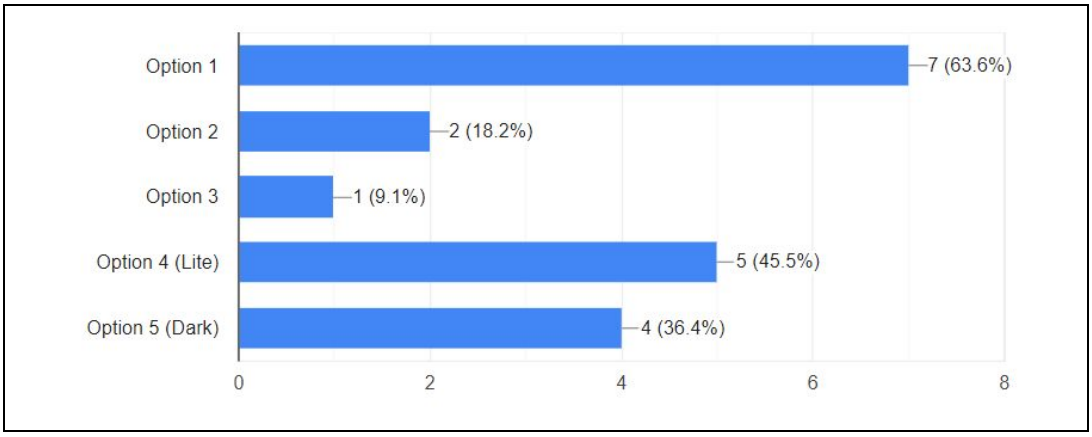
Testing out the different generated palette with both Basic and Updated version was done via forms. The answers are compiled from designers, front-end engineers, and people knowledgable about color theory.

Link to google form: https://forms.gle/UdCCBaywtjjsXLfm8

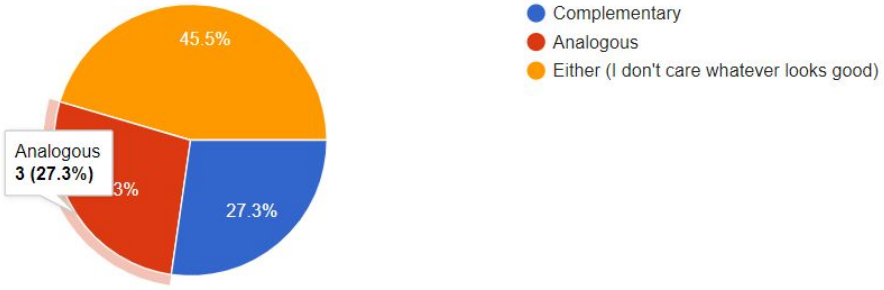
- ➢ Test color: #5495b6



- Test color: #e84711



- Secondary Color: (Analogous, Tertiary, either)



V2.0 Color Changes:

- Tertiary color, Dark and Lite Theme palette generator
