Lab 13: Color Palette Generator DevLog

Pluto Zitek – https://github.com/GemedetAdept (https://github.com/GemedetAdept)

Goals

- ✓ Implement remaining conversion methods
  - ✓ HEX
    - ✓ HEX to HSL
    - ✓ HEX to HSV
    - ✓ HEX to RGB
  - ✓ HSL
    - ✓ HSL to HEX
  - ✓ HSV
    - ✓ HEX
  - ✓ RGB
    - ✓ RGB to HEX
- ❑ Implement ValueDeviation class for controlled randomization of output colors
  - ✓ Implement custom, adjustable logistic function
    - ✓ Method to catch invalid floor and ceiling values
  - ❑ Implement ValueDeviation methods
    - ❑ Hue, HSV
    - ❑ Hue, HSL
    - ❑ (R,G,B), RGB
    - ❑ (R,G,B), HEX
- ❑ Create a menu(s) for each step requiring user input
  - ❑ Main menu
  - ❑ Color code input
  - ❑ Color harmony selection
- ❑ Link all methods together

Implementation and Progress

Challenges with Color Space Geometry

Because RGB and HEX both contain three values, each with the same range $(R, G, B) \in \mathbb{R}[0,1]$ and

$(H_R, H_G, H_B) \in \mathbb{R}[0_{10},1_{10}]$, which are all congruent, they can be geometrically visualized as *cubes* (Fig. 1).

HSV and HSL, albeit also containing three values, contain two distinct ranges and value types. The hue (H_V, H_L) has a different range and is measured in degrees: $H_V, H_L \in \mathbb{R}[0°,360°]$. This is distinct from the shared range and value type of saturation (S_V, S_L), value (V), and lightness (L): $S_V, S_L \in \mathbb{R}[0,1]$, $V \in \mathbb{R}[0,1]$, $L \in \mathbb{R}[0,1]$. This combination of dislike ranges and types means that both HSV and HSL are visualized as *cylinders* (Fig. 2).

While H_V and H_L are equal, S_V and S_L, & V and L are *not*. This requires four more, unique deviation methods, which will have to be individually tuned. As such, I've opted to limit the current scope and focus only on the deviation methods for the hues of HSV and HSL, and for all values of RGB and HEX.

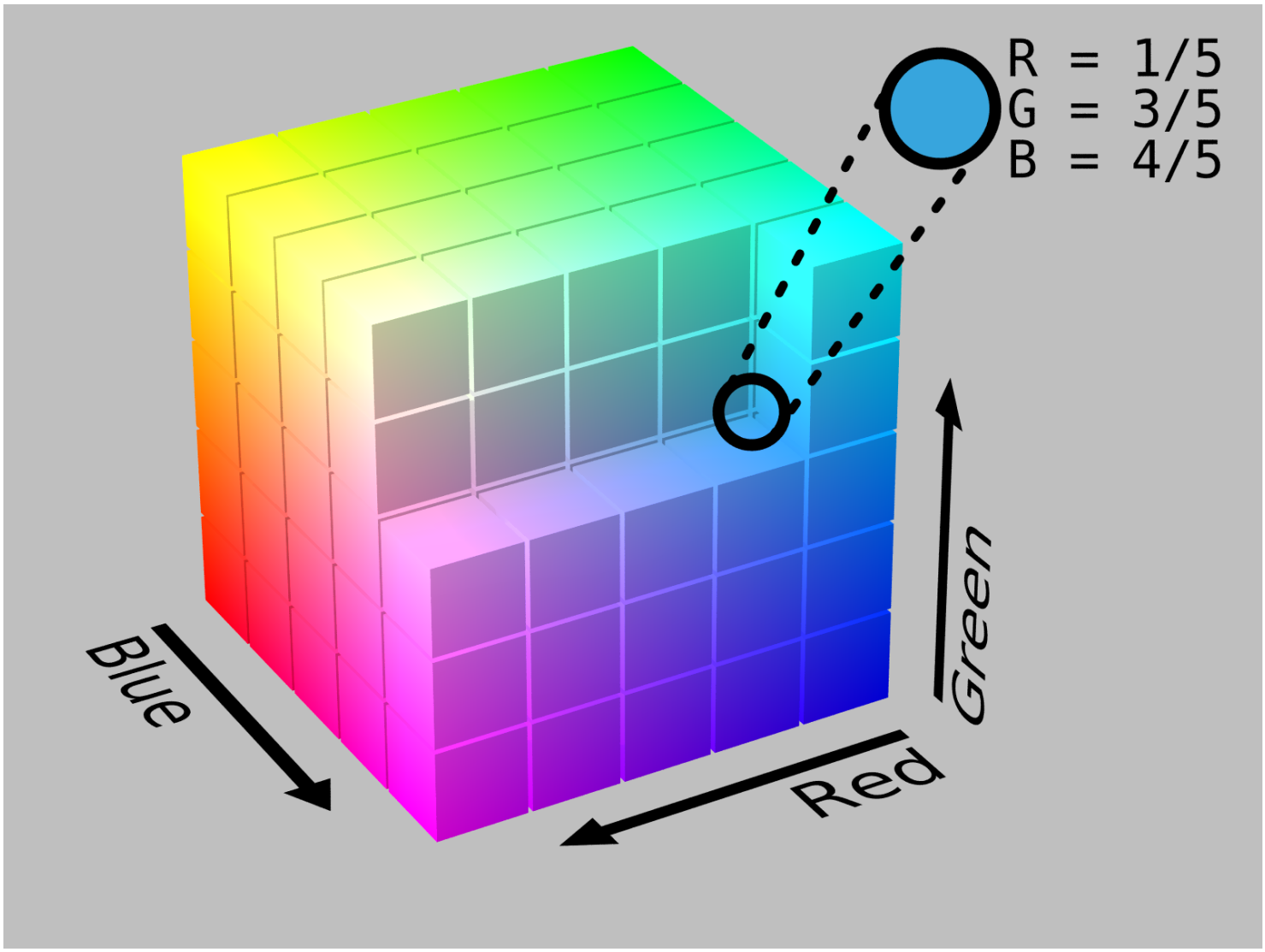

Figure 1. RGB color space cube

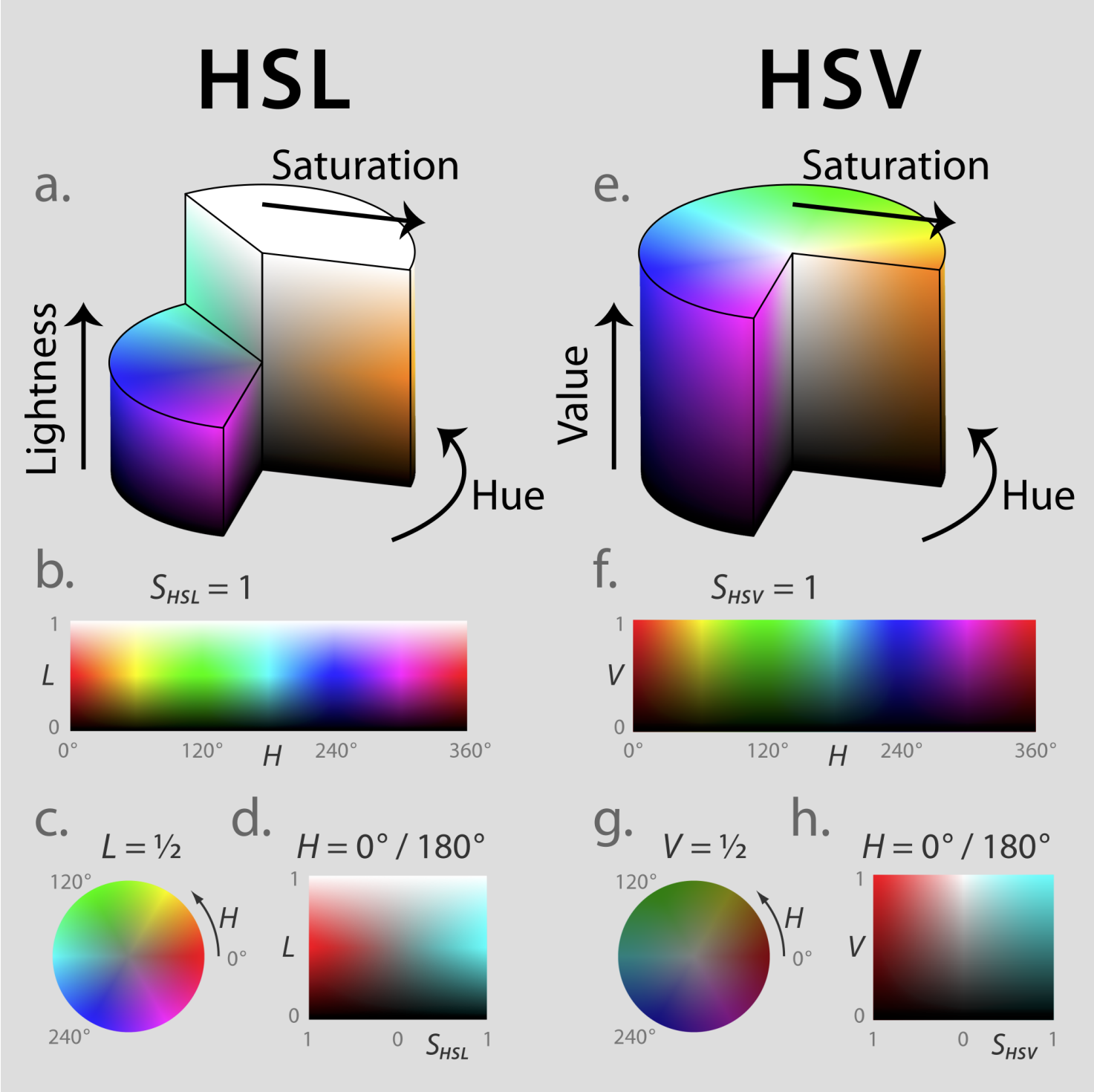

Figure 2. HSV and HSL color space cylinders

Color Conversion Methods

With Chad's Hexadecimal $\Longleftrightarrow$ Decimal (HtoD, DtoH) methods being completed, I have completed the rest of the color mode conversion methods. In doing so, I ran into a minor compatibility issue. That being the difference in the way that RGB value data is stored and the output of the HtoD methods. (Note that this is a dirct an issue with HEX$\Longleftrightarrow$RGB conversion, but an *indirect* issue with HSV and HSL. This is because, as will be show below, converting between HEX and HSL/HSV requires the respective RGB values as a stepping stone.)

Direct Data Discrepancy

The HtoD methods use and return a single, 32-bit/4-byte, signed integer. However, the current RGB conversion methods rely on there being 3, 8-bit/byte, unsigned integers, stored as a tuple. Luckily, the proccess for retrieving these values from the 32-bit integer (Int32) is straight-forward.

HEX to RGB

1. Using the HtoD method, retrieve each byte and store in a `byte` array. Relative to RGB color, the array is structured as such: [Blue value, Green value, Red value, Pos/Neg sign]
2. Remove the Pos/Neg byte, leaving an array of 3 bytes.
3. Because the array is in reverse-RGB order, reverse the array.
4. Cast to `double` and store the respective values in a tuple.

RGB to HEX

1. For each value in an RGB color, convert into a `byte` datatype.
2. Store the values in an array in reverse-RGB order, adding a `0` Pos/Neg byte as the last item.
3. Convert the array into an Int32.
4. Pass the Int32 through the DtoH method to retrieve the hexadecimal color string.

HSV and HSL

I have not found any methods or equations for converting directly to HSV or HSL from a HEX color code. Instead, these conversions are done by first converting the input code to RGB, and then converting the RGB value to the desired output code.

Incompatibility Issues

Although each section works great on its own, they operate differently from each other and cannot be directly connected. In order for the project to be finished, these differences will need to be adapted to each other. So far, this has ranged from simply fixing the output once it is returned, to making large changes in how the sections operate, take input, and return output.

Goals Retrospective

It does not look like I will be able to complete either the ValueDeviation method nor the menus in time for the submission of this DevLog. Additionally, I am unsure if there will be time for the ValueDeviation method at all. However, I will be able to add in the menus post-submission, especially because they are important for the completion of the project.

Last updated 2022-12-08 17:55:59 -0700