KTH Stockholm :: EECS :: CST
Introduction to Visualization and Computer Graphics
DH2320/DD2258, Spring 2022
Tino Weinkauf, Jiahui Liu, Robert Mellberg, Oliver Eriksson, Filip Berendt,
Yineng Wang, Sofiia Ufymtseva, Nils Merkel, Jacob Wahlgren

Homework assignment No. 04

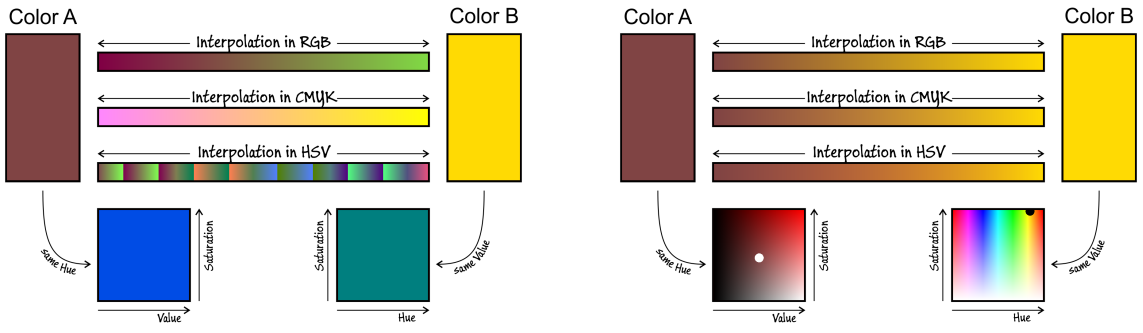
Due 10:00 AM, February 18, 2022

Task 4.1: Color Models and Interpolation — 10 P

Implement color interpolation and manipulation for different color models! The user interface allows you to choose two colors *Color A* and *Color B*. You will interpolate between them in RGB, CMYK, and HSV. Furthermore, you will change two dimensions of a given HSV color and obtain a slice through the HSV color model. The initial setup and the final result can be seen in Fig. 1.

Make sure to interactively change the two input colors *Color A* and *Color B* and observe the changes. Does the choice of color model matter when interpolating colors?

- **Inviwo Workspace:** *File* → *Example Workspaces* → *LabColor* → *ColorInterpolation.inv*
- **File:** `...\Inviwo\modules\labcolor\colorinterpolation.cpp`
- **Functions for color interpolation:** You need to edit the functions `void InterpolateInRGB(...)`, `void InterpolateInCMYK(...)`, and `void InterpolateInHSV(...)`. They all return the interpolation result as a color in the respective color model. All these interpolation functions have three inputs:
  - `const ...& ColorA`: color to be interpolated in the respective color model
  - `const ...& ColorB`: color to be interpolated in the respective color model
  - `const float t`: interpolation parameter in the range [0, 1].
- **Functions for color manipulation in HSV:** You need to edit the functions `void ChangeValueAndSaturation(...)` and `void ChangeHueAndSaturation(...)`. They both return a color in the HSV color model. The functions have three inputs: a HSV color and two parameters to change this color. See the source code comments for details.



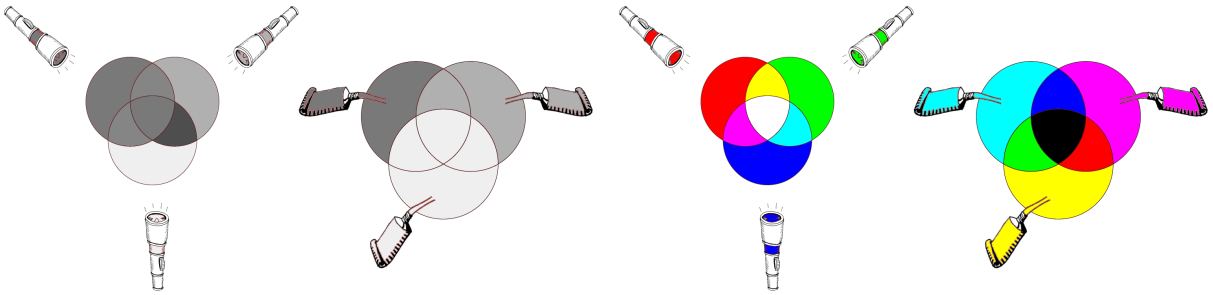
(a) Setup (b) Result

Figure 1: Color Interpolation

Task 4.2: Additive versus Subtractive Color Mixing **4 P**

Implement additive and subtractive color mixing! The user interface allows you to select the three primary colors in both setups. The initial setup and the final result can be seen in Fig. 2. The shown result reproduces the images from the lecture slides, which is a good testing scenario.

- **Inviwo Workspace:** *File* → *Example Workspaces* → *LabColor* → *ColorMixing.inv*
- **File:** `...\Inviwo\modules\labcolor\colormixing.cpp`
- **Function for additive color mixing:** You need to edit the function `void AdditiveColorMixing(...)` which returns the mixed color based on the following input parameters:
  - `const vec3& Color1`: color to be mixed
  - `const vec3& Color2`: color to be mixed
- **Function for subtractive color mixing:** You need to edit the function `void SubtractiveColorMixing(...)` which returns the mixed color based on the following input parameters:
  - `const vec3& ColorIncomingLight`: color of the incoming light
  - `const vec3& ColorSurface`: color of the surface where the light will be reflected



(a) Setup (b) Result

Figure 2: Color Mixing

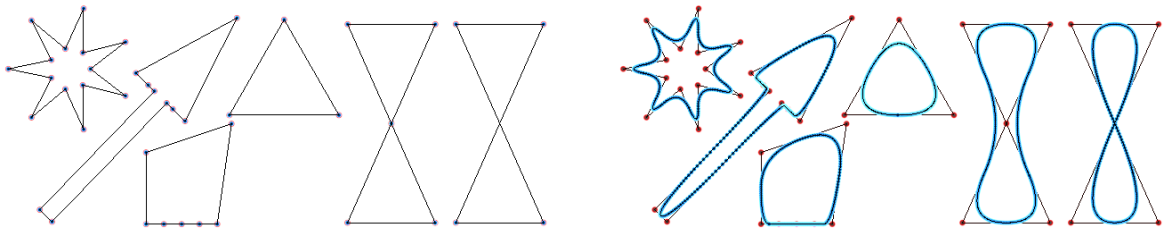
Task 4.3: Chaikin's Corner Cutting **15 P**

Implement Chaikin's corner cutting algorithm! The initial setup and the final result can be seen in Fig. 3.

- **Inviwo Workspace:** *File* → *Example Workspaces* → *LabSubdivision* → *Chaikin.inv*
- **File:** `...\Inviwo\modules\labsubdivision\chaikin.cpp`
- **Function:** You need to edit the function `void Chaikin::CornerCutting(...)` which takes the following parameters:
  - `const std::vector<vec3>& ControlPolygon`: points defining the line segments to be subdivided
  - `const size_t MinNumDesiredPoints`: minimum number of points the resulting curve should contain
  - `std::vector<vec3>& Curve`: points of the resulting curve after applying corner cutting

Task 4.4: Curve Simplification *(Extra Task)* **5 EP**

Chaikin's Corner Cutting creates polylines with many samples. In less curved parts of the polyline, a significantly smaller number of samples would suffice. Implement an algorithm to reduce the number of sample points while retaining the overall shape of the curve as much as possible.



(a) Setup (b) Result

Figure 3: Chaikin's Corner Cutting