Die Lösungen für diese Übung sind abzugeben bis Sonntag den 28.04.2024 um 18:00h.

Farbraum-Konvertierung

Aufbauend auf dem Beispiel aus der Lektion wird in dieser Aufgabe eine Farbkonvertierung im Fragment Shader vorgenommen, sodass wir Farben im HSV Farbraum angeben können.

Änderungen am Übungsformat

Wir haben ein paar Änderungen an dem Framework unternommen, um Euch und uns die Arbeit hoffentlich zu erleichtern. Jede Aufgabe ist jetzt ein vollständiges, abgekapseltes, eigenständiges Programm und die Dependencies werden automatisch geladen um den Projektordner möglichst klein zu halten.

Zur Abgabe ändert Ihr in der `CMakeLists.txt` in Zeile 16 `Aname-Bname-Cname` zu den Namen Eurer Teammitglieder. Wenn Ihr fertig seid, baut Ihr Eure Abgabe und das Buildskript generiert Euch automatisch eine `.zip` mit eurem Programmcode und legt diese im `build`-Ordner ab (z.B. als `cgintro-2a-Aname-Bname-Cname.zip`). Details zum Buildprozess findet ihr in der `README.de.md` innerhalb der jeweiligen Aufgabenordner.

Auf eCampus ladet Ihr dann die Lösungen der praktischen Teilaufgaben als `.zip` und die Lösungen der theoretischen Aufgaben als handschriftliche oder geTeXte `.pdf`-Dokumente hoch:

- `cgintro-2a-Aname-Bname-Cname.zip`
- `cgintro-2b-Aname-Bname-Cname.zip`
- `cgintro-2-theorie-Aname-Bname-Cname.pdf`

Wenn Ihr Anmerkungen, Verbesserungsvorschläge oder Fragen zu dem neuen Framework habt, könnt Ihr uns die immer gerne mitteilen per Mail, oder über den Discord Kanal.

Praktischer Aufgabenteil (6 Punkte)

1. Rendert zuerst ein symmetrisches Kreuz. Hierzu muss in `cgintro-2a/mainapp.cpp` der Code vervollständigt werden. Anschließend sollen in `cgintro-2a/transformation.vert` einfache animierte Transformationen realisiert werden:

a) Eine kreisförmige Translation
b) Eine gleichmäßige Skalierung (d.h. ein Zoom hin und zurück)
c) Eine vollständige Rotation

Tipp: Verwendet `uTime`, um die Transformationen zu animieren!

2. Der vorgegebene Code in `cgintro-2b/mainapp.cpp` soll Farben im HSV Raum bekommen und diese als RGB Farben (interpoliert in einem Viereck) rendern. Dazu muss im fragment shader `cgintro-2b/hsv.frag` die Funktion `vec3 hsv2rgb(vec3)` implementiert werden. Algorithmen dafür findet Ihr hier oder hier.

Das Ergebnis soll dann so aussehen:

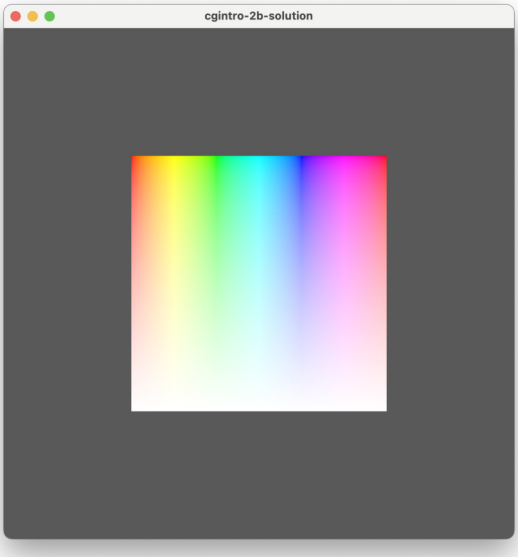

Abbildung 1: HSV Interpolation

Theoretischer Aufgabenteil (4 Punkte)

Gegeben seien 3 Punkte $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2 \in \mathbb{R}^2$ eines Dreiecks $\Delta(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$. Sei weiterhin $\mathbf{p} \in \mathbb{R}^2$ ein weiterer Punkt. *Baryzentrische Koordinaten* bezeichnen jene Koordinaten $\lambda_0, \lambda_1, \lambda_2 \in \mathbb{R}^3$ für die gilt:

$$\begin{aligned} \lambda_0\mathbf{p}_0 + \lambda_1\mathbf{p}_1 + \lambda_2\mathbf{p}_2 &= \mathbf{p} \\ \lambda_0 + \lambda_1 + \lambda_2 &= 1 \end{aligned}$$

Mit der zweiten Bedingung ist sichergestellt, dass die baryzentrischen Koordinaten eindeutig sind. Im weiteren nehmen wir an, dass $\lambda_i \geq 0, i \in \{0, 1, 2\}$. Letztere Bedingung impliziert, dass $\mathbf{p}$ in dem von $\mathbf{p}_0$, $\mathbf{p}_1$ und $\mathbf{p}_2$ aufgespannten Dreieck liegt.

Zeigen Sie, dass die baryzentrischen Koordinaten $\lambda_0, \lambda_1, \lambda_2$ zu einem Punkt $\mathbf{p} \in \Delta(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ durch folgende Flächenverhältnisse gegeben sind:

$$\lambda_0 = \frac{A(\Delta(\mathbf{p}, \mathbf{p}_1, \mathbf{p}_2))}{A(\Delta(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2))}$$
$$\lambda_1 = \frac{A(\Delta(\mathbf{p}_0, \mathbf{p}, \mathbf{p}_2))}{A(\Delta(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2))}$$
$$\lambda_2 = \frac{A(\Delta(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}))}{A(\Delta(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2))}$$

Dabei bezeichnet $A(\Delta(\mathbf{a}, \mathbf{b}, \mathbf{c}))$ die Fläche des durch $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^2$ aufgespannten Dreiecks.

Hinweis: Zeigen Sie die Behauptung zunächst für den Spezialfall $\mathbf{p}_0 = (0, 0)^T$. Stellen Sie ein Gleichungssystem für die λ_i auf und lösen Sie dieses (z.B. mit der Cramerschen Regel). Benutzen die dann $A(\Delta(\mathbf{0}, \mathbf{a}, \mathbf{b})) = \frac{|\det(\mathbf{a}\,\mathbf{b})|}{2}$.