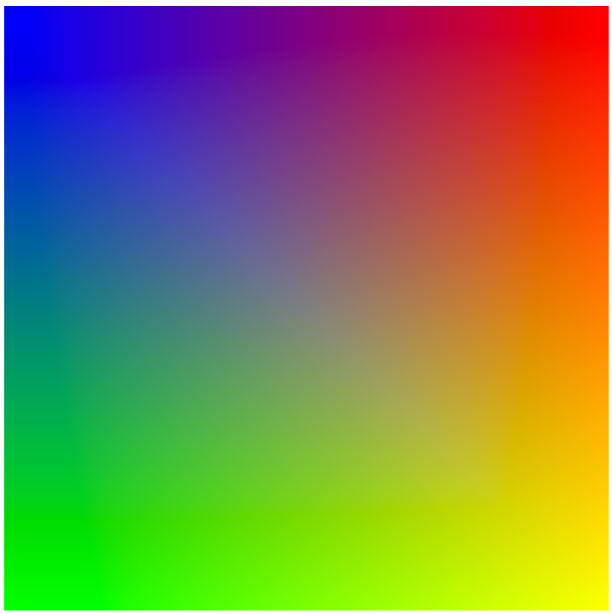
Human-Computer Interaction
ICG ToDo Liste Übung 10

Aufgabe 0 (Vorbereitung)

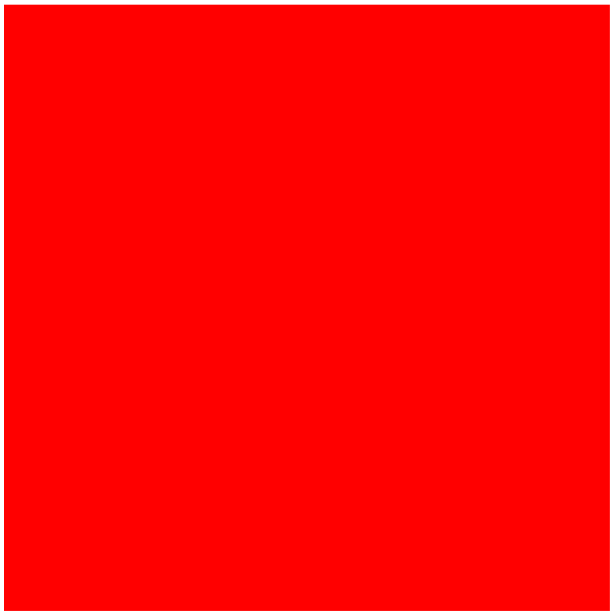
Diese Übung befasst sich mit Shading im Fragment-Shader. Dazu können Sie entweder Ihr vorhandenes Projekt verwenden, in dem Sie ein Rechteck erstellen, welches über den ganzen Bildschirm gespannt ist oder verwenden Sie das Beispielprojekt im Moodle in Woche 10.

Wenn Sie ihr eigenes Projekt verwenden möchten:
Achten Sie darauf, dass das Rechteck im Screenspace liegt, damit unabhängig von der Kameraposition angezeigt wird. Weisen Sie zusätzlich den vier Vertices die Farben Rot, Blau, Grün und Gelb zu. Nachdem Sie die Farben zugewiesen haben, sollte Ihr Rechteck wie folgt aussehen.



Aufgabe 1

- **Aufgabe:** Färben Sie das Rechteck im Fragment-Shader rot.



- **Tipps:**
  - Denken Sie daran, den Farbvektor in die korrekte Variable innerhalb des Fragment-Shaders zu schreiben.

Aufgabe 2

- **Aufgabe:** Stellen Sie den folgenden Farbverlauf in Graustufen dar.

- **Tipps:**
  - Denken Sie daran, den Farbvektor in die korrekte Variable innerhalb des Fragment-Shaders zu schreiben.
  - Um Grauwerte darzustellen benötigen Sie nur einen Parameter, welchen Sie dann allen Farbkomponenten zuweisen.

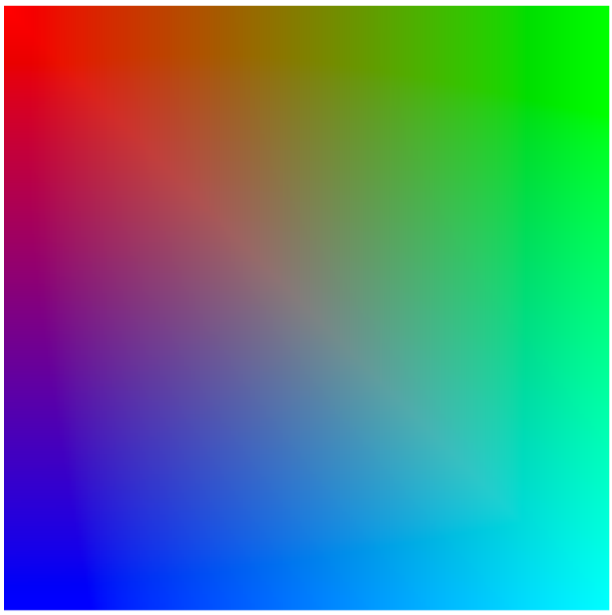
Aufgabe 3

- **Aufgabe:** Benutzen Sie nur den Blaukanal des urspünglichen Farbverlaufs um mit Hilfe des Fragment-Shaders das gezeigte Bild in Graustufen darzustellen.

- **Tipps:**
  - Denken Sie daran, in den Farbvektor in die korrekte Variable innerhalb des Fragment-Shaders zu schreiben.
  - Um Grauwerte darzustellen benötigen Sie nur einen Parameter, welchen Sie dann allen Farbkomponenten zuweisen.

Aufgabe 4

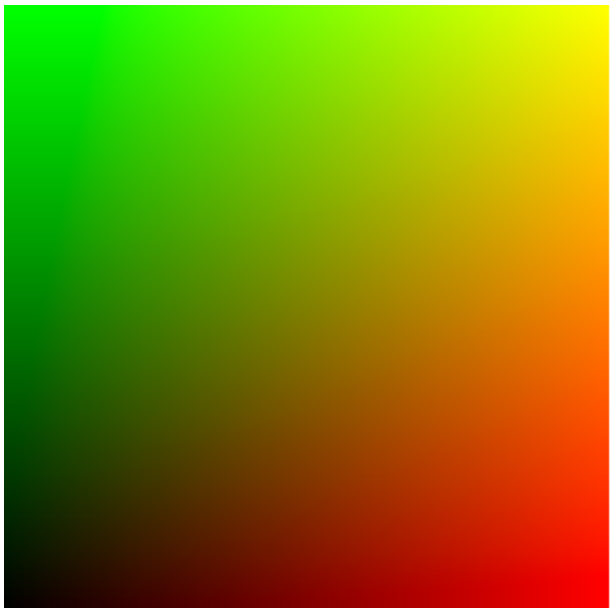
- **Aufgabe:** Erzeugen Sie mit Hilfe des Fragment-Shaders folgende Farbverteilung auf ihrem Rechteck, indem Sie die Farbkanäle miteinander vertauschen.



- **Tipps:**
  - Denken Sie daran, den Farbvektor in die korrekte Variable innerhalb des Fragment-Shaders zu schreiben.
  - Überlegen Sie sich, welche Farbwerte sie tauschen müssen, um das Bild zu erhalten.

Aufgabe 5

- **Aufgabe:** Erzeugen Sie mit Hilfe des Fragment-Shaders folgende Farbverteilung auf ihrem Rechteck in dem Sie den Rot-Kanal abhängig von der x-Position und den Grün-Kanal abhängig von der y-Position des Fragments bestimmt.

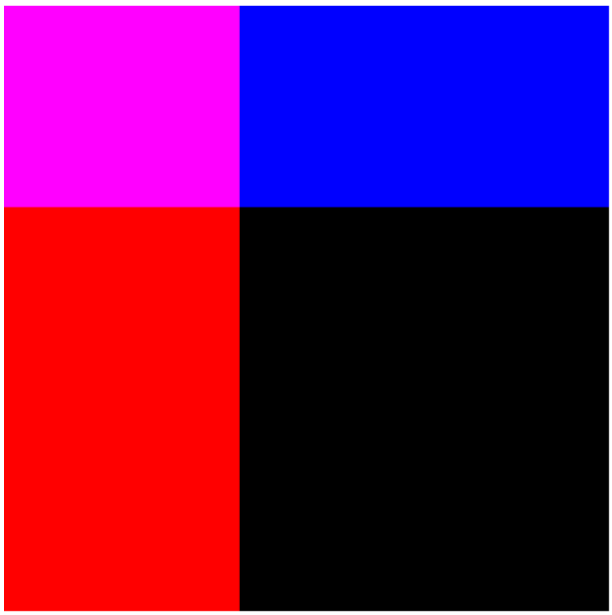


- **Tipps:**
  - Denken Sie daran, den Farbvektor in die korrekte Variable innerhalb des Fragment-Shaders zu schreiben.
  - Überlegen Sie sich auch wie sie die Fragmentposition ausnutzen können, um herauszufinden, wo Sie sich innerhalb des Rechtecks befinden.
- **Wichtige Dokumentation**

```
gl_FragCoord;
```

Aufgabe 6

- **Aufgabe:** Erzeugen Sie mit Hilfe des Fragment-Shaders folgende Farbverteilung auf ihrem Rechteck in dem Sie eine konstante Koordinate als Mittelpunkt setzen und die Quartale wie dargestellt färben.

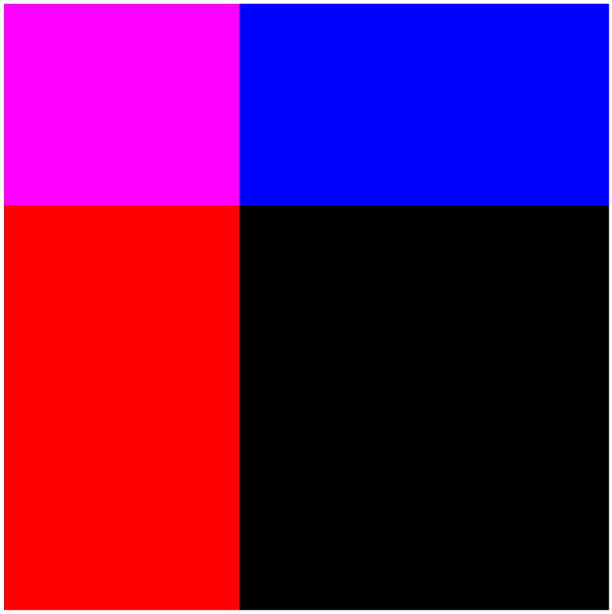


- **Tipps:**
  - Denken Sie daran, den Farbvektor in die korrekte Variable innerhalb des Fragment-Shaders zu schreiben.
  - Überlegen Sie sich auch wie sie die Fragmentposition ausnutzen können, um herauszufinden, wo Sie sich innerhalb des Rechtecks befinden.
- **Wichtige Dokumentation**

```
gl_FragCoord;
```

Aufgabe 7

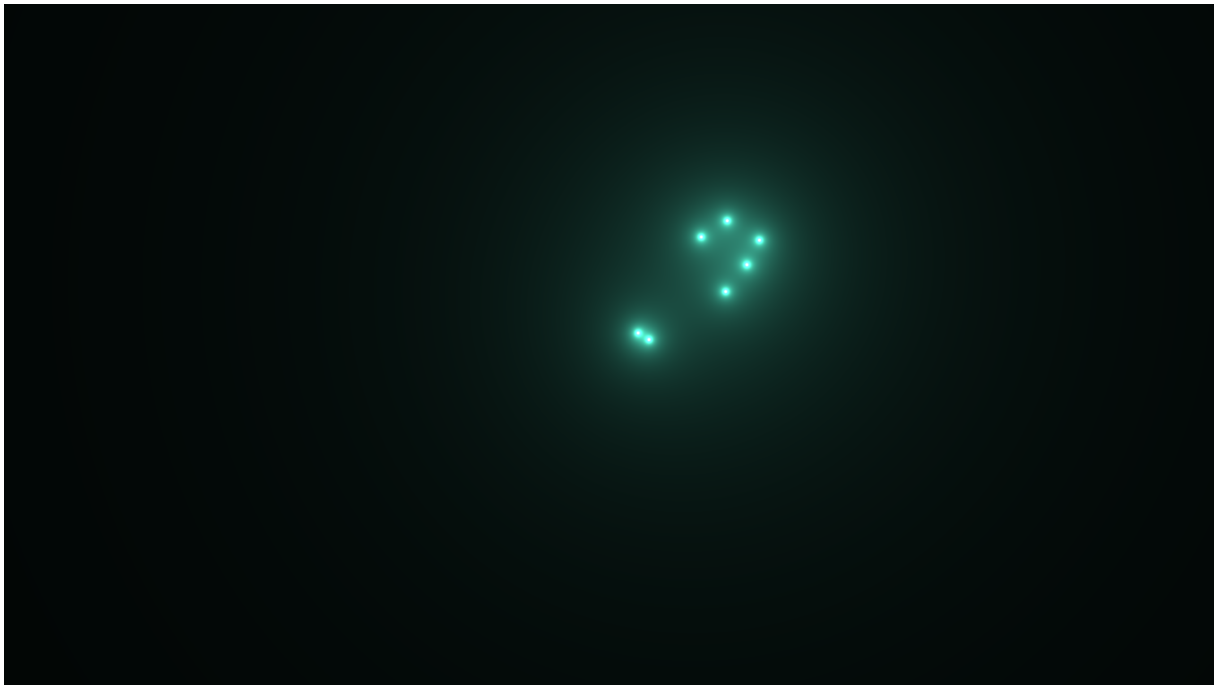
- **Aufgabe:** Erzeugen Sie mit Hilfe des Fragment-Shaders folgende Farbverteilung auf ihrem Rechteck in dem Sie beim Mausklick die Position der Maus als Mittelpunkt setzen und die Quartale wie dargestellt färben.



- **Tipps:**
  - Denken Sie daran, den Farbvektor in die korrekte Variable innerhalb des Fragment-Shaders zu schreiben.
  - Überlegen Sie sich auch, wie sie die Fragmentkoordinaten ausnutzen können, um herauszufinden, wo Sie sich innerhalb des Rechtecks befinden.
  - Denken Sie daran, eine Renderschleife zu implementieren.
  - Denken Sie auch daran, den Mausposition entsprechend vom Hauptprogramm an den Shader zu übergeben.
- **Wichtige Dokumentation**

```
gl_FragCoord;
MouseEvent.ClientX;
MouseEvent.ClientY;
```

Aufgabe 8 (Optional)



```
precision mediump float;
void mainImage( out vec4 fragColor, in vec2 fragCoord ) {
    float k = abs(sin(iTime * 0.8)) * 0.3;
    vec2 p = (fragCoord.xy * 2.0 - iResolution.xy) / min(iResolution.x,
        iResolution.y);
    vec3 destColor = vec3(0.3, 0.6 + k, 1.0 - k);
    float f = 0.0;
    for(float i = 0.0; i < 7.0; i++){
        float s = sin(0.7 * iTime + (i * 0.5) * iTime) * 0.2
                                  + 0.3 * sin(iTime * 0.6);
        float c = cos(0.2 * iTime + (i * 0.5) * iTime) * 0.2
                                  + 0.3 * cos(iTime * 0.2);
        f += 0.01 / abs(length(p*(abs(sin(iTime * 0.1)) + 0.1) + vec2(c, s)));
    }
    fragColor = vec4(vec3(destColor * f), 1.0);
}
```

Abbildung 1: Wer Lust hat, kann den Shader einmal nach ShaderToy.com kopieren und einen eigenen interessanten Shader schreiben.