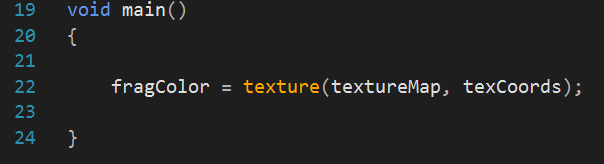
Praktikum 1

Lisa Obermaier, Simon Thum

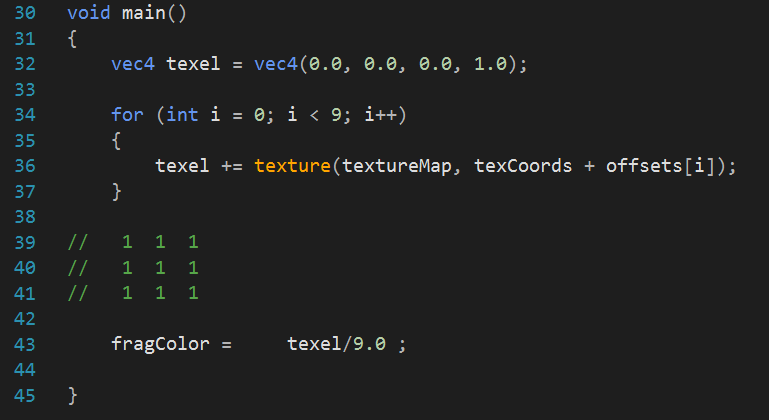
* 1. FragmentShaderPassthough.glsl Vs. FragmentShaderBewMit.glsl



Der Passthrough-Filter gibt lediglich den aktuellen Pixel wieder, er verursacht also keine Veränderung:

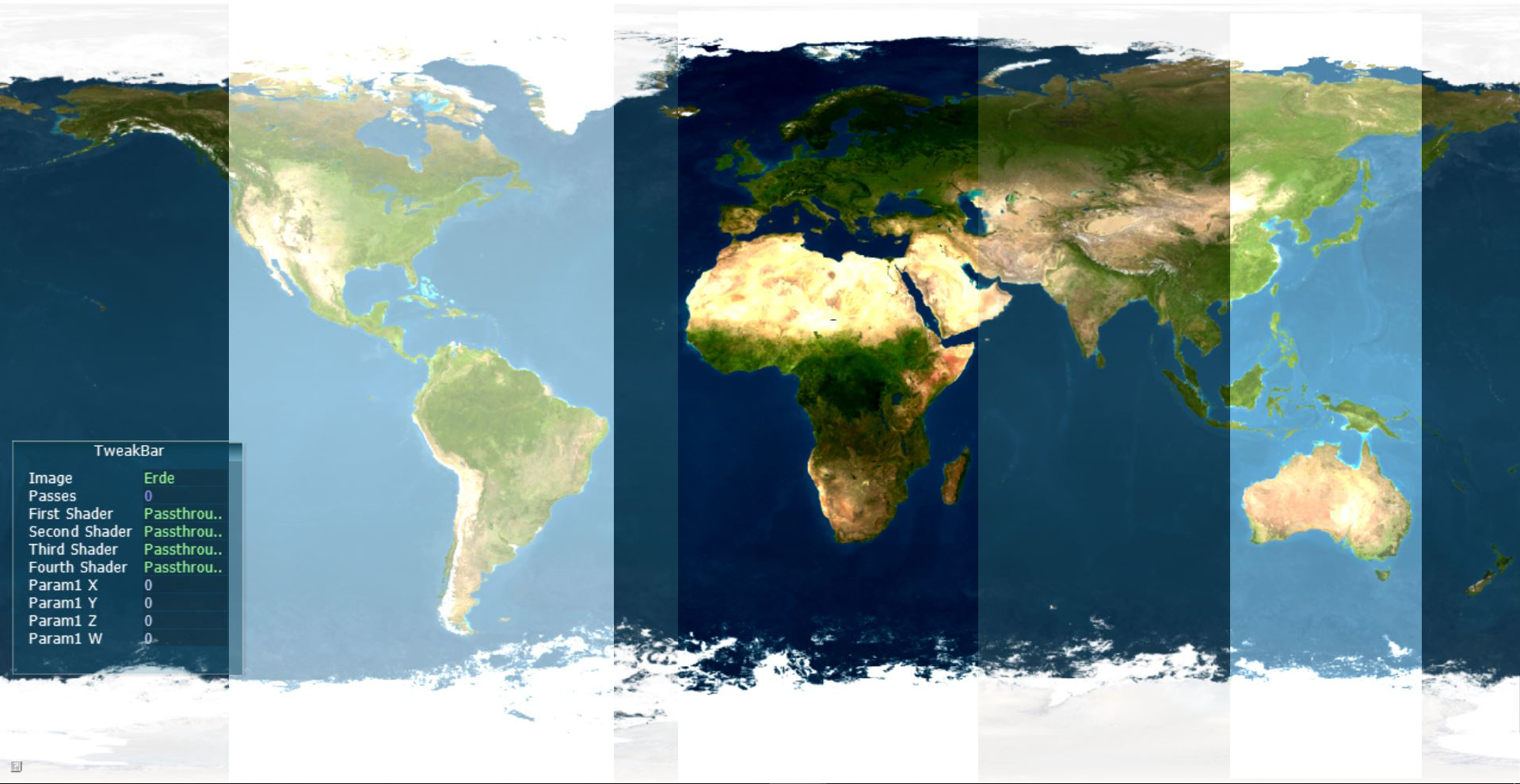


Der Bewegte-Mittelwert-Filter dagegen durchläuft ein neun-elementiges Array, addiert die Farbwerte und gibt dem aktuellen Pixel den berechneten Mittelwert aller neun Pixel. Wie auf dem Bild oben zu sehen, ist das Bild folglich verschwommen.



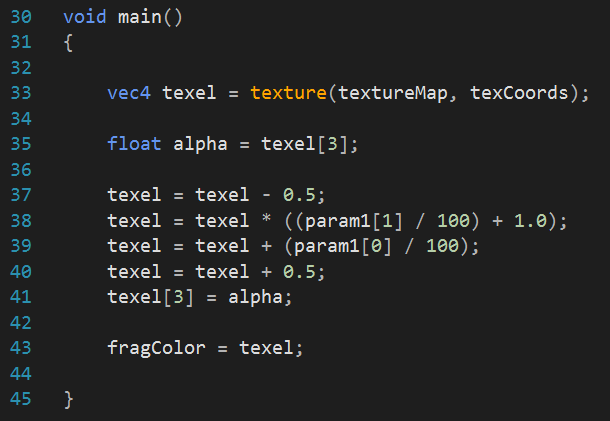
* 1. Weitere Filter

1. FragmenShaderBrightness\_Contrast.glsl

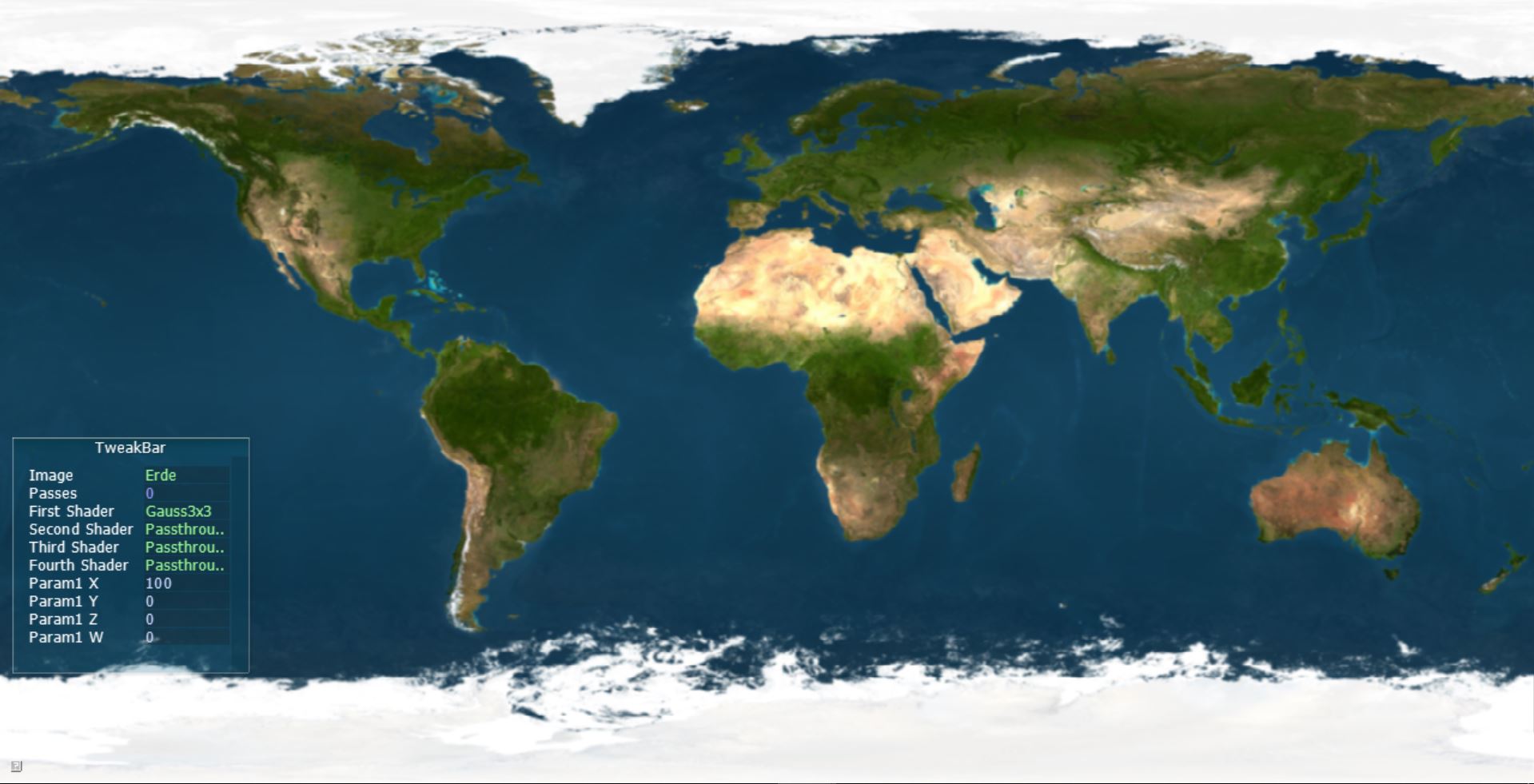


Um Brightness und Color zu modifizieren, müssen zwei Parameter veränderbar sein. param1[0] ist für die Helligkeit und param1[1] für den Kontrast verantwortlich. Zunächst wird die Darstellung des Pixels verschoben, anschließend der Kontrast multipliziert, dann die Helligkeit addiert und zuletzt die anfängliche Verschiebung wieder zurückgesetzt. Um den richtigen Alpha-Wert zu setzen, wird dieser vor den Operationen zwischengespeichert.

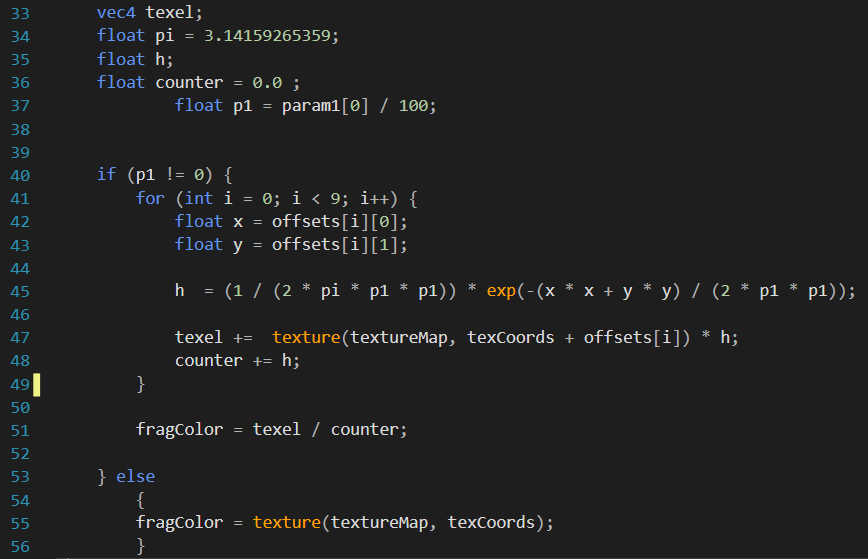
Im oben abgebildeten Bild ist Amerika mit modifizierter Helligkeit, Europa und Afrika mit modifiziertem Kontrast und Australien mit kombinierter Modifikation abgebildet.



1. FragmentShaderGauß3x3.glsl



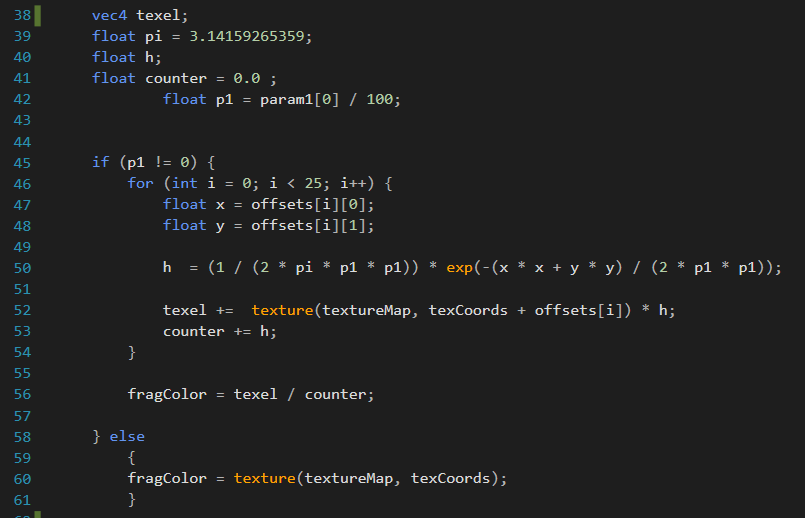
Um den Gauß-Tiefpass-Filter zu realisieren, verwenden wir eine entsprechende Formel für den Filter, die mit x und y aus dem Vektor und einem modifizierbaren Sigma (in unserem Beispiel param1[0]) ein h berechnet, das mit dem aktuell betrachteten Pixel im texel für alle neun Iterationen addiert wird. Anschließend wird der texel-Wert durch die Summe der berechneten h’s geteilt, um einen Mittelwert in Abhängigkeit des eingestellten Sigmas zu erhalten.



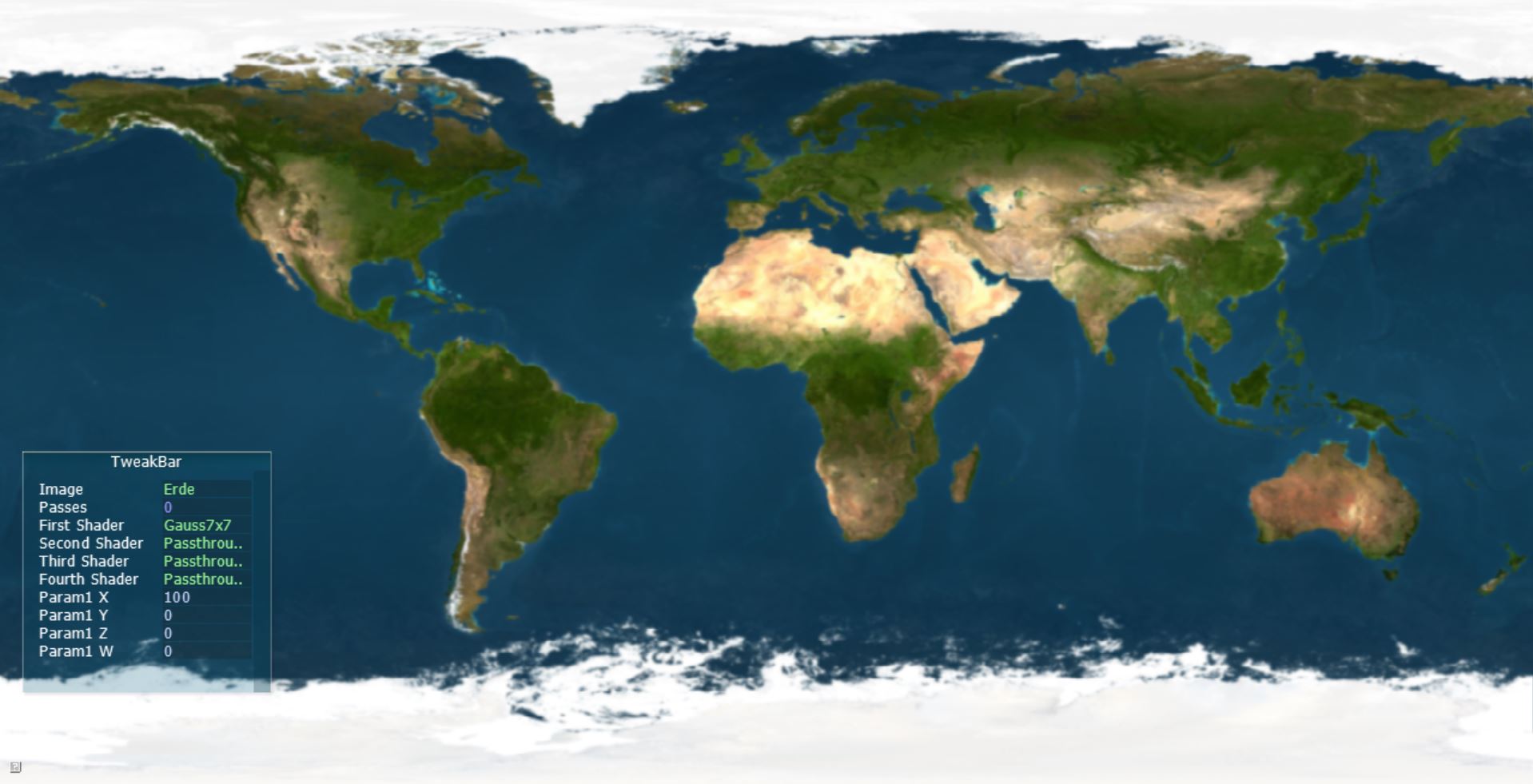
1. SegmentShaderGauß5x5.glsl



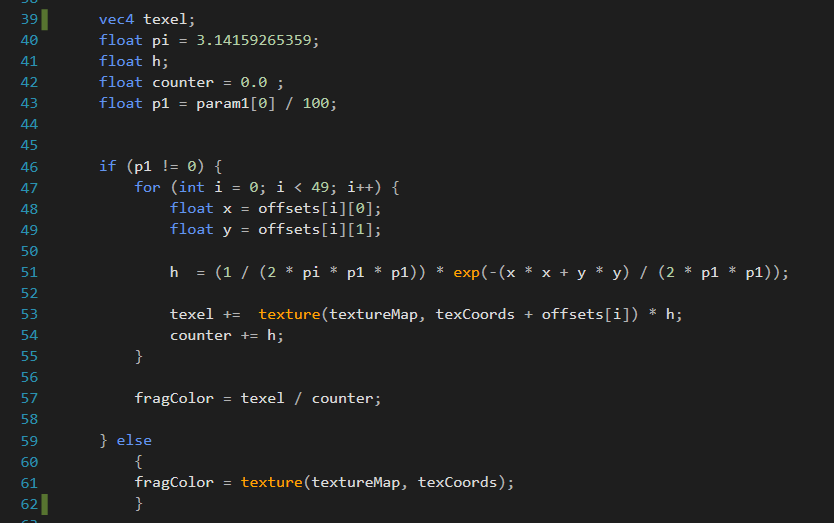
Der Gauß-Tiefpass-Filter mit einem 5x5-Fenster funktioniert äquivalent zu einem 3x3-Fenster, doch mit einem größeren Fenster wird die Veränderung im Bild deutlicher.



1. SegmentShaderGauß7x7.glsl



Mit einem 7x7-Fenster im Gauß-Tiefpass-Filter ist das Verschwimmen des Bildes nun gut wahrzunehmen. Ansonsten funktioniert er äquivalent zum Gauß-Tiefpass-Filter mit 3x3 oder 5x5-Fenster. Allerdings macht sich hier die Performance schon deutlich bemerkbar, denn nur die 49 Iterationen in der Schleife, werden nur noch etwa 100 FPS erreicht.



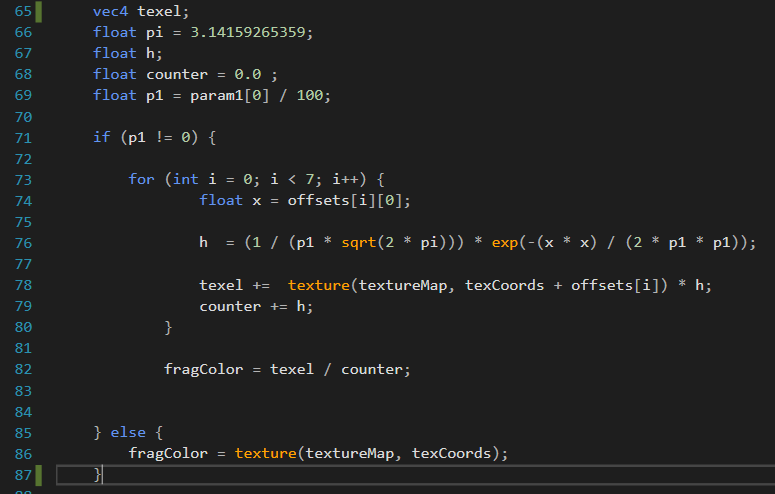
Gauß Tiefpass-Filter realisiert mit zwei Shadern:

Um die Performance zu verbessern, sollen den oben beschriebenen Gauß-Tifpass-Filter mit 7x7-Fenster zwei Shader übernehmen, die jeweils nur über die X- oder die Y-Koordinate iterieren und so weniger als 49 Iterationen für die Darstellung der Pixel benötigen.

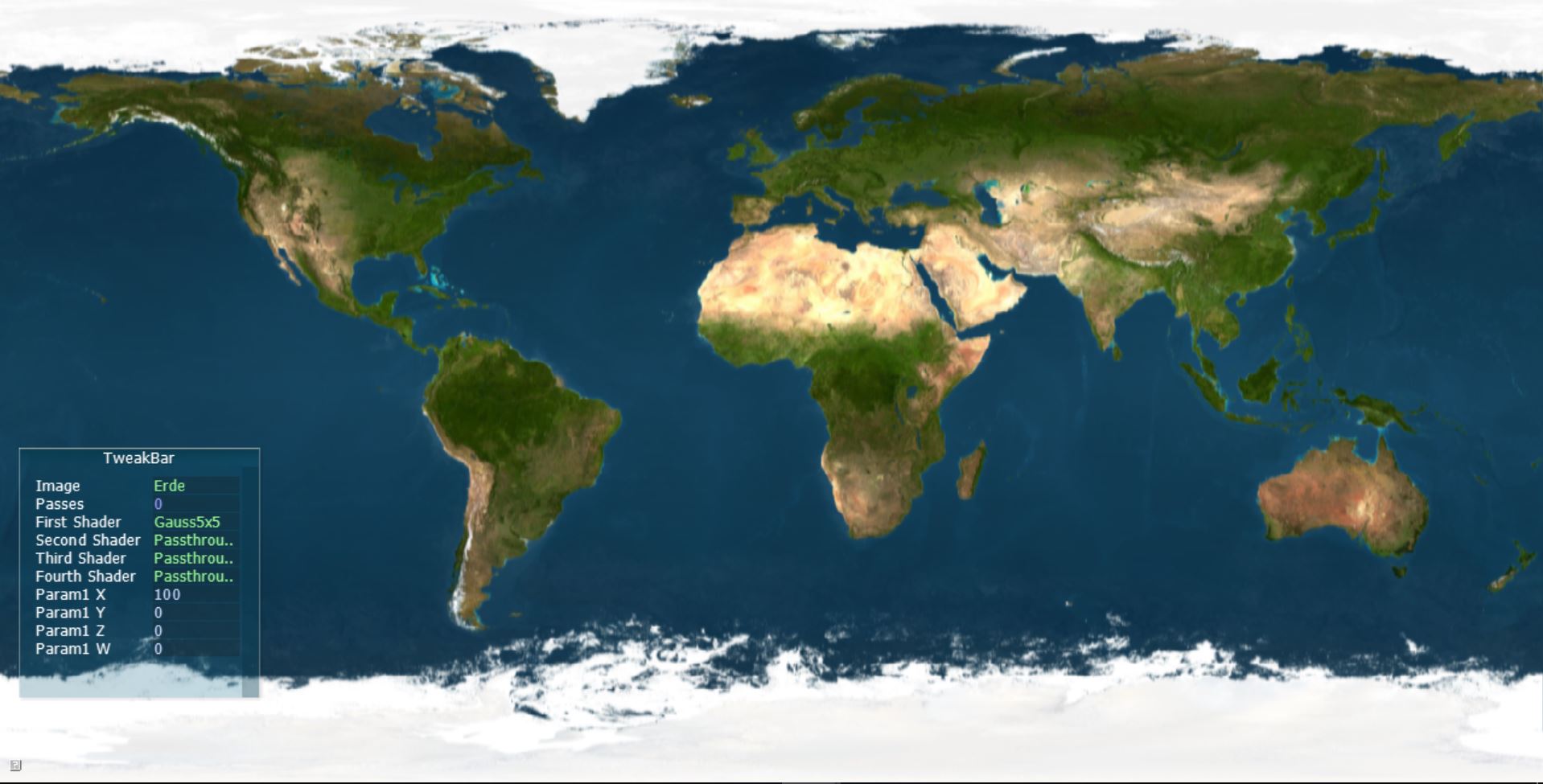
Shader in X-Richtung:



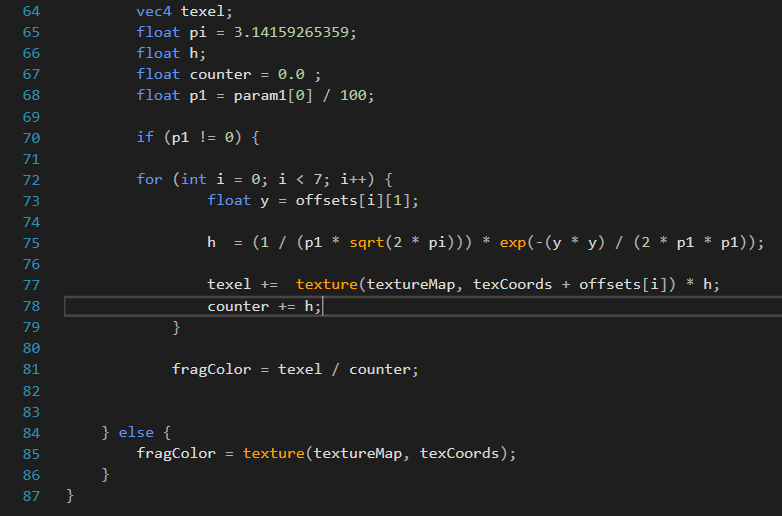
Mit einer Formel für den Gauß-Tiefpass-Filter, die lediglich eine Koordinate (in diesem Fall x) mit einem gegebenen Sigma berechnet, wird hier, analog zum 7x7-Gauß-Tiefpass-Filter von oben, das aktuelle Pixel mit dem errechneten h multipliziert, alles addiert und im Anschluss durch die Summe der errechneten h’s geteilt. So kann ein, der modifizierbaren Gauß-Glocke entsprechender, Farbton über das 7x7-Fenster angenommen werden. Im Bild oben ist erkennbar, dass die Pixel horizontal verwischen.



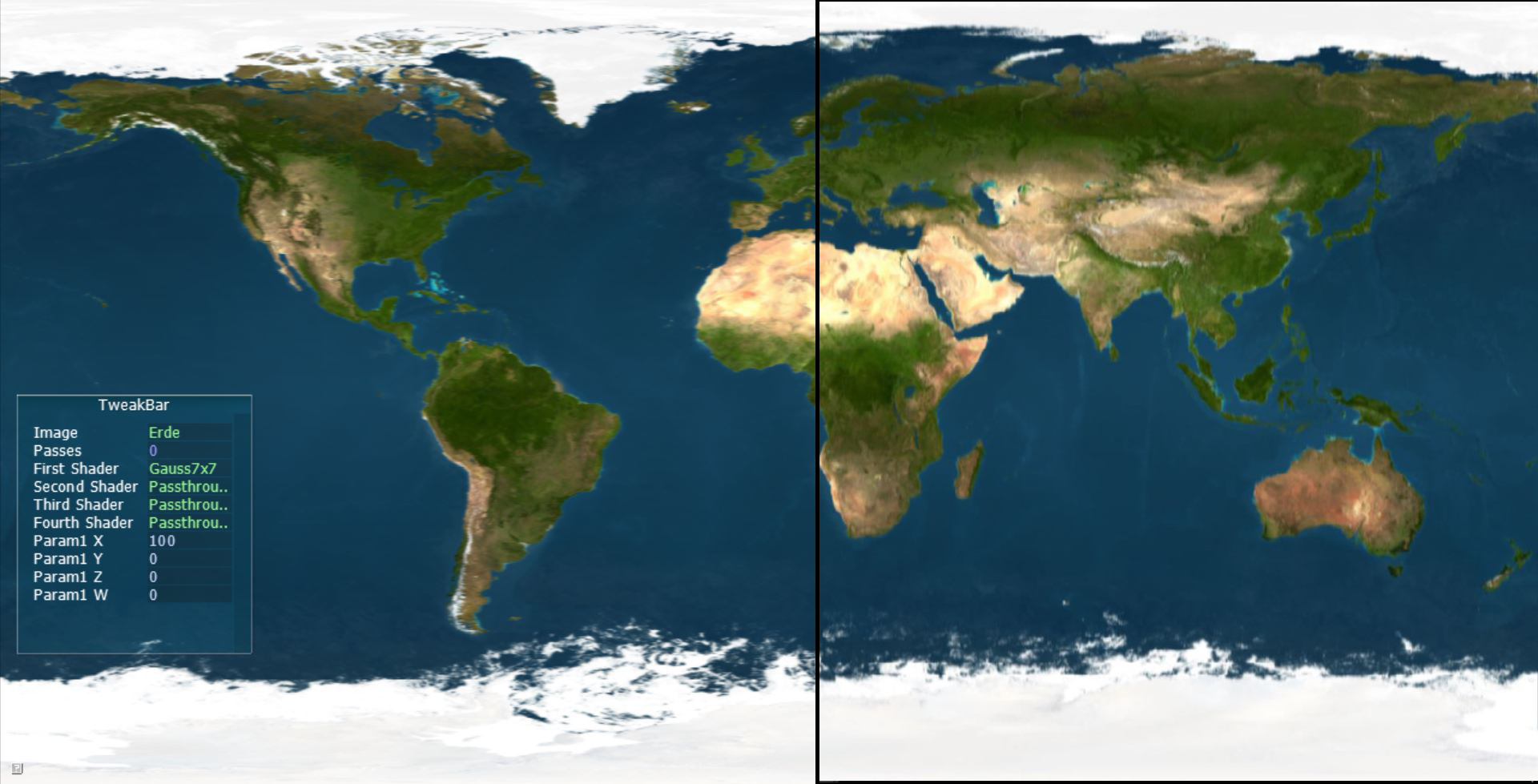
Shader in Y-Richtung:



Analog zur X-Richtung oben, verwischen die Pixel in diesem Bild horizontal. Die Berechnung bleibt gleich, mit dem Unterschied, dass die y-Koordinate zur Berechnung verwendet wird.

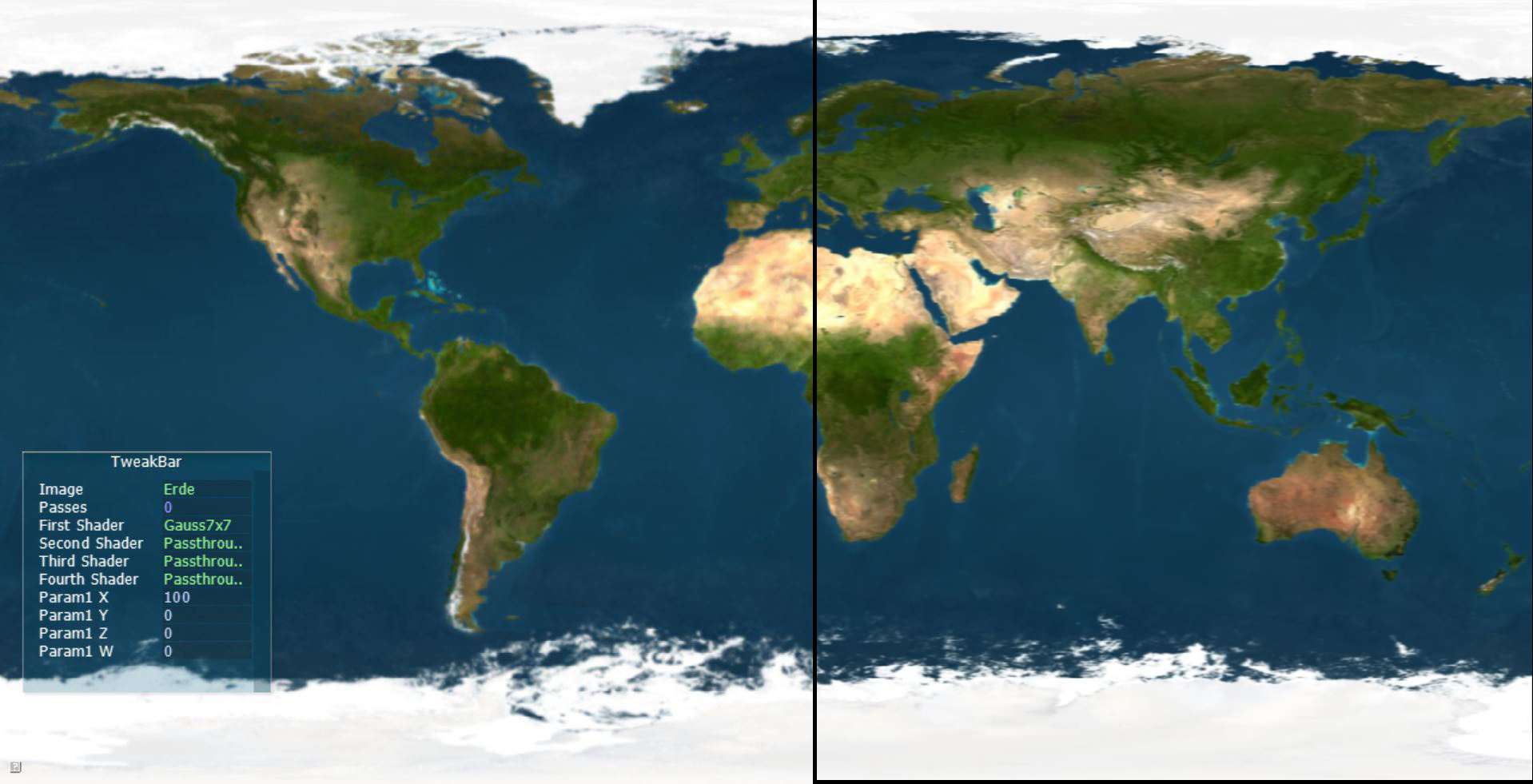


Vergleich Shader in X-Richtung mit Shader in Y-Richtung:



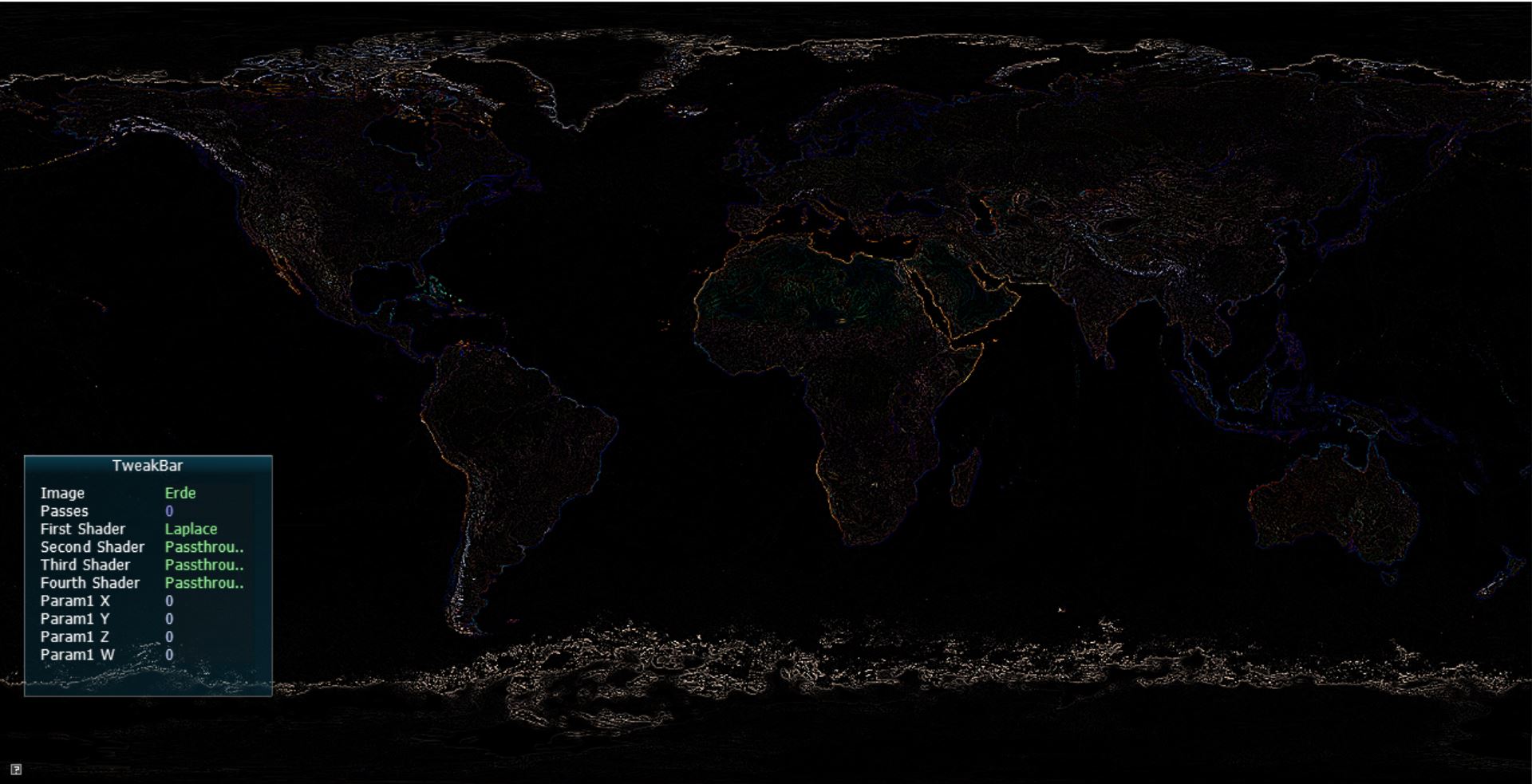
In diesem Bild oben sieht man nun den Unterschied zwischen der Verwischung auf der horizontalen und der Verwischung auf der vertikalen Linie.

Vergleich 7x7-Gauß-Tiefpass-Filter mit zwei getrennten Filtern nach X und Y:



Obiges Bild zeigt nun einen Vergleich der Filter über 49 Iterationen und zweimal über 7 Iterationen. Es ist kaum ein Unterschied erkennbar und mit Hilfe der zwei überlagernden Filter sind wieder FPS von 250 bis sogar 300 möglich, also eine Verbesserung um bis zu 200%.

1. FragmentShaderLaplace.glsl



Mit Hilfe des zweidimensionalen Laplace-Filters (siehe <https://de.wikipedia.org/wiki/Laplace-Filter>), können über eine Addition der texel die Kanten extrahiert werden.

Der Vektor sieht folgendermaßen aus: 0 -1 0

-1 4 -1

0 -1 0

und entsprechend werden die Pixel im Umfeld multipliziert. Das aktuelle texel wird mit 4 Multipliziert und die umliegenden texel oben, rechts, unten und links mit (-1). Diese Werte aufaddiert ergeben den neuen Farbwert des aktuellen Pixels.

