|  |
| --- |
|  |

**Shader in OpenGL**

**Computer Graphik Projekt**

des Studiengangs Informationstechnik an

der Dualen Hochschule Baden-Württemberg Stuttgart

21.12.2022

|  |  |
| --- | --- |
| **Matrikelnummer** | 8803945, XXX |
| **Kurs** | TINF20IN |

Inhaltsverzeichnis

[Abkürzungsverzeichnis III](#_Toc122593120)

[1 Einleitung 1](#_Toc122593121)

[1.1 Ausgangssituation und Problembeschreibung 1](#_Toc122593122)

[1.2 Zielsetzung 1](#_Toc122593123)

[1.3 Vorgehensweise 1](#_Toc122593124)

[1.4 Aufbau der Arbeit 1](#_Toc122593125)

[2 Grundlagen 2](#_Toc122593126)

[3 Anforderungen 3](#_Toc122593127)

[4 Implementierung 4](#_Toc122593128)

[5 Zusammenfassung und Ausblick 5](#_Toc122593129)

[Abbildungsverzeichnis IV](#_Toc122593130)

[Tabellenverzeichnis V](#_Toc122593131)

[Literaturverzeichnis VI](#_Toc122593132)

[Anhang VII](#_Toc122593133)

Abkürzungsverzeichnis

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | PLM |  | Product Lifecycle Management |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Einleitung

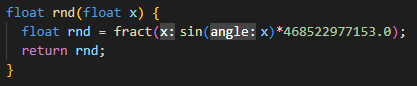
## Zielsetzung

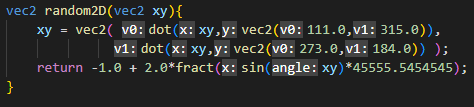
## Vorgehensweise

## Aufbau der Arbeit

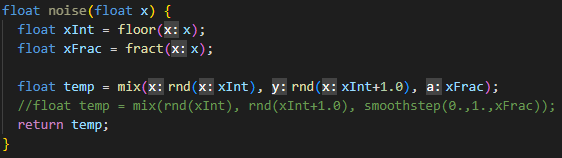
# Grundlagen

Das erste Prinzip ist Random. Hierbei geht es darum zufällige Zahlen zu erhalten um diese in dem Shader verwenden zu können. Es ist nicht einfach in einer Codeumgebung an Zufällig generierte Werte zu gelangen. Aus diesem Grund werden diese durch eine Funktion generiert.

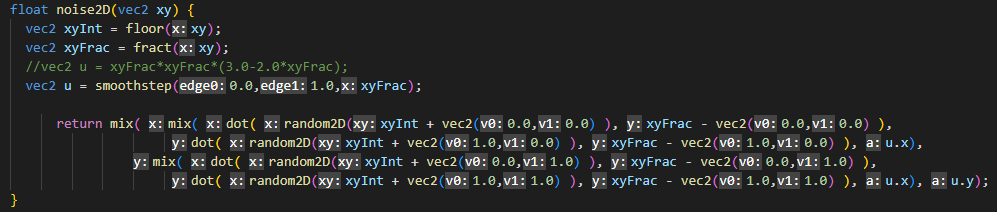
Dazu wird zuallererst der Bruchteil der Werte aus einer Sinuskurve genommen. Diese sind noch nicht wirklich zufällig, aber durch eine große Multiplikation wird die Sinus Kurve in immer kleinere Teile unterteilt. Wenn von der veränderten Sinus Kurve der Bruchteil verwendet wird erhält man zufällige Zahlen.

Die Zufallsmethode reicht für diesen Shader in dieser Form nicht aus, da diese Funktion nur 1-Dimensionelle Zufallswerte generieren kann und 2-Dimensionale Werte also Vektoren gebraucht werden. Um dies zu erreichen wird die dot() Funktion verwendet. Diese gibt einen float Wert zwischen 0.0 und 1.0 zurück je nachdem wo sich zwei Vektoren im Verhältnis zueinander befinden. Nachdem einer Variable der Wert der Funktion dot() zugewiesen wird, ist der Rest für die 2D Zufalls Funktion gleich wie darüber beschrieben.

Durch diese Funktion werden die Zufälligen 2D Werte für den Shader generiert.

Das nächste verwendete Prinizip ist Noise. Noise kann in vielen verschiedenen Dingen im Leben erkannt und bedrachtet werden, wie zum Beispiel bei den Wellen im Mehr oder den Wolken im Himmel. Im ersten moment kann das Verhalten von diesen Dingen als Zufall abgetan werden, aus diesem Grund ist die Implementierung von Noise ähnlich wie die der Random Funktionen. Um Noise zu generieren wird ein float Wert in seinen Ganzzahl-Wert und Bruchteil-Wert zerlegt. Dazu werden die Funktionen floor() und fract() verwendet. Dannach wird mit diesen Werten die Zufallsfunktion aufgerufen und das Ergebnis wird über die Funktion mix() berrechent.

Allerdings wird für diesen Shader ein 2-Dimentionaler Noise Wert benötigt der durch diese Funktion nicht generiert werden kann. Um einen solchen Wert generieren zu können, wird ein vec2 anstatt des float Parameters verwendet. Wie im folgenden Bild zu sehen ist bleibt die Grundfunktion bestehen und es werden nur Änderungen vorgenommen um einen vec2 anstatt des floats verwenden zu können.



# Anforderungen

## Schreiben Sie einen GLSL Shader in WebGL und dokumentieren Sie seine Funktionsweise. Bewertet wird Kreativität, Verständlichkeit der Dokumentation und die Implementierung. Aus der Dokumentation sollte hervorgehen, auf welchen theoretischen Prinzipien das bildgebende Verfahren basiert, wie es algorithmisch umgesetzt werden kann und wie es implementiert wurde. Als Orientierung für eine sehr gute Dokumentation dient <https://thebookofshaders.com/>. Ein Shader wird als nicht sonderlich kreativ bewertet, wenn er nur eine geringe Variabilität zu Standard Beispielen aus gängiger Literatur oder Seiten wie <https://thebookofshaders.com/> oder https://www.shadertoy.com/ aufweist. (100 Punkte)

Stichpunkte

* Shader implementieren
* Funktionsweise Dokumentieren
  + Theoretische Prinzipien
  + Algorithmische Umsetzung
  + Implementierung

# Konzepte des Shaders (Dokumentation)

## Hintergrund

Verwendete theoretische Prinzipien: Random, Noise, brownianMotion, fraktioneleBrownianMotion

## Viel-Ecke

## Bewegtes Muster

## Kunst in den Viel-Ecke

Verwendete theoretische Prinzipien: Random2D, Noise2D, Border?, Shapes?

Um die Kunst in den Viel-Ecken zu generieren werden 4 Prinzipen verwendet. Die ersten beiden stehen unter dem Kapitel Grundlagen, da diese in dem Shader an mehreren Stellen verwendet werden.

Das nächste theoretische Prinzip ist das des Kreises. Um einen Kreis zu erstellen wird ein Punkt benötigt an dem der Mittelpunkt des Kreises ist. Zusätzlich wird noch ein radius benötigt. Dieser kann auf verschiedene Arten erhalten werden wie zum Beispiel durch die length() oder distance() Funktion. Diese Funktionen verwenden als Parameter den Mittelpunkt des Kreises.

# Implementierung

VSCode

Wie man es ausführt

Extentions

# Zusammenfassung und Ausblick

Abbildungsverzeichnis

[Abbildung 2‑1: PLM-Kreislauf, basierend auf [2] 2](#_Toc59527846)

Tabellenverzeichnis

[Tabelle 2‑1: Übersicht der HTTP-Methoden, basierend auf [17, S. 604] 2](#_Toc59527850)

Literaturverzeichnis

[1] Werkzeugmaschinenlabor WZL der Rheinisch-Westfälischen Technischen Hochschule (RWTH) Aachen (Hg)., *Product Lifecycle Management (PLM).* [Online]. Verfügbar unter: http://www.plm-info.de/de/d7bc5950658c5c08c1257e99002f4d59.html (Zugriff am: 2. Juli 2020).

[2] Andrew S. Tanenbaum und Maarten van Steen, *Verteilte Systeme Prinzipien und Paradigmen*. Pearson Studium 2008, 2008.

Anhang

[1 Auszug aus … VIII](#_Toc59527951)

[2 Klassendiagramm IX](#_Toc59527952)

1. Auszug aus …