Simulation of AVM Blood Flow using VFX Graphs

Dokumentation

Otto-von-Guericke-Universität

Fakultät für Informatik

Arbeitsgruppe für Visualisierung

Autoren:
Ulrike Sprengel
Moritz Drittel
Sarah Mittenentzwei

Betreuer:
Dr.-Ing. Patrick Saalfeld
Prüfer:
Prof. Dr.-Ing. Bernhard Preim

01.04.2020 - 13.10.2020

Inhaltsverzeichnis

1 Einleitung

In diesem Abschnitt werden eine Motivation zum Thema und die gesetzte Aufgabenstellung behandelt.

1.1 Motivation

Arteriovenöse Malformationen (kurz AVMs) sind Gefäßmissbildungen, in denen die Blut zuführenden Arterieren ohne zwischengeschaltetes Kapillarbett direkt mit den Venen verwachsen. Das Zentrum dieser Gefäßmissbildung wird als Nidus bezeichnet und besteht aus miteinander verwobenen Gefäßkanälen. Die zum Nidus führenden Arterien werden Feeder genannt und versorgen nur den Nidus (nicht das umliegende Gewebe) mit Blut (siehe Abbildung 1). Die Folge sind ein erhöhter Blutfluss und eine erhöhte endotheliale Wandschubspannung bei den betroffenen Gefäßen [15, Seite 194].

Diese Gefäßmissbildungen treten vor allem im Gehirn auf und können vor der Ruptur zu neurologischen Störungen und epileptischen Anfällen führen. Nach der Ruptur kommt es durch die auftretende Blutung zum Schlaganfall und dadurch zu weiteren neurologischen Störungen die bis hin zum Tod führen können [15, Seite 194].

Die Behandlung von nicht rupturierten AVMs erfolgt mithilfe von drei verschiedenen Techniken:

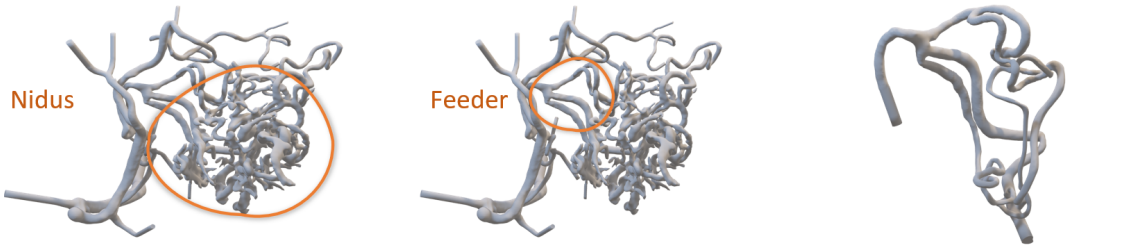
- Embolisation
- Bestrahlung
- Chirurgische Entfernung

Oft werden diese Therapieansätze miteinander kombiniert, um das Blutungsrisiko zu minimieren [15, Seite 201-209]. In diesem Paper wird ein Visualisierungstool beschrieben, das vor allem für die Embolisation einer AVM zu einem Informationsgewinn führt.

Bei AVMs wird eine Embolisation mithilfe eines sogenannten Sklerosierungsmittel durchgeführt. Dieses Mittel wird, um die Feederarterien zu verschließen, durch einen Katheter an der betreffenden Stelle des Feeders hinzugefügt. Da ein Nidus mehrere Feeder haben kann, ist die Reihenfolge der Embolisation dieser Feederarterien relevant. Durch den Verschluss einer Arterie kann der Blutfluss in den anderen Arterien so stark erhöht werden, dass eine Ruptur der AVM riskiert wird [12, Seite 1-3, Seite 106 ff.].

Um das Risiko für eine Ruptur zu minimieren, muss der behandelnde Arzt den Eingriff planen. Dazu werden möglichst viele Informationen über die Physiologie der AVM und deren Blutflusseigenschaften benötigt [15, Seite 201]. Mithilfe einer Blutflussvisualisierung, die die verschiedenen Embolisationsreihenfolgen darstellt, kann der behandelnde Arzt Aussagen über die bestmögliche Reihenfolge der Embolisation treffen und seinen Eingriff vorbereiten. Dadurch wird das Risiko einer Ruptur besser kalkulierbar.

Die Blutflussvisualsierung erfolgt in diesem Projekt aufgrund von Berechnungen und Simulationen. Es ist zwar auch möglich, den Blutfluss mit 7-Tesla-MRT-Scannern direkt



(a) AVM mit markiertem Nidus. (b) AVM mit markiertem Feeder. (c) Extrahierter Feeder des Nidus.

Abbildung 1: Der bereitgestellte Datensatz.

zu messen, die benötigte Technik ist jedoch kaum im klinischen Alltag verbreitet [16]. Dies liegt vor allem an hohen Anschaffungskosten, technischen Herausforderungen sowie mangelhafter Patientensicherheit [11, 21]. Cebral et al. [6] haben gezeigt, dass der computersimulierte Blutfluss gemessene Blutströme ausreichend genau rekonstruieren kann. Da Flusssimulationen den zusätzlichen Vorteil haben, dass sie im Gegensatz zu gemessenen Daten nicht verrauscht sind, wird in den meisten Fällen, so wie in diesem Projekt, auf Simulationen statt Messungen zurückgegriffen.

1.2 Aufgabenstellung

Ziel des Projektes ist es einen Workflow zu entwickeln, um den Blutfluss von AVM Modellen zu simulieren, in Unity zu importieren und mit Hilfe des *VFX Graphen* [23] zu visualisieren. Die Simulationen sollen die Auswirkung unterschiedlicher Varianten der Embolisation darstellen. Weiterhin sollen verschiedene Visualisierungs- und Interaktionsoptionen in einem *User Interface* bereit gestellt werden. Um dies zu erreichen werden folgende Arbeitsschritte umgesetzt:

- Erstellung verschiedener Versionen des Gefäßmodells, um unterschiedliche Variationen der Embolisation darzustellen
- Blutflusssimulation anhand aller erstellten Modelle
- Import der Simulationsdaten in den Unity VFX Graph
- Visualisierung der Simulation durch Partikel
- Interaktives User Interface
  - Möglichkeit zwischen den verschiedenen Modellen und Simulation umzuschalten
  - Basisinteraktion mit den 3D-Modellen
  - Veränderbare Visualisierungsparameter

Basierend auf der Umsetzung der vorgestellten Anwendung soll mit dieser Arbeit die wissenschaftliche Frage untersucht werden, inwieweit verschiedene Embolisationskombinationen der Feederarterien einer Arteriovenösen Malformation sich auf den Blutfluss

auswirken. Somit soll eine optimale Verschlussreihenfolge mit minimalem Rupturrisiko diskutiert werden.

1.3 Verwandte Arbeiten

Die visuelle Erkundung von Blutflussdaten nimmt durch die steigende Verfügbarkeit von gemessenen und simulierten Blutflussdaten immer weiter zu. Auf Grund der hohen Komplexität der Daten sind angemessene Visualisierungen nötig, um diese umfassend abzubilden. Häufig führt schon die Darstellung aller Strömungsvektoren im dreidimensionalen Raum zu *Visual Clutter*. Dabei werden die Strömungsdaten direkt auf verschiedene Attribute wie Glyphen oder Farbe abgebildet. Häufg werden dazu Pfeilglyphen verwendet, welche durch ihre Länge, Orientierung oder Farbe Attribute wie Flussrichtung und -geschwindigkeit darstellen [9]. Insbesondere große Mengen an Datenpunkten, die sich gegenseitig Verdecken sind schwer durch den Nutzer zu erfassen [19].

Häufig werden daher geometriebasierte Methoden zur Darstellung des Blutflusses verwendet, zum Beispiel Stromlinien oder Strömungsoberflächen. In [17] verwenden Oeltze et al. automatisch geclusterte Stromlinien um *Visual Clutter* zu reduzieren. Köhler et al. verbesserten außerdem die Darstellung animierter Stromlinien indem sie *Ambient Occlusion for Lines* auf 4D PC-MRI Daten anwenden und somit markante Strömungsstrukturen hervorheben [13]. Eine weitere Methode interessante Strömungen zu visualisieren stellen Köhler et al. in [14] vor. Dazu verwenden sie eine gefärbte Röhre um den *Flow Jet* im Kontext der anderen Stromlinien hervorzuheben. Van Pelt et al. haben Visualisierungstechniken entwickelt, die die Gefäßmorphologie mit Blutflussindikatoren vereint [22]. Auf Grund der hohen Komplexität der Daten schlagen sie einen *Details-on-Demand* Ansatz vor, bei dem verschiedene Glyphenarten kombiniert werden. Das Visualisieren der Morphologie in Kombination mit den Strömungsdaten stellt eine besondere Herausforderung dar, da die Gefäßwand den Blutstrom im Innern verdecken kann. Insbesondere wenn auf der Gefäßwand ebenfalls verschiedene Parameter visualisiert werden, beispielsweise die Wandschubspannung. In [3] entwickelten Behrendt et al. eine Kombination von Visualisierungs-, Filter- und Interaktionstechniken für die explorative Analyse des Blutflusses, wobei der Schwerpunkt auf der Beziehung zwischen lokalen Oberflächenparametern und zugrunde liegenden Flussstrukturen liegt. Gasteiger et al. [8] stellen eine Übersicht an unterschiedlichen Rendering-Optionen für Blutgefäße, im speziellen Aneurysmen, vor. Darunter zählt auch die *Ghosted View* [1], welche die Transparenz des Gefäßes anhand der Ausrichtung der Oberflächennormalen berechnet. Eine weitere Alternative um die Gefäßwand gemeinsam mit dem im Innern liegenden Blutfluss zu visualisieren, ist die *Flow Lens* [7]. Dabei wird eine virtuelle Linse über das Gefäß bewegt und die Gefäßwand an der aktuellen Stelle transparent dargestellt. Im Gegensatz dazu entwickelten Behrendt et al. in [4] eine Methode um nicht nur einen Ausschnitt sondern den gesamten Blutfluss darzustellen. Dazu nutzen sie eine Transferfunktion um

Flächen, die die im Gefäß liegenden *Path Lines* verdecken, nicht zu rendern, ähnlich zu dem Prinzip des *Frontface Culling*.

2 Workflow

Im Folgenden werden die einzelnen Schritte des Workflows (siehe Abbildung 2) sowie die verwendeten Programme und Dateiformate beschrieben.

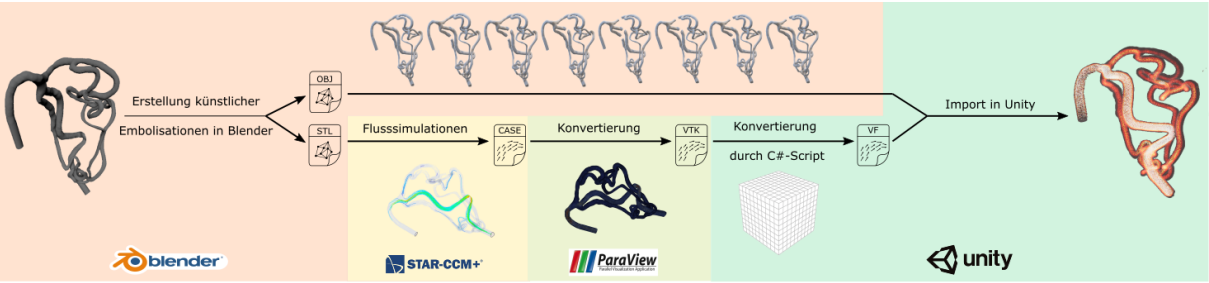


Abbildung 2: Workflow vom Ausgangsdatensatz bis hin zur fertigen Simulation.

2.1 Material

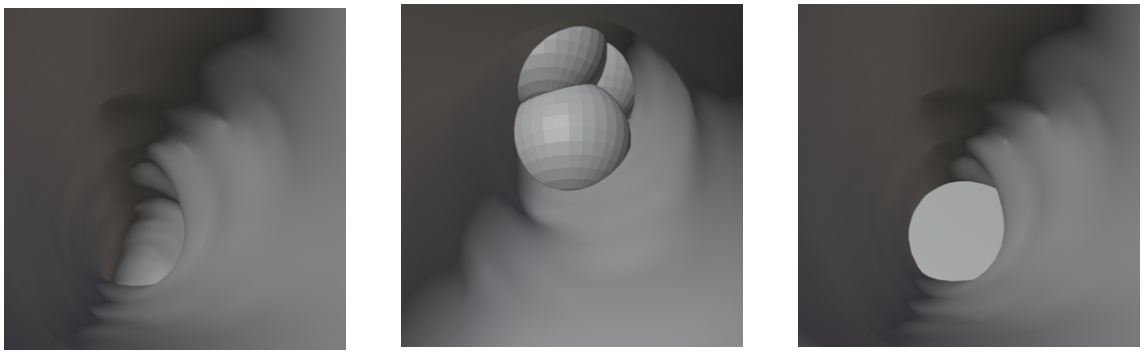
Für dieses Projekt wurde die AVM eines einzelnen Patienten verwendet. Bei den Ausgangsdaten handelt es sich um zwei OBJ Dateien (siehe Abbildung 1). Da bei der Embolisation vor allem der Feeder relevant ist, verwenden wir im Verlauf des Projekts das Modell, das nur den Feeder enthält (siehe Abbildung 1c). Der Hauptfeeder unterteilt sich in drei weiterführende Arterien. Für diese drei Arterien sollen mithilfe der Blutflussvisualisierung eine optimale Verschlussreihenfolge ermittelt werden.

2.2 Verschlusstechniken

Wie in Abschnitt 1.1 beschrieben, werden AVMs bei der Embolisation mit einem sogenannten Sklerosierungsmittel verschlossen. Um den Verschluss für die Visualisierung möglichst genau nachzuahmen, wurde diese Technik näher analysiert: Das Sklerosierungsmittel besteht meistens aus spherischen Partikeln und wird möglichst am Anfang einer Arterie hinzugefügt, um weitere Störungen im Blutfluss zu vermeiden [12, Seite 16 ff.]. Somit entsteht ein Verschluss mit einer Oberfläche aus vielen kleinen Spheren nah an der Verzweigung.

Um diesen Effekt nachzuahmen und zu überprüfen, ob dies die Simulationsergebnisse beeinflusst, wurden zwei verschiedene Verschlusstechniken gewählt:

Diese Verschlüsse wurden jeweils für das Feedermodell (Abbildung 1c) mithilfe von Blender erstellt. Bei der späteren Analyse der erstellten Blutflusssimulationen mit STAR-CCM+ war kein Unterschied zwischen den beiden Verschlusstechniken zu erkennen. Der Effekt der spherischen Oberfläche ist somit nicht relevant für die Flusssimulation. Aufgrund des geringeren Zeitaufwandes wurde die Verschlusstechnik ohne Spheren gewählt.



(a) Gefäß ohne Verschluss.

(b) Gefäß mit spherischem Verschluss.

(c) Gefäß mit glattem Verschluss.

Abbildung 3: Innenansicht von Gefäßen mit und ohne Verschluss.

Für diese Verschlusstechnik mussten 2^3 verschiedene STL-Dateien mithilfe von Blender erstellt werden (Modellübersicht in Abbildung 4). STL-Dateien sind nötig, da das im Workflow nachfolgende Programm STAR-CCM+ STL-Dateien benötigt. Da der Datensatz wie in Abschnitt 2.1 dargestellt, drei Feederarterien besitzt, waren acht Modelle (2^3) nötig, um alle möglichen Reihenfolgen (siehe Abschnitt 1.1) darzustellen.

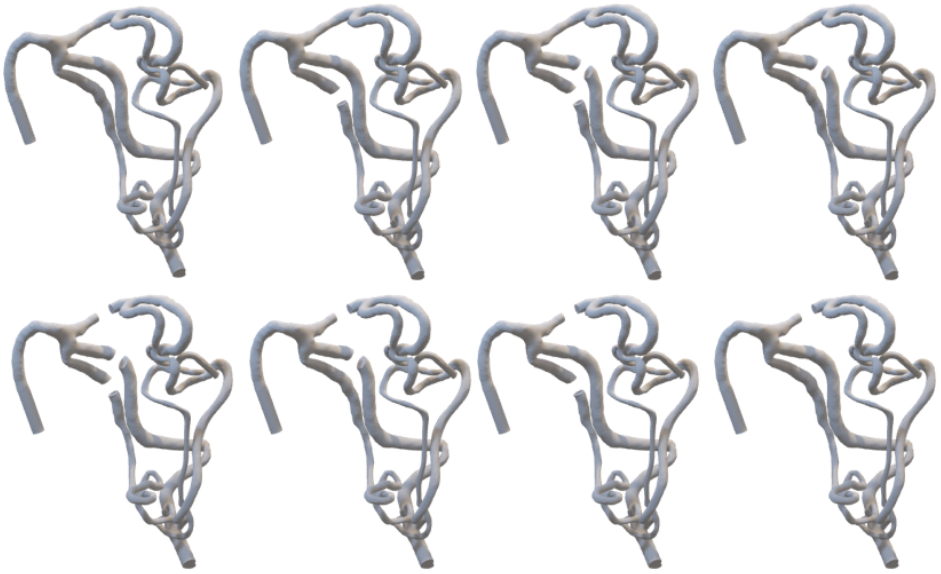


Abbildung 4: Acht Modelle mit unterschiedlichen Verschlusskombinationen.

2.3 Simulation

Die Blutflusssimulation wurde mit dem Programm STAR-CCM+ von Siemens durchgeführt. Im Rahmen des Projektes wurde eine ausführliche Präsentation erstellt, die die einzelnen Schritte sowie alle Parameter der Simulation erklärt, sodass diese unter gleichen Bedingungen für weitere Modelle wiederholt werden kann.

2.3.1 Mesh

Um den Blutfluss mit der Software STAR-CCM+ zu simulieren, werden die Oberflächenmodelle als STL-Dateien in das Programm importiert, *geremeshed* und in ein Volumenmesh umgewandelt. Außerdem werden zwei *Prism Layer* erstellt. Diese bilden dünne, fein aufgelöste Zellschichten am äußeren Rand des Gefäßes, so dass dort der Blutfluss besonders genau simuliert und Kräfte, die auf der Gefäßoberfläche wirken, berechnet werden können.

2.3.2 Flusssimulation

Für die Flusssimulation wird ein vereinfachtes Modell genutzt. Verschiedene Studien haben gezeigt, dass vereinfachte Modelle den Blutfluss trotzdem ausreichend realistisch simulieren können während sie mit deutlich geringerem Aufwand zu berechnen sind, siehe Abschnitt 1.1. In dem gewählten Modell werden die folgenden Annahmen über das Fluid getroffen:

- Stetig
- Flüssig
- Laminar
- Segregierter Fluss
- Konstante Dichte

Die Einströmgeschwindigkeit am *Inlet* beträgt $0,1m/s$ und der Druck am *Outlet* $0Pa$. Die konstante Dichte in der Simulation beträgt $1055kg/m^3$ und $0.004Pa-s$. Als Abbruchkriterium für die Simulation werden zwei Kriterien definiert. So soll die Simulation stoppen, sobald ein Kontinuitätskriterium von $1.0E-5$ erreicht ist. Wird dieses Kriterium nicht innerhalb von 800 Schritten erreicht, wird die Simulation beendet. Diese Einstellungen wurden basierend auf Literaturrecherche [18] [2] [20] sowie in Absprache mit zwei Strömungsmechanikern getroffen.

2.4 Konvertierung

Um die Simulationsdaten mithilfe des Unity VFX Graphen visualisieren zu können, müssen diese in in das Format VectorFieldFile (.vf) überführt werden [10]. Da dieses Datenformat nicht von STAR-CCM+ unterstützt wird, wurde die Konvertierung wie folgt in verschiedene Schritte eingeteilt. Die Daten werden aus STAR-CCM+ exportiert, mithilfe von ParaView konvertiert und anschließend in Unity von uns in das VectorFieldFile Format umgewandelt.

2.4.1 STAR-CCM+ Export

Anfänglich wurde versucht die Simulationsdaten direkt durch ein von STAR-CCM+ exportierbares Format in Unity einzulesen. Der Großteil der exportierbaren Formate ist binärkodiert und eine solche Implementierung hätte den geplanten Arbeitsaufwand überschritten. Daher wurde zunächst das Format .cel gewählt, welches als einziges Format die Simulationsergebnisse enthält und ASCII-kodiert ist. Nach näherer Betrachtung wurde festgestellt, dass in diesem Format nur die Informationen für *Inlet* sowie *Outlet* enthalten sind (siehe Abbildung 5). Da es also nicht möglich war ein Format zu finden, welches Direkt in Unity importiert werden kann, wurde als Zwischenschritt ParaView für die Erzeugung von ASCII-kodierten Dateien genutzt.

Abbildung 5: In .cel Format enthaltene Daten: nur an den Inlet und Outlet Stellen.

Aus STAR-CCM+ wurde das Encase Gold File Format (.case) exportiert. Für die genauen Exportparameter siehe Abbildung 6. Es ist anzumerken, dass beim Parameter *Velocity* nur die Geschwindigkeit und nicht zusätzlich die Magnitude sowie die Komponenten der Geschwindigkeit separat abgespeichert wurden (siehe Abbildung 7).

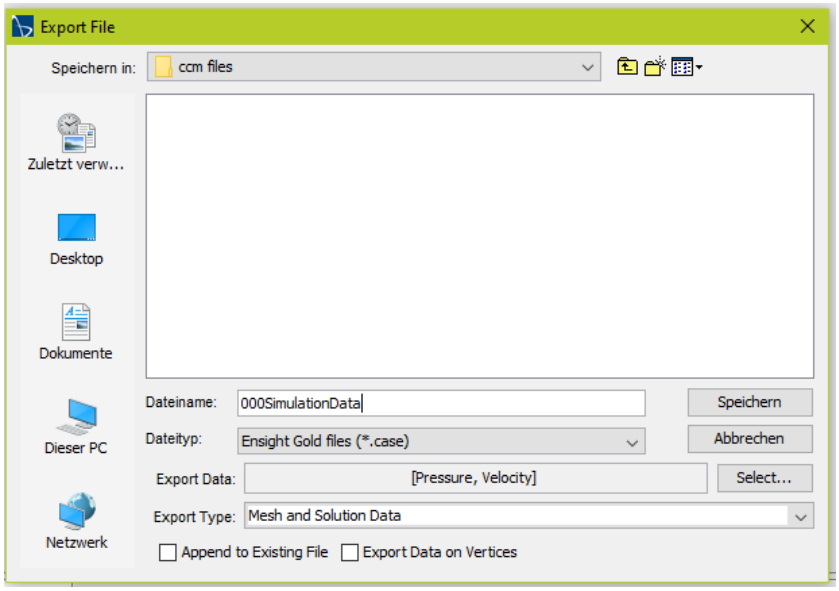


Abbildung 6: Exportparameter für STAR-CCM+.

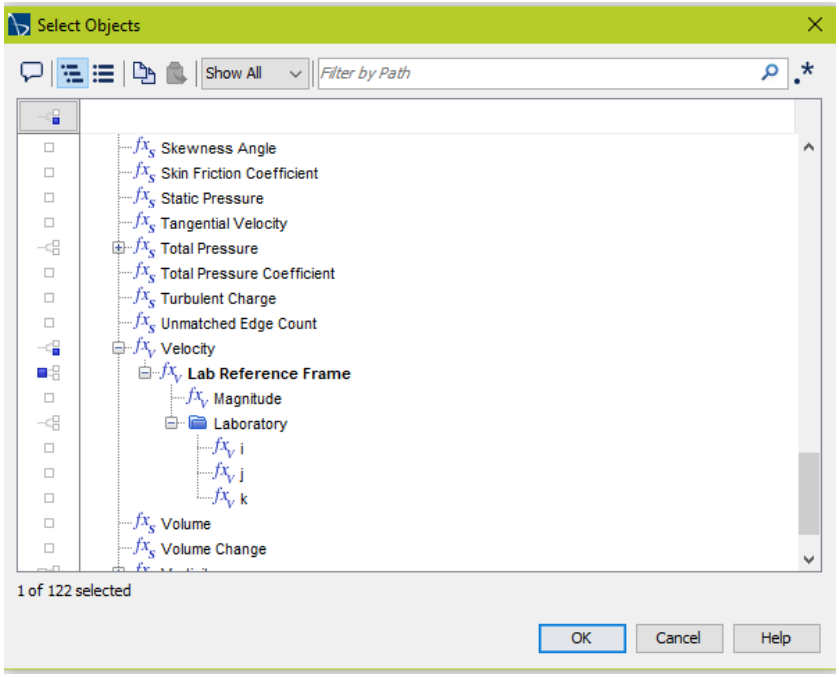


Abbildung 7: Exportoptionen für den Parameter Geschwindigkeit.

2.4.2 ParaView

Mithilfe von ParaView (Version 5.7.0) wurden die Daten in das VTK-Format übertragen. Die Konvertierung bestand aus folgenden Schritten:

1. File/ Open (Datei einlesen)
2. Filters/ Alphabetical/ Extract Block
3. Filters/ Alphabetical/ Cell Data to Point Data
4. Filters/ Alphabetical/ Merge Blocks
5. File/ Save Data (Files of Type: .vtk, Filetype: ASCII)

Zwischen den Schritten 1. bis 4. muss jeweils „Apply“, gedrückt werden.

2.4.3 Unity

In Unity (Version 2019.3.0f3) wurden die VTK Daten eingelesen, um sie letztendlich als VectorFieldFile abzuspeichern. Dafür wurden die Daten zuerst eingelesen, anschließend auf ein kartesisches Gitter projiziert und danach als VectorFieldFile abgespeichert. Für die Umwandlung in Unity werden nur zwei Parameter benötigt: der Pfad zu den VTK Daten und die Schrittweise für das Gitter, auf welches die Datenpunkte in einem späteren übertragen werden.

Zum Einlesen der Daten wird in dem angegebenen Pfad die erste gefundene Datei mit der Endung .vtk benutzt und zeilenweise eingelesen. Dabei wird von folgendem Aufbau der VTK-Dateien ausgegangen:

1. Metadaten (hier vernachlässigt)
2. Points
    - Erste Zeile enthält Anzahl und Datentyp der Punkte.
    - Anschließende Zahlen sind Positionsdaten der Punkte.
    - Drei aufeinanderfolgende Zahlen gehören zu einem Punkt, danach beginnt der nächste.
    - Durch Reihenfolge der Punkte sind implizit deren IDs bekannt.
3. Cells (hier vernachlässigt, benötigt IDs der Points)
4. CellTypes (hier vernachlässigt)
5. Pressure (eingelesen aber nicht weiter benutzt)
    - Anfangs Metadaten (Datenart SCALAR, Parametername, Datentyp)
    - Für jeden Point ein Pressure Wert
    - Reihenfolge der Druck-Werte entspricht Reihenfolge in der Points eingelesen wurden.
6. Velocity
    - Anfangs Metadaten (Datenart VECTOR, Parametername, Datentyp)
    - Drei aufeinanderfolgende Zahlen gehören zu einem Velocity Vektor, danach beginnt der nächste.
    - Reihenfolge der Druck-Werte entspricht Reihenfolge in der Points eingelesen wurden.

Nachdem die Daten eingelesen wurden liegen die Simulationsdaten als Punkte im dreidimensionalen Raum mit dazugehörigen Parametern vor. Diese müssen, um eine VectorField-Datei erzeugen zu können, in ein kartesisches Gitter überführt werden. Dafür wird als erstes das Gitter erzeugt. Die Schrittweite für die Abstände der einzelnen Gitterpunkte wurde vorher vom Nutzer in Millimeter angegeben. Nachfolgend werden die eingelesenen Punkte aus den VTK Dateien als „freie Punkte“bezeichnet, da sie nicht an ein Gitter gebunden sind.

Die Punkte werden auf das Gitter projiziert. Ein naiver Ansatz dafür ist, für jeden Gitterpunkt die gesamte Liste an freien Punkten zu durchsuchen und zu entscheiden, ob der aktuell behandelte Punkt in der Nachbarschaft des aktuellen Gitterpunktes liegt. Damit ergibt sich ein Aufwand bei g Gitterpunkten und f freien Punkten von $O = f * g$.

Daher wurden invertierte Listen verwendet, bei denen jeder Gitterpunkt eine Nachbarschaftsliste, bestehend aus freien Punkten, besitzt. Dabei wird die Liste mit den freien Punkten genau einmal durchlaufen und aus der Position des jeweiligen freien Punktes die umliegenden Gitterpunkte berechnet und in deren Nachbarschaftsliste eingetragen. Anschließend werden alle Gitterpunkte durchlaufen und aus den freien Punkten der Nachbarschaftsliste ein *Velocity* (oder *Pressure*) Wert errechnet. Damit wird der Aufwand auf $O = f + g$ gesenkt. Die Aussage über den Aufwand trifft zu, solange ein geeigneter Wert für die Schrittweite des Gitters gewählt ist, da die freien Punkte ansonsten in den Nachbarschaftslisten fast aller Gitterpunkte eingetragen werden und der Algorithmus zu dem naiven Ansatz degenerieren würde. Bei dem in dieser Arbeit genutzten Beispiel waren in den nicht-leeren Nachbarschaftslisten durchschnittlich 4.4 freie Punkte enthalten. Bisher werden für die Berechnung des Parameterwertes aus den Nachbarschaftslisten alle freien Punkte gleich gewichtet und daher der Durchschnitt aus den Parameterwerten gebildet. In Zukunft wäre es möglich die Parameterwerte mit geringerem Abstand zum Gitterpunkt höher zu gewichten, dies wurde aber durch Zeitmangel nicht umgesetzt.
Es ist anzumerken, dass die Geometrie ähnlich einer Dilatation an Volumen gewinnt. Allerdings ist diese Vergößerung so geringfügig, dass sie optisch kaum wahrnehmbar ist und daher bei der Flussanalyse keinen Einfluss hat. Davon abgesehen ist der Fluss an den Gefäßwänden, wo die Vergrösserung stattfindet, ohnehin sehr langsam. Abschließend wird das Gitter als VectorField-Datei abgespeichert. Die Struktur solcher Dateien wurde basierend auf der Beschreibung von Thomas Iché [10] umgesetzt. Die Gittergröße wurde soweit erhöht, dass jede Gitterdimension gleichlang und eine Zweierpotenz ist, wobei an den Seiten gleichmäßig mit 0 aufgefüllt wurden. Anfänglich wurde versucht anstatt des VectorField Formats das PCache Format zu nutzen, welches kein reguläres Gitter benötigt. Allerdings wurden die Partikel bei diesem Format nicht aktualisiert, sodass diese nach der anfänglichen Geschwindigkeit diese nicht verändert haben. Da aber ein ständiges Update der Partikel benötigt wird wurde dieses Format ausgeschlossen. Die Konvertierung in Unity hat je nach eingestellter Schrittweite für das Gitter 20 Sekunden bis eine Minute pro Datensatz gedauert.

2.5 Visualisierung

In Unity werden das Blutgefäß als 3D-Oberflächenmodell sowie der simulierte Blutfluss in Form von Partikeln visualisiert. Für die Exploration der Daten wird ein User Interface erstellt, welches Interaktionsmöglichkeiten mit der Visualisierung ermöglicht.

2.5.1 User Interface

Vor der Erstellung des User Interfaces wurden folgende drei Interaktionsarten für den Nutzer festgelegt:

1. die Rotation, Skalierung, Translation und das Zurücksetzen des Modells

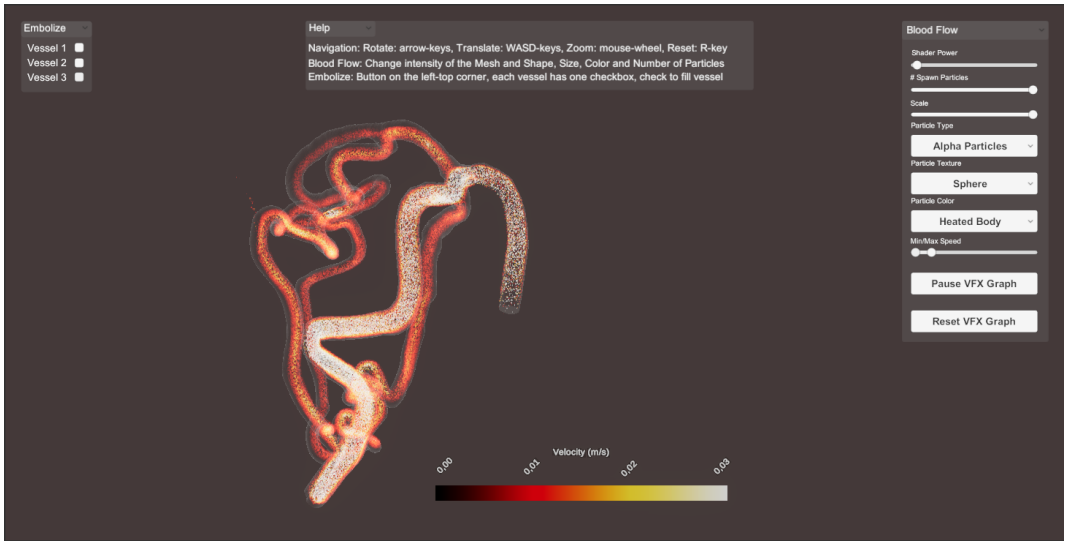


Abbildung 8: User Interface des Visualisierugstools

2. das Wechseln zwischen den Embolisationen der einzelnen Feederarterien (2^3 verschiedene Möglichkeiten)
3. die Einstellung des Shaders und der Partikel

Ein selbst gewähltes Ziel der Interaktion war es, dem Nutzer die simultane Navigation und Anpassung der Partikel beziehungsweise des Mesh-Shaders zu ermöglichen. Deshalb wurde für die Navigation die Interaktion mithilfe der Tastatur anstatt der Maus gewählt. Dadurch ist die Maus frei für die Interaktion mit den einzelnen Fenstern des User Interfaces. Das User Interface (siehe Abbildung 8) besteht aus vier verschiedenen Teilen:

1. dem Bildschirm mit dem AVM-Modell und dem zugehörigen Blutfluss
2. dem „Embolisations“-Fenster
3. dem „Shader & Partikel“-Fenster
4. dem „Hilfe“-Fenster

Im „Embolisations“-Fenster (Abbildung 9) kann der Nutzer mithilfe des aus- oder abwählens von drei Checkboxen entscheiden, welche Feederarterie verschlossen werden soll. Das

Abbildung 9: „Embolisations-Fenster“

„Shader & Partikel“-Fenster (Abbildung 10) ermöglicht unter anderem die Einstellung wie stark das AVM-Modell gerendert werden soll. Die Einstellung erfolgt mithilfe eines Sliders, auf dessen niedrigster Stufe das Modell durchsichtig ist (d.h. nur der Blutfluss ist zu

sehen). Dadurch kann der Nutzer seinen Fokus zwischen dem Modell und den Partikeln des Blutflusses variieren. Des Weiteren kann der Nutzer die Größe und Anzahl der zu erstellenden Partikel des Blutflusses einstellen. Änderbar sind auch die Textur der Partikel und wie eine Überlappung von Partikeln dargestellt werden soll. Außerdem kann der Nutzer als farbliche Darstellung der Partikel entweder die Farbe Rot, die Regenbogen-Skala oder die Heated-Body-Skala auswählen. Die Umsetzung dieser Einstellungsmöglichkeiten des Shaders und der Partikel werden im nachfolgenden Absatz näher beschrieben.

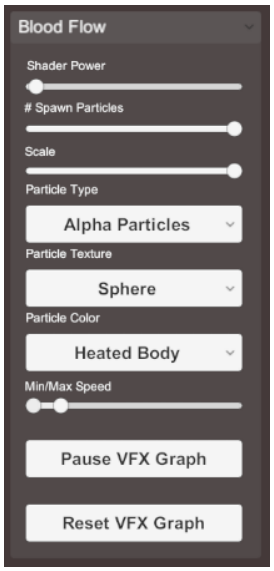


Abbildung 10: „Shader & Partikel“-Fenster

Um dem User seine Möglichkeiten in der Interaktion kurz und leicht zugänglich darzustellen, wurde ein „Hilfe“-Fenster (Abbildung 11) hinzugefügt. In diesem Fenster werden dem Nutzer die zuvor genannten Fenster kurz erklärt und die entsprechenden Tasten für die Skalierung, Rotation, Translation und das Zurücksetzen der Transformation dargestellt.

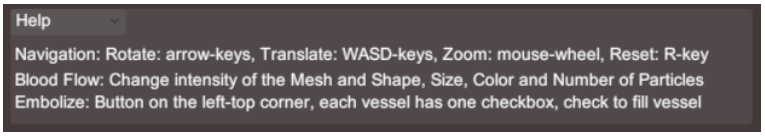


Abbildung 11: Hilfe-Fenster

Außerdem sind alle drei beschriebenen Fenster einklappbar. Dadurch kann der Nutzer individuell entscheiden, wie viel Platz er für die Darstellung des Modells braucht, und wo sein Fokus liegen soll.

2.5.2 Visualisierung von Blutfluss und Gefäßen

In Unity werden das Oberflächenmodell des Gefäßes sowie der Blutfluss visualisiert. Damit der Blutfluss im Gefäßinnern gut zu erkennen ist, wird das Gefäßmodell teilweise transparent dargestellt. Dazu wird das Prinzip der *Ghosted View* genutzt [1]. Die

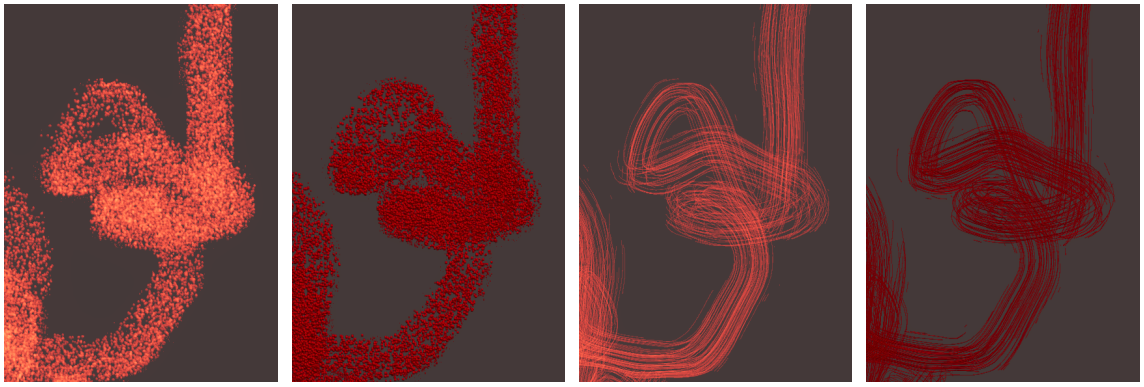
(a) Shader mit Wert 0,05. (b) Shader mit Wert 1. (c) Shader mit Wert 5.

Abbildung 12: Der Surface Shader mit verschiedenen, durch den Nutzer eingestellten Faktoren.

Transparenz wird anhand der Ausrichtung der Oberflächennormalen des Gefäßmodells berechnet. Flächen, deren Normale orthogonal zur Kameraebene ist, werden komplett transparent, Oberflächen mit Normalen parallel zur Kameraebene opaque angezeigt. Der Grad der Transparenz für die Normalenausrichtungen dazwischen wird interpoliert. Umgesetzt wurde der Shader mit dem *Unity Shader Graph*. Die Stärke des Shaders kann im *User Interface* angepasst werden (siehe Abbildung 12). Dazu wird das Skalarprodukt von Normalenvektor und Kameraebene mit dem von Nutzer gewählten Wert potenziert. Die Transparenz α des Modells an einer Stelle wird dementsprechend folgendermaßen ermittelt: $\alpha = min(max(0, 1 - (normalize(CameraPlane) \cdot SurfaceNormal)^x, 1)$, wobei x der in der UI eingestellte Wert ist, um die Stärke des Shaders zu variieren.

Der Blutfluss wird durch Partikel dargestellt. Dazu wird der *Unity VFX Graph* (siehe Abbildung 17, Abbildung 18 und Abbildung 19) genutzt. Der VFX Graph simuliert das Partikelverhalten auf der GPU, wodurch deutlich mehr Partikel simuliert werden können, als mit dem normalen Unity Partikelsystem. Der VFX Graph ist seit Version 2018.3 in Unity integriert. Es ist jedoch ratsam, eine Version ab 2019.3 zu nutzen, da der VFX Graph in den vorherigen Versionen nur als Prototyp implementiert war. Um Vektorfelder in dem VFX Graphen verwenden zu können, müssen diese im VectorFieldFile Format vorliegen, siehe Abschnitt 2.4.3.

Der VFX Graph besteht aus den vier Hauptkomponenten *Spawn*, *Initialize*, *Update* und *Output*. In *Spawn* (siehe Abbildung 17) wird festgelegt, wie viele Partikel pro Sekunde erstellt werden sollen. In den für das Projekt erstellten Graphen, wurde eine Obergrenze von 1000000 Partikeln pro Sekunde festgelegt. Dieser Wert wurde durch das Probieren verschiedener Einstellungen ermittelt. Wie viele Partikel tatsächlich erstellt werden sollen, kann in der UI vom Nutzer eingestellt werden, siehe Abschnitt 2.5.1. In der Komponente *Initialize* (siehe Abbildung 17) kann die maximale Anzahl an Partikeln, die gleichzeitig in dem Projekt existieren dürfen, festgelegt werden. Diese beträgt in



(a) Particle Quads mit Additivem Blemding.

(b) Particle Quads mit Alpha Blending.

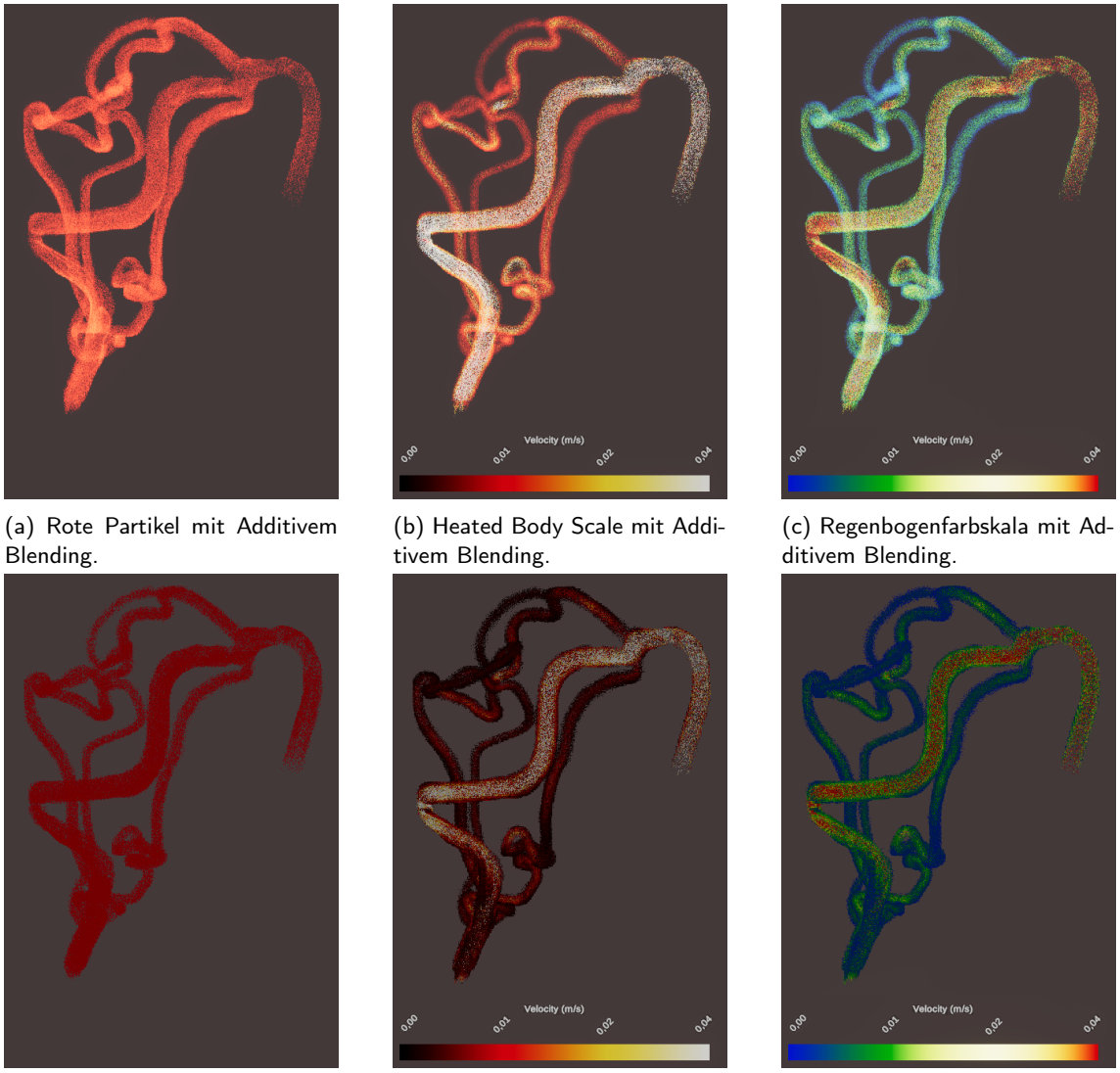
(c) Particle Strips mit Additivem Blending.

(d) Particle Strips mit Alpha Blending.

Abbildung 13: Ergebnisse der vier unterschiedlichen VFX Graphen.

den vorliegenden Graphen 2000000 Partikel. Außerdem kann die Lebenszeit der Partikel angegeben werden, diese wird für jedes Partikel auf eine zufällige Zahl zwischen 8 und 80 gesetzt. Die Größe der Partikel liegt bei 0,3. Die Werte wurden durch ausprobieren ermittelt. Die Partikel werden in einem würfelförmigen Volumen mit Seitenlänge 1000 und Mittelpunkt 0 erstellt. In der *Update*-Komponente (siehe Abbildung 18) wird die Größe des Vektorfeldes auf die gleichen Werte gesetzt. Die Intensität des Vektorfeldes ist 1, um den simulierten Blutfluss nicht zu verfälschen. Der Drag wurde auf eine Hohe Zahl gestellt, in diesem Falle 99, um zu verhindern, dass Partikel aus dem Vektorfeld ausbrechen. Da die Partikel auf allen möglichen Positionen in dem Spawn-Würfel erstellt werden, werden alle Partikel mit Geschwindigkeit 0 auch auf eine Größe von 0 gesetzt. Somit werden nur noch Partikel angezeigt, die sich im Gefäß bewegen. Auch unsichtbare Partikel, würden in die maximale Kapazität von 2000000 Partikeln zählen und so die Anzahl an Partikeln, die im Gefäß erstellt werden verringern. Um dies zu vermeiden, wird in jedem Update Schritt eine negative Lebenszeit auf die Partikel mit Geschwindigkeit 0 addiert, sodass diese kurz nach ihrer Erstellung wieder verschwinden. In der Komponente *Output* (siehe Abbildung 19) werden viele Anzeigeoptionen für die Partikel gesetzt, unter anderem die Partikeltypen (z.B. Quads), der Blend Mode (z.B. Additive oder Alpha) und auch die Farbe der Partikel. Letztere wird in den verwendeten VFX Graphen basierend auf der Geschwindigkeit des Partikels gesetzt.

Insgesamt werden vier VFX Graphen erstellt, mit denen der Blutfluss unterschiedlich visualisiert werden kann (siehe Abbildung 13). Zwei der Graphen haben als Output *Particle Quads*, die zwei anderen Graphen erzeugen *Particle Strips*. *Particle Quads* visualisieren viele kleine einzelne Partikel, basierend auf einer zur Kamera orientierten Textur, die in den VFX Graphen geladen wird. *Particle Strips* nutzen diese Textur um daraus Stromlinien entlang des Vektorfeldes zu zeichnen. Pro Output gibt es einen Graphen der *Additives Blending* sowie einen, der *Alpha Blending* verwendet. Beim *Alpha Blending* verdecken sich opaque Partikel gegenseitig (siehe Abbildung 13b und Abbildung 13d), im Falle von *additivem Blending* addieren sich die Farbwerte der übereinanderliegenden



(a) Rote Partikel mit Additivem Blending.

(b) Heated Body Scale mit Additivem Blending.

(c) Regenbogenfarbskala mit Additivem Blending.

(d) Rote Partikel mit Alpha Blending.

(e) Heated Body Scale mit Alpha Blending.

(f) Regenbogenfarbskala mit Alpha Blending.

Abbildung 14: Die drei unterschiedlichen Färbungen der Partikel mit Additivem und Alpha Blending.

Partikel auf, sodass Regionen, an denen sich viele Partikel sammeln, hell leuchten (siehe Abbildung 13a und Abbildung 13c). Somit bieten die Partikel mit *Additivem Blending* dem Nutzer eine bessere Tiefenwahrnehmung, da durch die unterschiedlichen Helligkeitsintensitäten erkennbar ist, an welchen Stellen Gefäße hintereinander liegen. Die *Particle Strips* heben zusätzlich hervor, in welche Richtung der Blutstrom fließt. So kann der Nutzer selbst in engen, sich überlappenden Windungen des Blutgefäßes nachvollziehen, wie der Gefäßverlauf aussieht, während die *Particle Quads* nur eine Fläche an den Partikeln ohne sichtbare Orientierung darstellen.

Um die Geschwindigkeit des Blutflusses darzustellen, werden verschiedene Farbskalen verwendet, zwischen denen der Nutzer wählen kann (siehe Abbildung 14). Die Regenbogenfarbskala ist in Strömungssimulationssoftware, so auch in *Star-CCM+*, weit verbreitet. Daher wurde diese auch in das User Interface Integriert. Auf Grund der Kritik an dieser Farbskala wurde auch eine weitere Skala, die *Heated Body Scale* verwendet [5]. Durch

das *Additive Blending* werden die Farben der Partikel verändert, so sehen Stellen, an denen sich viele ursprünglich rote Partikel verlagern für den Nutzer orange bis gelb aus (siehe Abbildung 14a). Dadurch sind auch die Farbskalen nicht mehr korrekt dargestellt. Insbesondere die *Heated Body Scale* verändert sich deutlich, zu sehen in Abbildung 14b und Abbildung 14e. Im *Additiven Blendmode* werden die schwarzen Partikel gar nicht angezeigt, wodurch es den Anschein hat, dass es keine Partikel mit dieser Geschwindigkeit gibt.

Da das Vektorfeld auf einer geremeshten Version des Gefäßmodells erstellt und mehrmals konvertiert wurde, liegt es nach dem Import in Unity nicht perfekt in dem originalen Gefäßmodell. Skalierung sowie Rotation können aus den Daten ermittelt werden. Die Position muss jedoch per Hand und durch Augenmaß angepasst werden. Auch danach liegt das Vektorfeld noch nicht perfekt im Gefäßinnern. Neben den durch externe Programme durchgeführten Schritten des Remeshens für die Simulation sowie der Konvertierung in das Datenformat VTK, könnten Ungenauigkeiten auch durch das selbst geschriebene C#-Script entstehen, welches die VTK Daten parsed und daraus die für den VFX Graphen benötigten VF Dateien erstellt.

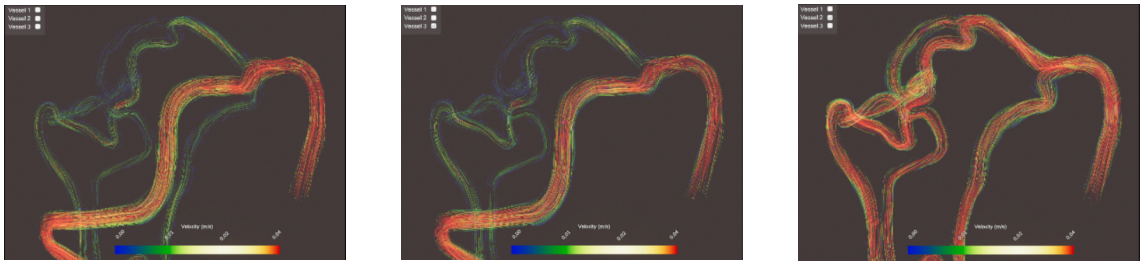
3 Zusammenfassung

In dieser Arbeit wurden die einzelnen Schritte, die für den Workflow aus Abbildung 2 nötig waren, erklärt. Dieser Workflow wurde erarbeitet um folgende wissenschaftliche Frage zu beantworten: Wie wirkt sich der Verschluss (Embolisation) von *Feederarterien* auf den Blutfluss aus. In den nachfolgenden Abschnitten werden die einzelnen Schritte dieses Workflows diskutiert und Anregungen für mögliche Weiterentwicklungen gegeben.

3.1 Diskussion

Im ersten Schritt, der Modellerstellung, wurde darauf geachtet, die Modelle möglichst realitätsnah zu gestalten. Es wurde recherchiert an welcher Stelle und mit welchem Material AVMs embolisiert werden. Da zwei verschiedene Verschlusstechniken mit STAR-CCM+ visualisiert wurden und keine Unterschiede zwischen den beiden Simulationen erkennbar waren, wurde gezeigt, dass die später weiter verwendete Verschlusstechnik für die Simulation in STAR-CCM+ ausreichend ist.

Ein großer Teil des Arbeitsaufwandes wurde in die Suche nach den passenden Datenformaten investiert. Obwohl das .cel Format nur kurz im Abschnitt 2.4.1 erwähnt wurde, hat es viel Zeit in Anspruch genommen, dessen Daten einzulesen und als unbrauchbar zu erkennen. Ähnlich war es mit dem pCache Format (es wurde zuerst favorisiert da es kein Gitter benötigt), bis erkannt wurde dass sich die Partikelupdates nur mit dem Vector-Field Format lösen lassen. Daher liegt ein großer Mehrwert dieses Projektes darin, dass ein Workflow mitsamt der nötigen Exportschritte und Datenformate vorliegt. Eventuelle



(a) Keine Verschlüsse. (b) Verschluss: unteres Gefäß. (c) Verschluss: mittleres Gefäß.

Abbildung 15: Blutflusssimulation mit unterschiedlichen Verschlüssen.

Änderungen wie das Auslassen von Paraview oder die Nutzung anderer Datenformate können nachträglich mit diesem Projekt als Vergleich erforscht werden.

Die wissenschaftliche Fragestellung dieser Arbeit, die unter Abschnitt 1.2 erläutert wurde, lautet: wie wirkt sich der Verschluss (die Embolisation) der Feederarterien auf den Blutfluss aus. In Abbildung 15 sind verschiedene Blutflusssimulationen dargestellt. In Abbildung 15a ist keine der drei abzweigenden Gefäße verschlossen. Hier kann man erkennen, dass das untere Gefäß am wenigsten und das mittlere Gefäß am stärksten durchblutet werden. Wenn das untere Gefäß verschlossen wird (Abbildung 15b), ändert sich wenig an der Blutflusssimulation des oberen und mittleren Gefäßes. Wenn stattdessen das mittlere Gefäß verschlossen wird (Abbildung 15c), ist eine starke Zunahme des Blutflusses für das obere und untere Gefäß zu erkennen.

Somit kann man durch die erstellte Visualisierung erkennen, wie sich der Verschluss der Feederarterien auf den Blutfluss auswirkt. Eine erste Auswertung der Blutflusssimulation ergibt, dass die Gefäße in folgender Reihenfolge embolisiert werden sollten: das untere, das obere und zum Schluss das mittlere Gefäß. Allerdings ist diese Reihenfolge anfechtbar, wenn im mittleren Gefäß zum Beispiel ein ungünstige Wandschubspannung herrscht, einige Wände sehr dünn sind oder/und der Blutdruck nach der Embolisation zu hoch wäre. Erweiterungen die diese Information zur Verfügung stellen, könnten das erstellte Programm verbessern und werden deshalb in Abschnitt 3.2 näher erläutert.

3.2 Ausblick

Die Visualisierung des Blutflusses kann um drei zusätzliche Informationen erweitert werden, damit die Risikoabwägung für die Embolisation verbessert werden kann.

Wie in Abbildung 16 dargestellt, könnte man dem Nutzer noch die Gefäßwände der AVM hervorheben, die im Vergleich zu den anderen Wänden besonders dünn sind. Wenn eine dünne Wand viel Blutfluss tragen muss, ist das Rupturrisiko höher einzuschätzen.

Zwei weitere Informationen sind der Blutdruck und die sogenannte Wandschubspannung. Der Blutdruck beeinflusst mit welcher Kraft der Blutfluss gegen die Gefäßwand prallt. Die Wandschubspannung wird durch die Geschwindigkeit des Blutes beeinflusst und gibt Aufschluss darüber wie viel Druck auf eine Gefäßwand ausgeübt wird. Durch

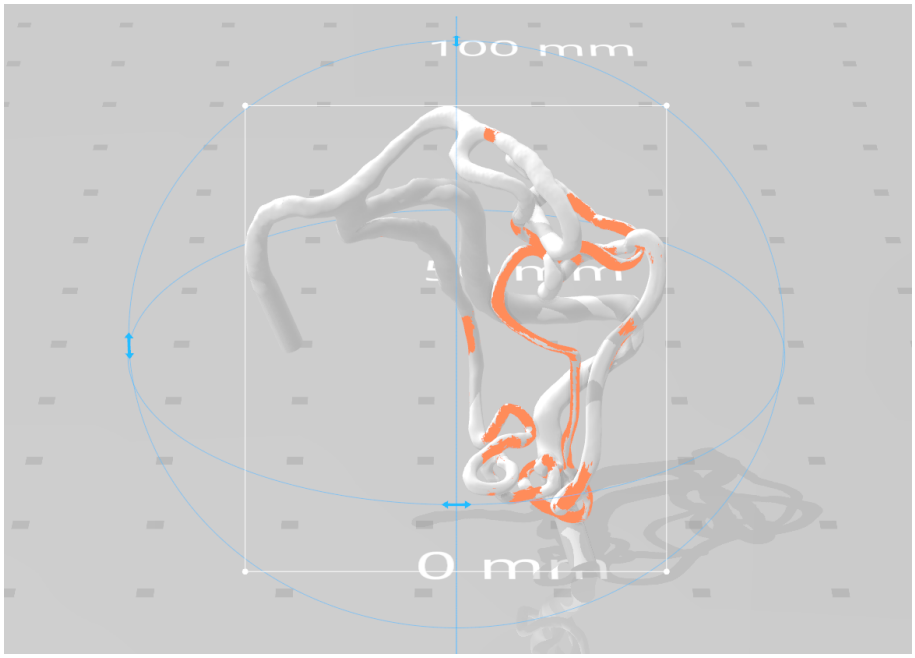


Abbildung 16: Hervorgehobene dünne Gefäßwandabschnitte des AVM Modells

diese drei Informationen kann eine Embolisation in Hinsicht auf das Rupturrisiko besser geplant werden.

Da ein solches Tool auch für die Schulung von Medzinstudenten und unerfahrenen Ärzten dienen kann, könnte das Programm in eine VR-Anwendug konvertiert werden. Dadurch könnte eine Embolisation als OP für den Nutzer simuliert werden. Dafür wäre eine größere Auswahl an verschiedenen AVM-Modellen relevant.

Außerdem können bisher keine eigenen Modelle in das Programm geladen werden. Die im Workflow beschriebenen Schritte müssten automatisiert werden, damit das Programm auf neue Modelle reagieren kann.

Literatur

[1] Alexandra Baer, Rocco Gasteiger, Douglas Cunningham, and Bernhard Preim. Perceptual Evaluation of Ghosted View Techniques for the Exploration of Vascular Structures and Embedded Flow. *Computer Graphics Forum*, 2011.

[2] P.D. Ballyk, D.A. Steinman, and C.R. Ethier. Simulation of non-newtonian blood flow in an end-to-side anastomosis. *Biorheology*, 31(5):565–586, October 1994.

[3] B. Behrendt, P. Berg, O. Beuing, B. Preim, and S. Saalfeld. Explorative blood flow visualization using dynamic line filtering based on surface features. *Computer Graphics Forum*, 37(3):183–194, June 2018.

[4] B. Behrendt, B. Köhler, U. Preim, and B. Preim. Enhancing visibility of blood flow in volume rendered cardiac 4d PC-MRI data. In *Informatik aktuell*, pages 188–193. Springer Berlin Heidelberg, 2016.

[5] David Borland and Russell Taylor Ii. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications*, 27(2):14–17, March 2007.

[6] Juan R. Cebral, Richard S. Pergolizzi, and Christopher M. Putman. Computational fluid dynamics modeling of intracranial aneurysms:. *Academic Radiology*, 14(7):804–813, July 2007.

[7] R. Gasteiger, M. Neugebauer, O. Beuing, and B. Preim. The FLOWLENS: A focus-and-context visualization approach for exploration of blood flow in cerebral aneurysms. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2183–2192, December 2011.

[8] R. Gasteiger, M. Neugebauer, C. Kubisch, and B. Preim. Adapted surface visualization of cerebral aneurysms with embedded blood flow information, 2010.

[9] Rocco Gasteiger. *Visual Exploration of Cardiovascular Hemodynamics*. PhD thesis, Otto von Guericke University, February 2014.

[10] Thomas Iché. Vectorfieldfile data format. `https://github.com/peeweek/VectorFieldFile`.

[11] Muhammad Irfan Karamat, Sahar Darvish-Molla, and Alejandro Santos-Diaz. Opportunities and challenges of 7 tesla magnetic resonance imaging: A review. *Critical Reviews in Biomedical Engineering*, 44(1-02):73–89, 2016.

[12] Srilakshmi Adhyapak Kiron Varghese. *Therapeutic Embolization*. Springer, February 2017.

[13] B. Köhler, M. Grothoff, M. Gutberlet, and B. Preim. Visualization of cardiac blood flow using anisotropic ambient occlusion for lines. In *Proceedings of the Conference on Vision, Modeling and Visualization*, VMV '17, page 29–36. Eurographics Association, 2017.

[14] B. Köhler, M. Grothoff, M. Gutberlet, and B. Preim. Visual and quantitative analysis of great arteries' blood flow jets in cardiac 4d PC-MRI data. *Computer Graphics Forum*, 37(3):195–204, June 2018.

[15] Thomas Kretschmer. *Zerebrale Aneurysmen und Gefäßmalformationen, Behandlungsgrundlagen und neurochirugische Therapie in Fallbeispielen*. Springer, 2017.

[16] Jing Liu, Louise Koskas, Farshid Faraji, Evan Kao, Yan Wang, Henrik Haraldsson, Sarah Kefayati, Chengcheng Zhu, Sinyeob Ahn, Gerhard Laub, and David Saloner. Highly accelerated intracranial 4d flow MRI: evaluation of healthy volunteers and patients with intracranial aneurysms. *Magnetic Resonance Materials in Physics, Biology and Medicine*, 31(2):295–307, August 2017.

[17] S. Oeltze, D. J. Lehmann, A. Kuhn, G. Janiga, H. Theisel, and B. Preim. Blood flow clustering and applications in virtual stenting of intracranial aneurysms. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):686–701, May 2014.

[18] K. Perktold, R. Peter, and M. Resch. Pulsatile non-newtonian blood flow simulation through a bifurcation with an aneurysm. *Biorheology*, 26(6):1011–1030, December 1989.

[19] B. Preim and C. P. Botha. Acquisition of medical image data. In *Visual Computing for Medicine: Theory, Algorithms, and Applications*, chapter 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2013.

[20] Luisa Costa Sousa, Catarina F. Castro, Carlos Conceição António, and Rui Chaves. Blood flow simulation and vascular reconstruction. *Journal of Biomechanics*, 45(15):2549–2555, October 2012.

[21] Anja G. van der Kolk, Jeroen Hendrikse, Jaco J.M. Zwanenburg, Fredy Visser, and Peter R. Luijten. Clinical applications of 7t MRI in the brain. *European Journal of Radiology*, 82(5):708–718, May 2013.

[22] R. van Pelt, R. Gasteiger, K. Lawonn, M. Meuschke, and B. Preim. Comparative blood flow visualization for cerebral aneurysm treatment assessment. *Computer Graphics Forum*, 33(3):131–140, June 2014.

[23] Yitian Zhang. Using the unity game engine as a platform for prototyping cinematic visual effects. Master's thesis, Aalto University, 2020.

Anhang

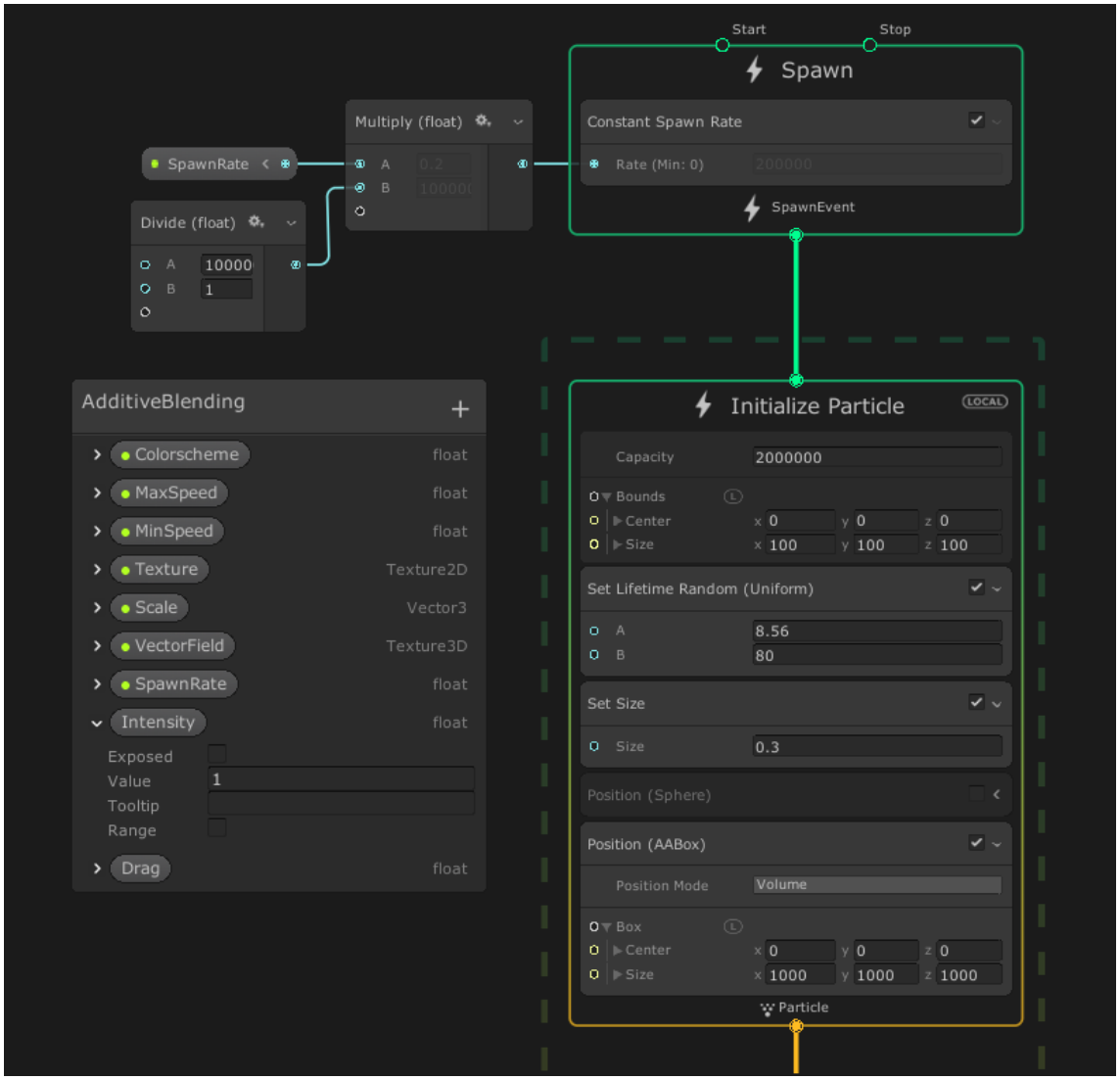


Abbildung 17: Die Komponenten *Spawn* und *Initialize* des VFX Graphen für Particle Quads mit Additivem Blending. Auf der linken Seite sind die Variablen des Graphen zu sehen.

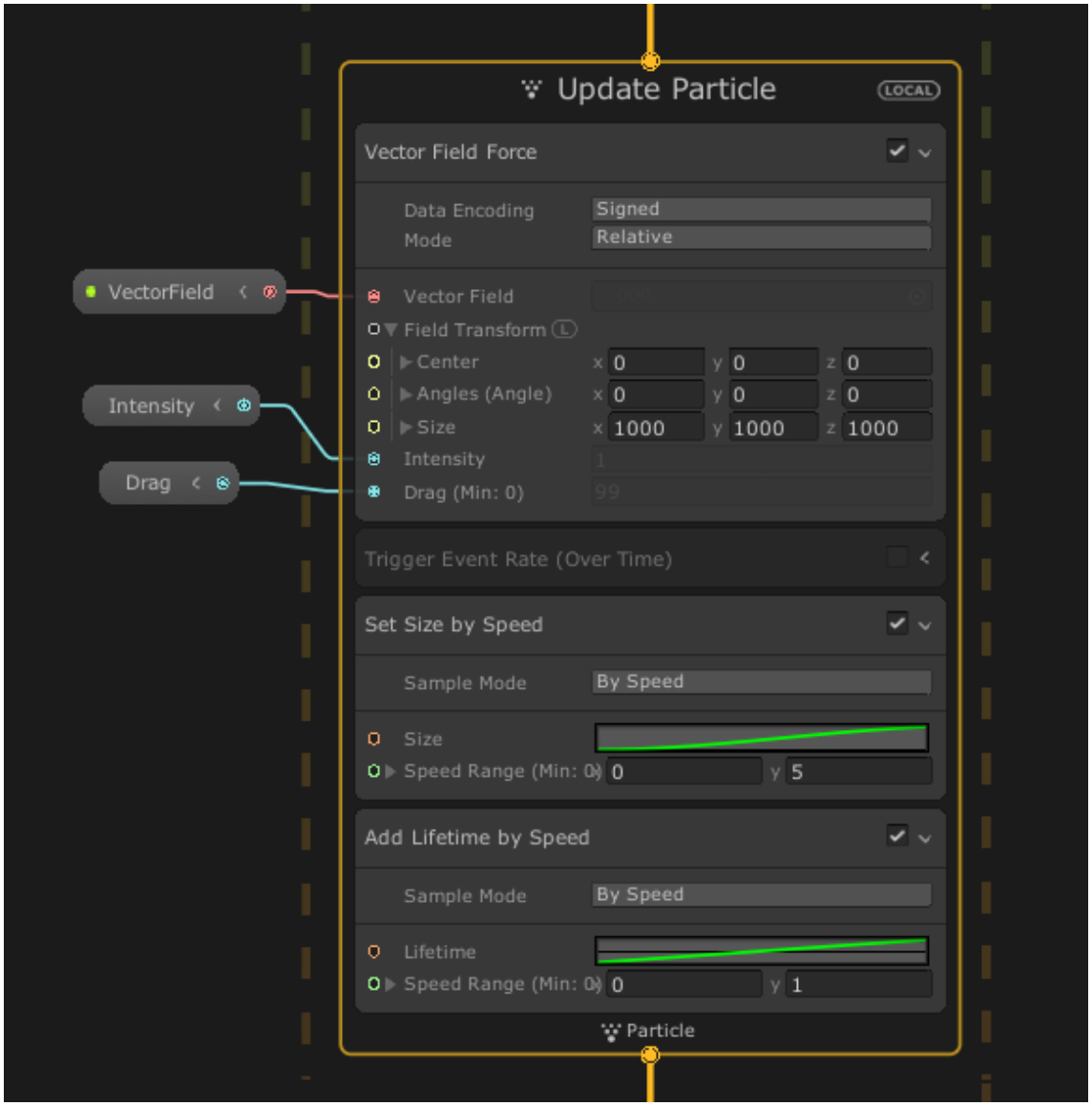


Abbildung 18: Die Komponente *Update* des VFX Graphen für Particle Quads mit Alpha Blending.

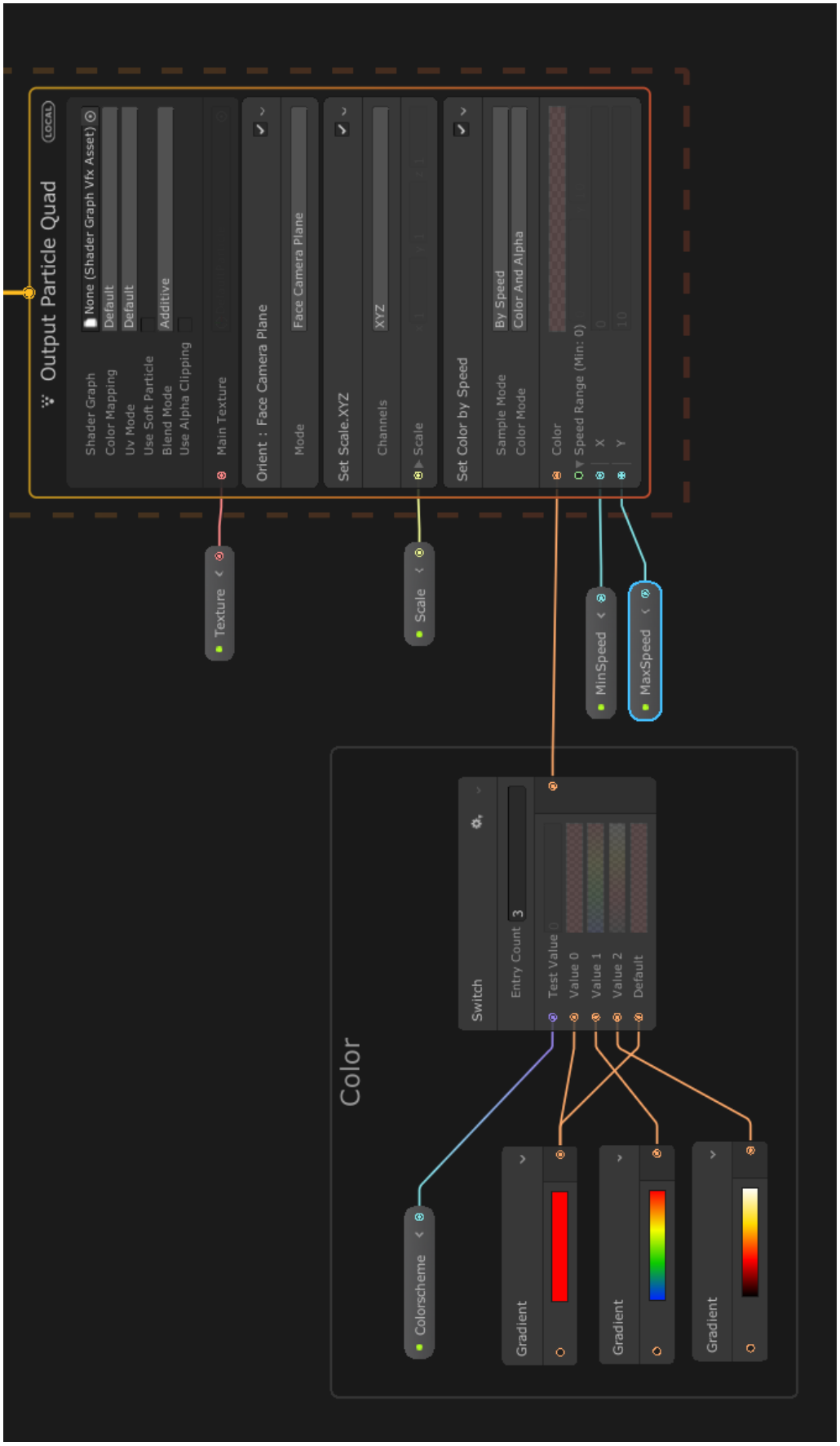


Abbildung 19: Die Komponente *Output* des VFX Graphen für Particle Quads mit Alpha Blending.