TECHNISCHE HOCHSCHULE
OSTWESTFALEN-LIPPE
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS

Fachbereich Medienproduktion

Sommersemester 2022

Abgabedatum: 18.08.2022

Möglichkeiten zur Simulationsbeschleunigung unter Berücksichtigung der optischen Qualität

---

Bachelorarbeit

Patrick Schmidtmeyer

Prüfer

Prof. Dipl.-Ing. Rico Dober

Zweitprüfer

Prof. Dr. Guido Falkemeier

Inhaltsverzeichnis

1. Vorwort

Im Laufe der letzten Jahre ist die Taktung neuer Kinofilme und Serien deutlich angestiegen. Zudem werden diese immer aufwändiger produziert und bringen einen höheren Ertrag. So haben zum Beispiel die Filmreihen des Marvel Universums weltweit bis 2022 25,16 Milliarden US-Dollar eingenommen (Statista, 2022). Keine der Filme oder Serien kam ohne visuelle Effekte aus. Zudem werden die verwendeten Effekte komplexer. Ein anderes Beispiel für die Komplexität in Kinofilmen ist der Film „Tennet“ aus dem Jahr 2020. Hier wird unter anderem das erstellte Filmmaterial vorwärts und rückwärts zeitgleich abgespielt. Dieser Film gewann eine Auszeichnung bei der Oscar Verleihung im Jahr 2021 für die besten visuellen Effekte. Diese Beispiele zeigen, dass an der visuellen Effekt-Branche ein stets wachsendes Interesse besteht.

1.1 Motivation

Der Grund für die Auswahl des Themas ist mein großes Interesse an der Marvel Filmreihe. In meiner Jugendzeit war ich von den Effekten und den Simulationen immer sehr beeindruckt, ohne es großartig zu hinterfragen. Im Verlauf des Studiums bin ich etwas vertrauter mit der Materie geworden und mein Interesse an der Effekt-Simulation hat sich mehr und mehr gesteigert.

Als in der Coronapandemie die Streaming-Plattform Disney+ herauskam und die Rechte an allen Marvel Filmen hatte, hat es nicht lang gedauert, bis ich alle Filme aus meiner Kindheit und Jugend erneut angeschaut habe. Nur diesmal sind mir viele Einstellungen aufgefallen, die ich gedanklich versucht habe nachzustellen, beziehungsweise überlegt habe, wie genau diese Szenen in diversen 3D-Programmen erstellt worden sind.

Je mehr ich mich damit befasste und je mehr ich mir über die Vorgehensweise Gedanken gemacht habe, desto größer war meine Frage über den betriebenen Zeitaufwand und wie lange es wohl dauern würde, diese Art von Simulationen selbst anzufertigen. Somit kam ich schließlich auf das Thema der Simulationsbeschleunigung bezogen auf Feuer Simulationen.

1.2 Ziel der Arbeit

Das Ziel dieser Bachelorarbeit ist es, die verschiedenen Simulationstypen aufzuzeigen und herauszufinden, ob und welche Möglichkeiten es gibt, die Simulationszeit unter Berücksichtigung der optischen Qualität zu beschleunigen. Um dieses Vorgehen zu belegen, wurde ein Interview auf Englisch erstellt. Dieses richtet sich an Personen, die aktuell in der Industrie bei verschiedenen VFX-Firmen angestellt sind. Die Ergebnisse der Interviews fließen dann in die Analyse der verschiedenen Einsatzgebiete ein und sind Grundlage für den praktischen Teil der Arbeit. In diesem wird ein Anwendungsbeispiel erstellt und eine hypothetische Hochrechnung der Simulationszeit durchgeführt. Hierbei werden Möglichkeiten erläutert, um die Simulationszeit einzusparen.

1.3 Vorgehensweise

Der theoretische Teil der Arbeit beginnt mit einer eigenen Definition der verschiedenen Simulationen. Darauf folgt das Kapitel zu der Fluid Gas Simulation. Dieses Kapitel beinhaltet die Funktionsweise, welche anhand verschiedener Begriffe erklärt werden. Dazu zählen die Begriffe Voxel, Solver, Regeln, Emitter und Kräfte. Abgeschlossen wird das Kapitel von einer Gegenüberstellung und Erläuterung unterschiedlicher Kameraeinstellungen.

Anschließend werden die Einsatzgebiete der verschiedenen Simulationstypen erklärt und verglichen. Diese sind *Bespoke Simulationen, Pre-cached Simulationen, das Upresing Verfahren* und *vorgefertigte 2D Elemente*.

Darauf folgt die Beschreibung eines Praxisprojekts, das von mir angefertigt wurde. In diesem Kapitel wird auf die verschiedenen Arbeitsschritte eingegangen und die jeweiligen Programme kurz erläutert. Da die Arbeit den Schwerpunkt auf die Simulationsbeschleunigung von Pyro-Simulationen legt, wird in diesem Kapitel der Bereich der Simulation in Houdini genauer erläutert. Abgeschlossen wird dieses Kapitel mit einer hypothetischen Hochrechnung der Simulationszeit anhand einer ausgewählten Szene und wie man diese Zeit beschleunigen kann.

Als letztes Kapitel folgt ein Fazit, worin die Ergebnisse der Hochrechnung ausgewertet werden. Zudem werden darin zwei wichtige Fragen geklärt. Zum einen, welche

Möglichkeiten gibt es Simulationszeit ohne Verlust von Qualität zu beschleunigen und zum anderen, gibt es dafür eine Universallösung.

1.4 Themeneingrenzung

Die nachfolgende wissenschaftliche Arbeit bezieht sich ausschließlich auf Fluid Gas Simulationen in Bewegbildformaten. Als Software für die Erstellung dieser Simulationen dient das Programm Houdini von Side FX. Für die Erstellung der nötigen 3D-Szene wird das Programm Maya von Autodesk genutzt. Das Rendern der 3D-Szene erfolgt mit dem Arnold Renderer in Maya. Für die abschließende Videonachbearbeitung dient das Programm Nuke von Foundry.

Vergleiche zu anderen Partikelsystemen, Renderprogrammen oder 3D-Programmen werden in dieser Arbeit nicht berücksichtigt, da es sonst den Umfang dieser Arbeit überschreiten würde.

2. Glossar

Mesh

Als Mesh (3D-Netz) bezeichnet man den strukturellen Aufbau eines 3D-Modells, das aus Polygonen besteht. Ein Mesh nutzt Referenzpunkte in den Achsen X, Y und Z um Formen mit Höhe, Breite und Tiefe zu definieren.

Frame

Ein Frame ist ein Einzelbild in einer Videosequenz. Ab 12 Einzelbildern pro Sekunde nimmt das Auge eine flüssige Bewegung war. Bei Spielfilmen wird in der Regel mit 24 Bilder pro Sekunde gearbeitet, wohingegen beim Fernsehen 25 Bilder pro Sekunde genutzt werden.

Timestep

Als Timestep bezeichnet man die Frequenz, in der die Berechnung der Simulation ausgeführt wird. Bei einem Timestep Wert von 1, wird jeder Frame einmal berechnet.

Collider

Ein Collider definiert geometrische Körper, woran die Auswirkung physikalischer Kräfte auf ein Objekt berechnet werden. Sie bewirken, dass Objekte miteinander kollidieren können. Als Beispiel ein Würfel, der auf einen Boden fällt und von ihm abprallt oder auch ein Objekt was sich durch Rauch bewegt und Schlieren nach sich zieht.

Skalarfeld

Skalare sind Größen, die einen Zahlenwert, aber keine Richtung haben. Bei einem Skalarfeld wird jedem Punkt im Raum ein Skalar zugewiesen, beispielsweise eine Temperatur zu einem bestimmten Timestep.

Vektorfeld

Vektoren sind Größen, die einen Zahlenwert und eine Richtung haben. Bei einem Vektorfeld wird jedem Punkt im Raum ein Vektor zugewiesen. Dies kann zum Beispiel die Erdanziehung oder eine Geschwindigkeit mit einer anderen Richtung sein.

Solver

Advektion

Die Advektion beschreibt allgemein das Heranführen von Dingen. Hier bezieht sich die Advektion auf die Heranführung der Bewegung der verschiedenen Felder, (Pyro Solver).

Node

Houdini nutzt für jede Operation, die durchgeführt wird Knoten, sogenannten Nodes. Das können unter anderem Geometrien oder Transformationen sein. Die Knoten speichern die ausgeführten Aktionen und die jeweiligen verwendeten Parameter. Dies ermöglicht das nachträgliche Ändern vorangegangener Aktionen.

Bounding Box

Die Bounding Box ist im Allgemeinen ein Begrenzungswürfel. In dessen Inneren können sich Objekte oder auch Simulationen befinden. In der Pyro Simulation in SideFX Houdini dient sie als ein dreidimensionaler Simulations-Behälter, in dessen Inneren eine Simulation durchgeführt wird.

DOP

Ein DOP ist ein Dynamischer Operator Node in Houdini. Er wird benutzt, um Simulationen zu konstruieren, zu manipulieren und zu steuern.

Rendern

Rendern, oder auf Deutsch Bildsynthese, beschreibt den Prozess, aus Daten ein Bild zu erzeugen. Dazu gehören typischerweise die Beschreibung der 3D Szene in Lage und Größe der Objekte sowie Materialeigenschaften und Abstrahlcharakteristiken der platzierten Objekte. Mittels dieser Daten wird ein neues Bild von einer virtuellen Kamera erzeugt (Dutre et al., 2018).

Features und Kameratrack

Ein Feature ist ein bestimmter Punkt im Bild, den ein Tracking-Algorithmus erfassen und über mehrere Frames hinweg verfolgen kann. Durch die Verfolgung des Punktes werden Informationen zu der Position generiert und können anschließend zur Bewegungserstellung berechnet werden. Nach der Berechnung kann eine virtuelle Kamera erzeugt werden. Dieser Vorgang bewirkt den sogenannten **Kameratrack**.

Keyframes

Durch einen Keyframe werden bestimmte Werte auf einem Frame festgesetzt. Werden beispielsweise zwei Keyframes gesetzt, wird die Differenz der Werte automatisch für jeden Frame der zwischen den Keyframes liegt berechnet. Als Beispiel: Hat jeweils ein Keyframe auf Frame 1 den Wert „1“ und auf Frame 10 den Wert „0“, wird pro Frame der Wert „0,1“ abgezogen. Somit ist der Wert ab Frame 10 „0“.

3. Definitionen von Simulation

Um die Effizienzsteigerung von Simulationen zu analysieren, werden zunächst die verschiedenen Simulationstypen erläutert. Grundsätzlich können Simulationen in zwei verschiedene Kategorien unterteilt werden, die physikalisch korrekte Simulation und die artistische Simulation. Diese werden im folgenden Kapitel näher betrachtet.

Eine physikalisch korrekte beziehungsweise realistische Simulation berücksichtigt alle relevanten vorhandenen physikalischen Größen. Dazu gehört zum Beispiel die Schwerkraft, die Dichte und das Gewicht eines Materials oder der Luftwiderstand. Diese Simulationen werden vor allem in den Branchen der Luft- und Raumfahrt, der Automobilbranche oder auch der Schiffbau Branche eingesetzt. Grund dafür sind ausgiebige Tests im Vorfeld des eigentlichen Produktionsvorgangs. Dadurch können einzelne Bauteile bereits auf die erforderlichen Ansprüche getestet und gegebenenfalls optimiert werden, sollten die Vorgaben nicht erreicht werden. Zu solchen Tests gehören beispielsweise die Messung des Auftriebs oder des aerodynamischen Verhaltens eines Flugzeugflügels oder die Druckstabilität, der Auftrieb eines Schiffsrumpfs oder auch die Messung des Abriebs während eines Bremsvorgangs bei einem Automobil.

Eine artistische Simulation kommt häufig in dem Film- und Fernsehbereich zum Einsatz. Hierbei handelt es sich meist um realistisch nachgestellte Elemente, nur folgen diese in der Regel nicht allen physikalischen Gesetzen. Beispielsweise ist es üblich, die Schwerkraft bei bestimmten Elementen so zu verändern, dass diese sich zwar nicht mehr physikalisch korrekt verhalten, aber für das menschliche Auge als „richtig” wahrgenommen werden, einfach steuerbar sind und der erstellten Choreografie folgen.

Die im Rahmen dieser Arbeit angefertigten Simulationen werden ohne Rücksicht auf physikalische Korrektheit erstellt und fallen somit in den Bereich der artistischen Simulation.

4. Fluid Simulationen

4.1 Fluid Simulationen allgemein

Mit Fluid Simulationen können die Wirkungen von Flüssigkeiten in einer Umgebung nachgestellt werden. Der Begriff Flüssigkeit kann hierbei neben Wasser auch Feuer, Rauch, Gas oder ähnliches enthalten. Somit ist mit dem Begriff Flüssigkeit nicht explizit eine fließende Masse gemeint, sondern wird er eher als Konzept verstanden. Bei der Feuer-Simulation bilden diese die Veränderung beim Übergang von einem Brennstoff in den gasförmigen Zustand ab. Ab einer bestimmten Temperatur nimmt die Menge an aufsteigendem Rauch zu. Dies ist auf den Transport des entstehenden Rußes durch die Hitze der Verbrennung zurückzuführen. Wenn diese Rauchpartikel abkühlen, lösen sie sich in der Luft auf und zerstreuen sich (*Cineversity Fluid Simulation Wiki*, o. D.).

Um diese Wirkungen zu beschreiben werden zunächst die zugrunde liegenden Begriffe erläutert. Hierzu gehören die Voxel, der Pyro-Solver, die verschiedenen Emitter, VDBs und die verschiedenen Kräfte innerhalb der Simulation.

4.2 Funktionsweise Fluid Gas Simulationen

4.2.1 Voxel

Der Begriff Voxel beschreibt die Menge an Raum, die ein 3D-Objekt einnimmt. Jedes Voxel stellt unterschiedlich gefüllte und regelmäßig abgetastete Räume dar, ähnlich wie ein dreidimensionaler, leerer Würfel. Das Abtasten dient zur Sicherstellung, dass alle Inhalte in dem Voxel angezeigt und gegebenenfalls aktualisiert werden. Um dieses Vorgehen möglich zu machen, wird ein polygonales Mesh in eine volumetrische Repräsentation umgewandelt. Diese kommt dem Eingabe-Mesh sehr nah, ähnlich wie bei einem Rasterisierungsprozess im zweidimensionalen Raum (*Voxel*, o. D.). Hierbei gilt, je kleiner die Voxelgröße, desto detaillierter ist die Auflösung des Objektes.

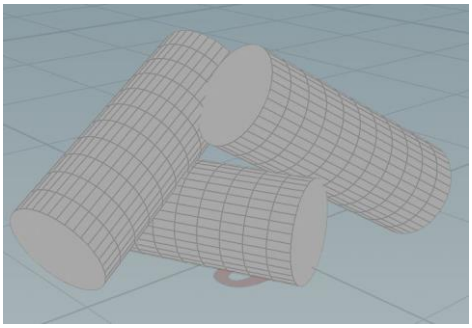

Abbildung 1: Originalgeometrie

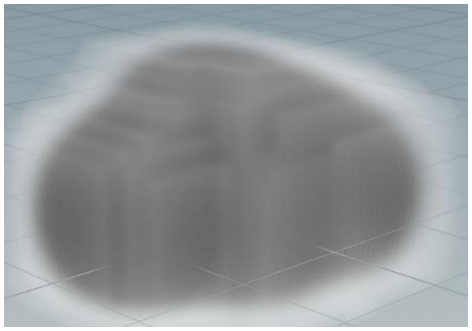

Abbildung 2: Voxelgröße 0,02

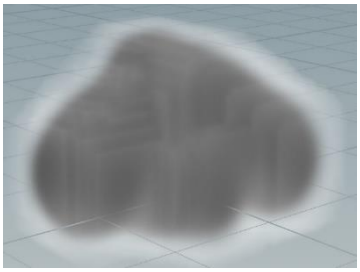

Abbildung 3: Voxelgröße 0,01

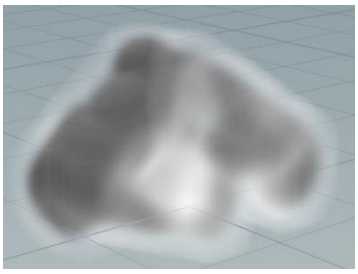

Abbildung 4: Voxelgröße 0,005

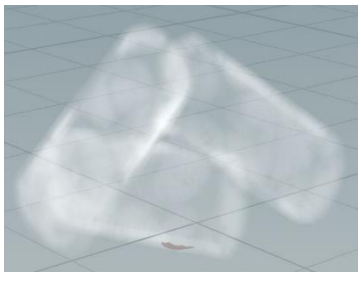

Abbildung 5: Voxelgröße 0,001

4.2.2 Pyro Solver

Der Pyro-Solver in Houdini ist ein rein volumetrischer Flüssigkeits-Solver. Die skalaren und vektoriellen Felder enthalten die Daten, um den Zustand der Flüssigkeit darzustellen. Diese Felder werden von einem Pyro-Source-Objekt und einem Smoke-Objekt erstellt. Dazu gehören unter anderem ein velocity-Feld (Abb. 6), ein *density*-Feld (Abb. 7), ein *temperature*-Feld (Abb. 8) und ein *heat*-Feld (Abb.9). Im Folgenden werden die verschiedenen Felder aufgeführt und kurz erläutert.

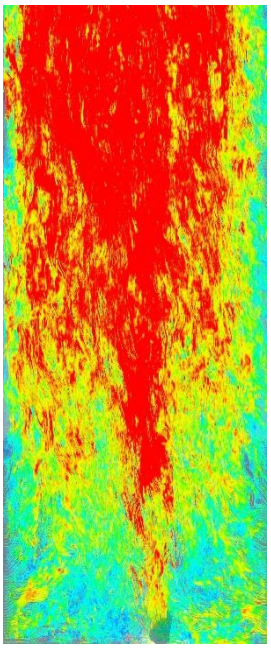

Abbildung 6: velocity-Feld

Abbildung 7: density-Feld

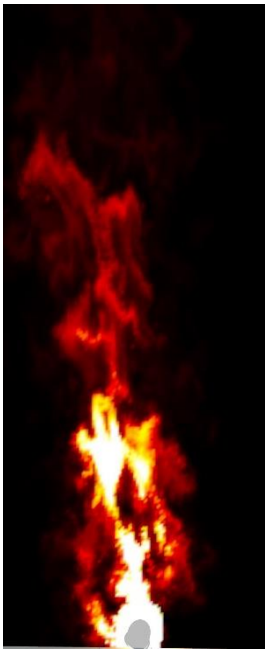

Abbildung 8: temperature-Feld

Abbildung 9: heat-Feld

Das *velocity*-Feld ist ein Vektorfeld und enthält die momentanen Geschwindigkeiten der Flüssigkeit. Die Bewegung des Rauchs wird durch sein eigenes *velocity*-Feld bestimmt.

Das *density*-Feld ist ein Skalarfeld und erfasst die Menge des sichtbaren Rußes. In Regionen mit hoher Dichte erscheint der Rauch dicker.

Ebenfalls ein Skalarfeld ist das *temperature*-Feld. Dieses erfasst die Verteilung der Wärme. Heißeres Gas steigt aufgrund des Auftriebs schneller nach oben.

Das *heat*-Feld ist ein Skalarfeld und speichert die verbleibende Lebensdauer von bspw. Brennstoff, welcher von der Strömung transportiert wird. Das Feld wird über die Source aufgefüllt, während der Pyro-Solver sich um die Reduzierung seiner Werte und die Erzeugung der gewünschten Ergebnisse kümmert. Je nachdem, wie nah der Brennstoff an der Erschöpfung ist, kann er verschiedene Outputs erzeugen. Er kann

zum Beispiel Ruß erzeugen, die Temperatur erhöhen und eine Ausdehnung verursachen. Darüber hinaus kann das *flame*-Feld als Emissionsfeld für die Visualisierung und das Rendering verwendet werden (*Flames*, o.D.).
Das *divergency*-Skalarfeld ist zuständig für die eingestellte Expansion und Kompression. Es koppelt die Flüssigkeit mit Collidern, um somit die Entstehung von Wirbeln innerhalb von Strömungen zu unterstützen.
Die *collision*- und *collisionvel*-Feld enthalten eine Maske, die die Existenz von Collidern bzw. deren Geschwindigkeiten überprüft. Das *collision*-Feld ist in den Bereichen, in denen ein Kollisionsobjekt vorhanden ist, positiv und überall wo keines vorhanden ist negativ.
Der Pyro-Solver ist für die Entwicklung der unter anderem zuvor genannten Simulationsfelder zuständig. Die Art und Weise der Entwicklung entspricht hierbei dem Rauch. Bei jedem Zeitschritt werden folgende Operationen des Solvers durchgeführt:

Zunächst wird die Temperatur verbreitet und abgekühlt. Hierbei repräsentiert die Verbreitung den Transport der Hitze von den heißen zu den kühleren Bereichen. Die Kühlung zeigt die Wärmeabnahme beziehungsweise die Wärmestrahlungsabnahme.
Im Anschluss werden das *flame*-Feld, die Dichte, die Temperatur und die Geschwindigkeit advektiert. Dabei bezieht sich die Advektion auf den Transport der Flüssigkeit und bewirkt die Bewegung aller relevanten Felder.
An dritter Stelle wird das *flame*-Feld aktualisiert und dessen Ausgaben generiert. Hierbei speichert das *flame*-Feld die verbleibende Lebensdauer der Partikel.
Ist dieser Schritt abgeschlossen, wird die Dichte zerstreut, sodass die Rauchdichte über die Zeit reduziert wird und letztlich verschwindet.
Sollten in dem Quellen-Eingang des Solvers (source input) noch zusätzliche Größen vorhanden sein, werden sie in diesem Schritt der Simulation hinzugefügt.
Anschließend wird der Simulations-Container angepasst. Dabei sollte der Container stets groß genug sein, um die Bewegungen der Simulation einzufangen, aber auch klein genug, um zusätzlich unnötige Rechenleistungen zu vermeiden.
Danach werden verschiedenen Kräfte auf das Geschwindigkeitsfeld angewandt, wie zum Beispiel Wind oder Turbulenz. Diese werden im Kapitel 4.2.5 Kräfte genauer erläutert.

Im nächsten Schritt wird der Auftrieb hinzugefügt. Das heiße Gas hat eine geringere Dichte und wird durch den Auftrieb nach oben geleitet. Gesteuert wird dieser Effekt durch die Temperatur. Das Ergebnis wird zur Geschwindigkeit addiert.
Anschließend wird die Viskosität in der Geschwindigkeit berücksichtigt. Sie bezieht sich auf die dynamische Dicke der Flüssigkeit. Viskosere Flüssigkeiten neigen weniger dazu, chaotischen Bewegungsmuster zu folgen.
Es folgt die Anwendung der Shape-Operatoren. Der Pyro-Solver enthält diverse Operatoren, die die Erscheinung und die Bewegung des Rauchs beeinflussen. Hierzu gehören zum Beispiel die Turbulenz sowie Unruhe, Zerkleinerung oder Spaltung der einzelnen Partikel.
Im vorletzten Schritt werden benutzerdefinierte Kräfte hinzugefügt, sofern der Benutzer welche erstellt hat. Diese werden mit dem Force-Input des Solver verbunden.
Als letzten Schritt wird die *pressure projection* (Druckprojektion) ausgeführt. Diese stellt sicher, dass das Geschwindigkeitsfeld am Ende eines Timesteps mit der vorgegebenen Zielabweichung der Expansion und Schrumpfung übereinstimmt, indem sie die aktuellen Werte mit den zu erreichenden Werten abgleicht *(Understanding how pyro works*, o. D.).

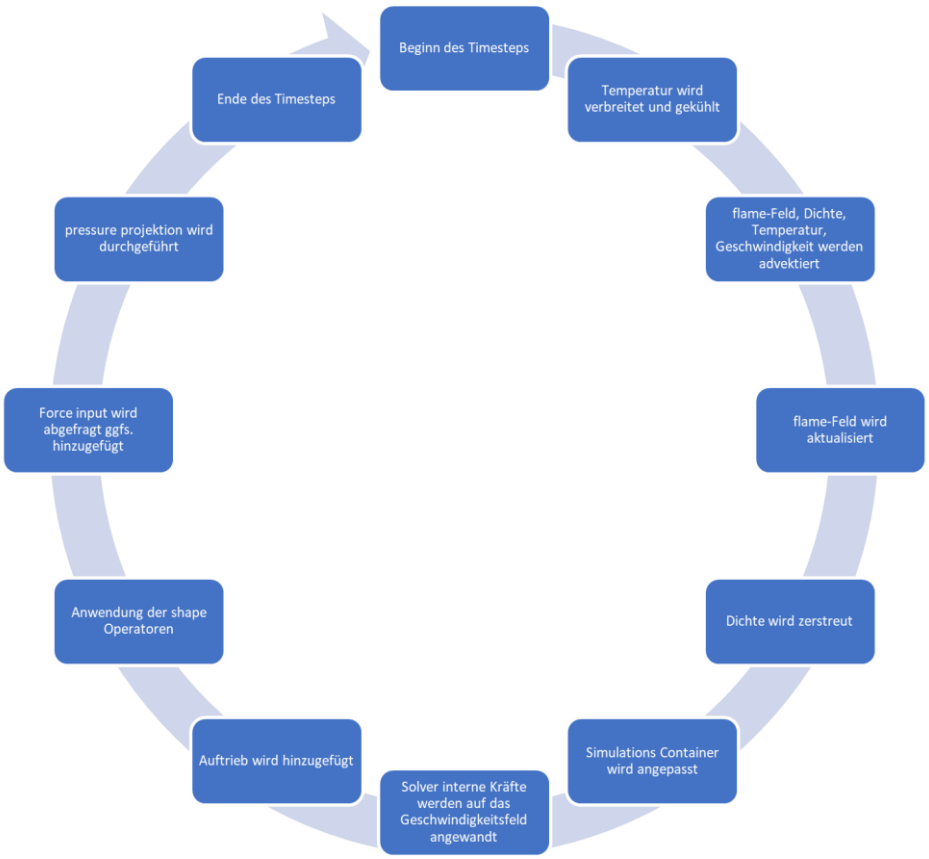

Abbildung 10: Operationen Pyro Solver pro Timestep

4.2.3 Emitter

Emitter sind die Grundlage der Simulation. Man unterscheidet im Allgemeinen zwischen emitter geometry, emitter particles und emitter volume.

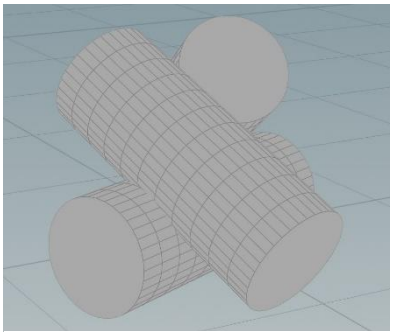

Abbildung 11: emitter geometry

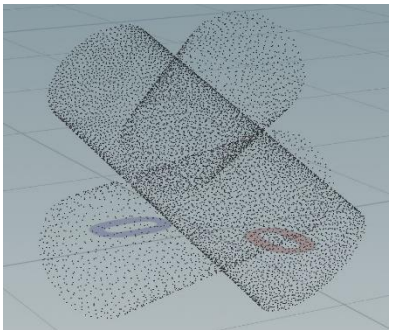

Abbildung 12: emitter particles

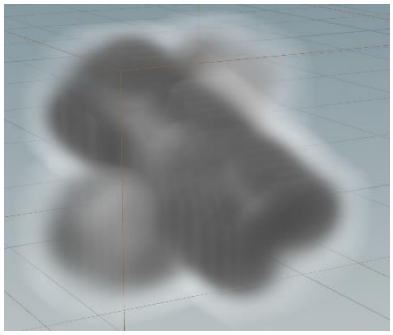

Abbildung 13: emitter volume

Mit der *emitter geometry* (Abb. 11) ist die Geometrie gemeint, die als Ausgangspunkt für die Simulation genutzt wird, wie zum Beispiel ein Würfel, eine Kugel oder ähnliches.

Um die *emitter particles* zu erzeugen (Abb. 12), werden auf der emitter geometry punkte erzeugt. Dies geschieht mithilfe eines Iso-Offset-Nodes und eines Scatter-Nodes. Der Scatter-Node generiert Punkte auf der Oberfläche der Geometrie. Die Anzahl kann hierbei frei gewählt werden. Der Iso-Offset-Node hingegen erzeugt ein density Feld. Aus diesem density Feld generiert er Punkte, die auch im Inneren der Geometrie zu finden sind. Anschließend werden die erzeugten Punkte in ein Pyro-Source-Node gegeben, die die nötigen Felder für die Simulation erstellt. Die Pyro-Source kann automatisch Punkte erzeugen, doch hat man hierbei weniger Kontrolle als mit der manuellen Variante mittels dem Iso-Offset- und Scatter-Node.

Für die Erzeugung der *volume-Source* (Abb.13) werden die zuvor generierten Punkte in ein Volumen umgewandelt. Dies ist nötig, da der Pyro-Solver ein rein volumetrischer Flüssigkeits-Solver ist und somit wird eine volumetrische Quelle benötigt. Hierfür dienen die von der Pyro-Source erzeugten Felder *fuel*, *temperature* und *density* als Grundlage. Die *emitter volume* ist die Quelle für die Visualisierung im Pyro-Solver (*Pyro for Beginners: Smoke Emitters in Houdini*, 2020).

4.2.4 VDB

VDB ist eine Abkürzung für ein **v**olumetrisches, **d**ynamisches Gitter, das diverse Eigenschaften mit sogenannten **B**+Trees teilt. Ein B-Tree Algorithmus ist eine Weiterentwicklung des Binary Search Trees (Abb. 14). Der Binary Search Tree ist ein einfacher Algorithmus, zur Suche von speziellen Werten. Wird nach einem Wert gesucht, sogenannten Key Nodes, beginnt es bei einem Root Node. Dieser Root Node ist der Ausgangswert des Binary Search Trees, in diesem Fall der Wert 50. Ist der gesuchte Key Node größer als der Root Node, wird er auf die rechte Seite gespeichert. Ist er kleiner, wird er auf die linke Seite gespeichert. Hierbei kann jeder Key Node nur mit zwei neuen Key Nodes verknüpft sein (Tarjan, 1984). Die von einem B-Tree verarbeiteten Daten sind in der Regel so groß, dass sie nicht auf einmal in den Arbeitsspeicher passen. Der B-Tree kopiert einzelne Seiten von der Festplatte in den Hauptspeicher und schreiben die Seiten, die sich geändert haben, zurück auf die Festplatte (Cormen et al, 2009).

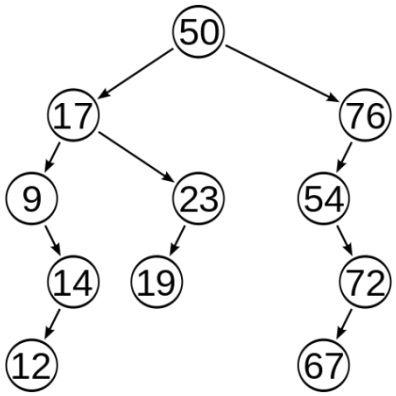

Abbildung 14: Binary Search Tree

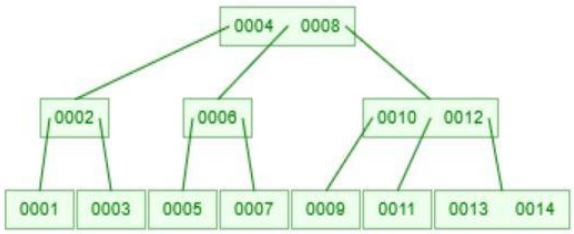

Abbildung 15: B-Tree bis Wert 14

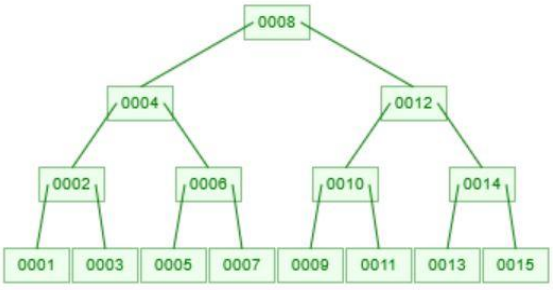

Abbildung 16: B-Tree bis Wert 15

In einem B-Tree (Abb.15) kann jeder Node eine maximale Anzahl (**m**) children besitzen, in diesem Beispiel ist **m=3**. Die unterste Reihe an Werten wird Leaf Nodes genannt. Jeder Weg zu den Leaf Nodes ist gleich lang. Jeder Node, außer es handelt sich um einen Leaf Node, mit **k** children beinhaltet **k-1** Keys. Zudem besitzt jeder nicht Leaf Node, ausgenommen dem Root Node, mindestens⌈m2⌉ children. Der Root Node besitzt mindestens zwei children, solange er selbst kein Leaf Node ist. Wird die Anzahl der Keys durch Hinzufügen eines Wertes überschritten, wird der B-Tree um eine Stufe erhöht. Zur Veranschaulichung wurde hier der Wert 15 hinzugefügt (Abb. 16).

Der Unterschied zu dem B+Tree besteht darin, dass bei dem B+Tree im Vergleich zu dem B-Tree alle Informationen in den Leaf Nodes gespeichert werden. Dadurch ist der Verzweigungsfaktor der Key maximiert (Cormen et al, 2009). Das beschleunigt das Lesen verschiedener Werte, da die Leaf Nodes aufsteigend angeordnet sind.

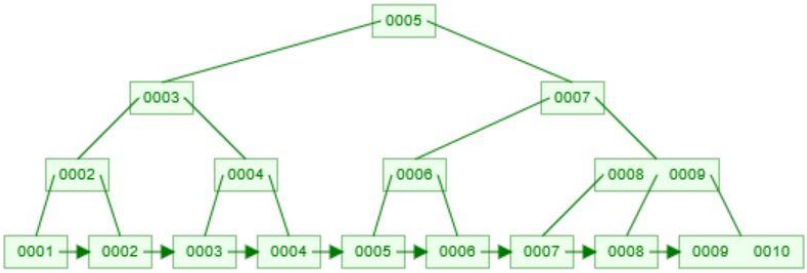

Abbildung 17: B+Tree

Durch diese Anordnung ist es effizienter beispielsweise die Werte 7 und 10 zu lesen, indem das Programm durch die Verkettung der Leaf Nodes nur zwei Nodes weiter gehen muss (Abb.17).

Ein VDB nutzt den räumlichen Zusammenhang von zeitvariablen Daten, um die Datenwerte und die Gitter-Topologie getrennt und kompakt zu kodieren. Dadurch benötigt eine VDB-Operation weniger Speicherplatz und ermöglicht einen schnellen Zugriff auf die vorhandenen Daten (*Museth*, 2013). Vereinfacht dargestellt wird bei einer Pyro-Simulation durch eine Umwandlung in ein VDB nicht der ganze Inhalt der Bounding Box berechnet, sondern nur der Inhalt, der tatsächlich vorhanden ist. Durch das Auslassen der Berechnung von leeren Voxeln, benötigt die Simulation weniger Speicherplatz und lässt sich dadurch flüssiger abspielen Siehe (Abb. 18).

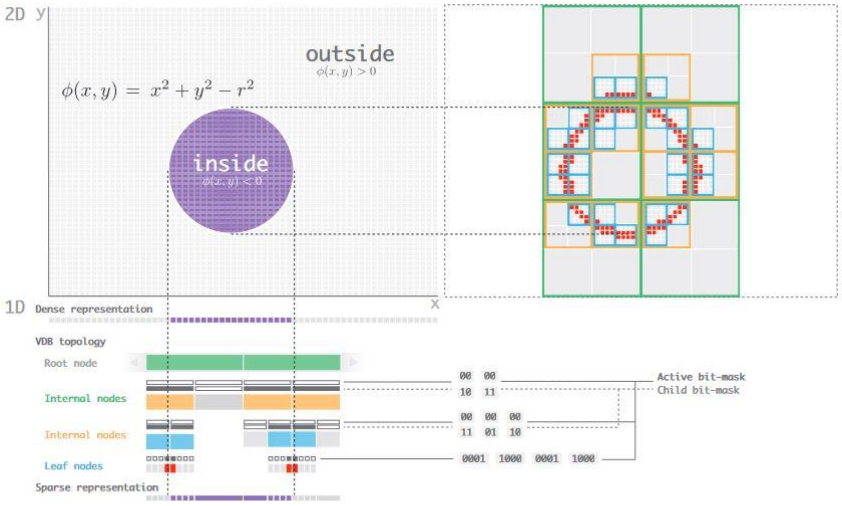

Fig. 3. Illustration of a narrow-band level set of a circle represented in, respectively, a 1D and 2D VDB. Top Left: The implicit signed distance, that is, level set, of a circle is discretized on a uniform dense grid. Bottom: Tree structure of a 1D VDB representing a single y-row of the narrow-band level set. Top Right: Illustration of the adaptive grid corresponding to a VDB representation of the 2D narrow-band level set. The tree structure of the 2D VDB is too big to be shown. Voxels correspond to the smallest squares, and tiles to the larger squares. The small branching factors at each level of the tree are chosen to avoid visual cluttering; in practice they are typically much larger.

Abbildung 18 : 1D und 2D VDB

4.2.5 Kräfte

Der Pyro-Solver ermöglicht es, diverse Kräfte hinzuzufügen, wie unter anderem eine separate Windkraft oder zusätzlich erzeugte Turbulenzen. Diese Kräfte werden dem Pyro-Solver mittels eines Force-Input hinzugefügt und bei jedem Timestep addiert, den der Solver ausführt. Die wohl bekannteste Kraft hierbei ist die Gravitation. Sie wirkt sich auf die gesamte Simulation aus.

Die Windkraft wird durch ein Gas-Wind-DOP angewendet. Hierbei passt sie das Geschwindigkeitsfeld in Richtung des Umgebungswindes an. Diese Kraft ist ein Mikrosolver, der bei dem Aufbau komplexerer Simulationen verwendet wird. Ein Mikrosolver hat im Vergleich zu dem Pyro-Solver mehr Anpassungsmöglichkeiten und kann dadurch feiner auf die Simulation abgestimmt werden. Der Pyro-Solver ermöglicht es, Mikrosolver zur Optimierung oder Erweiterung nach oder vor dem Hauptsolver-Schritt hinzuzufügen (*Gas Wind*, o. D.).

Die Kraft *turbulence* erzeugt ein globales Turbulenz-Feld und wendet dieses an. Diese Kraft ist ebenfalls ein Mikrosolver und kann benutzt werden, um Wirbel zu erzeugen, die sich im Laufe der Simulations-Zeit entwickeln, siehe Abbildung 19 und Abbildung 20. Um sicherzustellen, dass das turbulente Geschwindigkeitsfeld nur in den gewünschten Bereichen auftritt, gibt es ein Kontrollfeld (*Gas Turbulence*, o.D.).

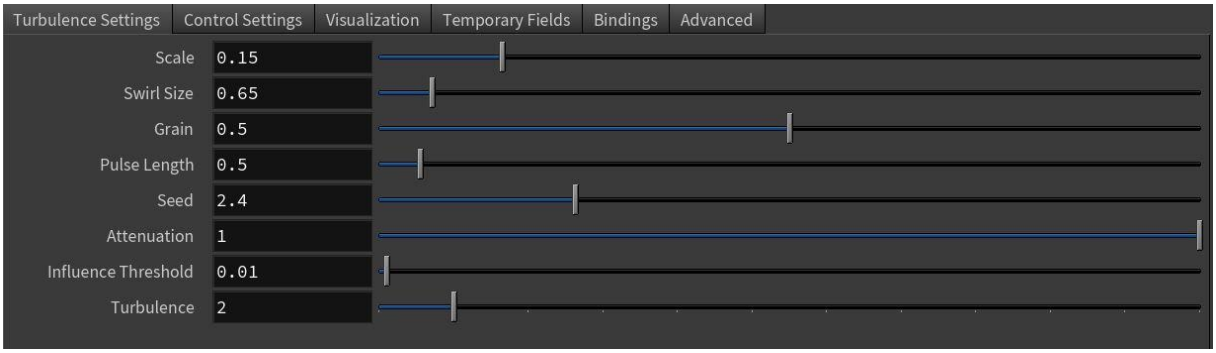

Abbildung 19: turbulence microsolver

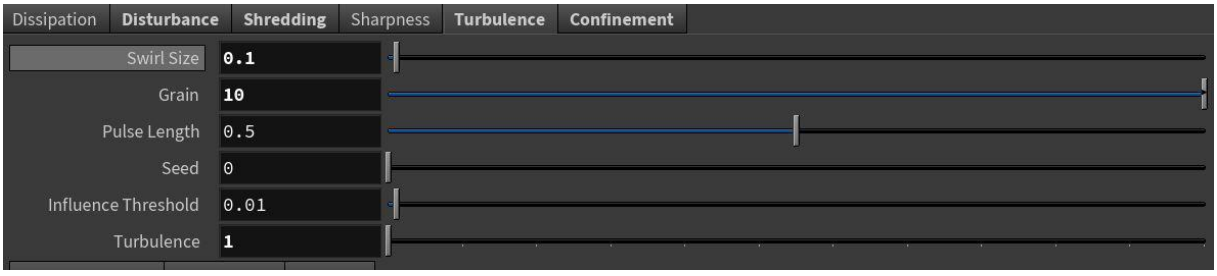

Abbildung 20: turbulence pyro solver

Alle Kräfte werden vor dem Auftrieb dem bestehendem Geschwindigkeitsfeld hinzugefügt und berechnet, wie im Pyro-Solver-Kapitel beschrieben.

4.3 Erläuterung Kameraeinstellungsgrößen

Um die Einsatzgebiete der verschiedenen Simulationstypen besser zu verstehen, werden im Folgenden die wichtigsten Kameraeinstellungsgrößen vorgestellt und jeweils kurz erläutert. Einstellungsgrößen der Kamera werden eingesetzt, um verschiedene Situationen zu verdeutlichen oder weiter zu erzählen, ohne dabei Text von Schauspielern zu benötigen. Manchmal werden sie auch eingesetzt, um einen Ortswechsel zu verdeutlichen oder die Beziehung von Protagonisten zueinander. In diesem Abschnitt werden die Haupt-Einstellungsgrößen aufgezeigt, beginnend mit dem *Establishing Shot*.

Der *Establishing Shot* wird genutzt, um den Ort beziehungsweise die Welt oder die Tageszeit zu etablieren. Diese Einstellung wird zudem auch für den Übergang zwischen verschiedenen Szenen eingesetzt. Dadurch wird es den Zuschauer:innen leichter gemacht, dem Film zu folgen (Maio, 2021).

Abbildung 21: Establishing Shot. Quelle: Breaking Bad Staffel 1 Folge 1, 6:50

Der *Mastershot* beinhaltet ebenfalls, ähnlich wie der Establishing Shot, den Ort und die Tageszeit. Außerdem zeigt er die verschiedenen Charaktere und dessen Beziehungen zueinander auf. Diese Art von Einstellung spielt die Szene in ihrer Gesamtheit ab.

Abbildung 22: Master Shot. Quelle: Breaking Bad Staffel 1 Folge 1, 5:40

Im *Wide Shot* kann emotionale und physische Distanz transportiert werden. Er wird eingesetzt, um beispielsweise die Beziehung eines Charakters oder eines Objekts zu deren Umgebung darzustellen.

Abbildung 23: Wide Shot. Quelle: Breaking Bad Staffel 1 Folge 1, 40:03

Unter dem *Full Shot* versteht man eine Einstellungsgröße von Kopf bis Fuß. Sie ist nah genug dran, um die Mimik eines Charakters einzufangen, jedoch auch weit genug weg, um zu erkennen, was in der Umgebung passiert.

Abbildung 24: Full Shot. Quelle: Breaking Bad Staffel 1 Folge 1, 38:01

Der *Medium Full Shot* wird auch der *Cowboy Shot* genannt. Die Einstellungsgröße ist ungefähr von der Hüfte bis über den Kopf. Sie wird benutzt, um einen Charakter konfliktfreudig oder konfrontationsfreudig zu zeigen.

Abbildung 25: Medium Full Shot. Quelle: Breaking Bad Staffel 1 Folge 1, 27:56

Die beliebteste Einstellungsgröße ist der *Medium Shot*. Grund dafür ist die Neutralität. Er vermittelt weder Drama, wie zum Beispiel ein Close Up, noch Distanz, wie bei einem Wide Shot. Die Darstellung der Charaktere ist sehr nah an der Entfernung wie man allgemein mit Menschen interagiert. Dadurch erweckt es einen bekannten beziehungsweise normalen Eindruck.

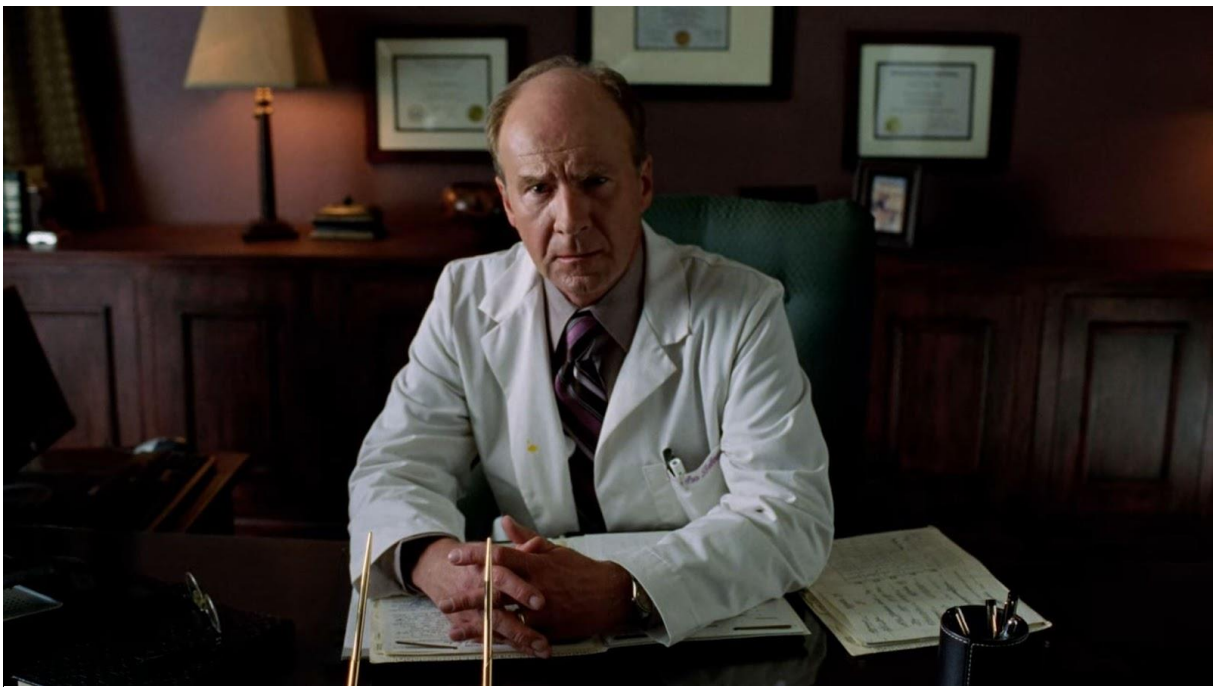

Abbildung 26: Medium Shot. Quelle: Breaking Bad Staffel 1 Folge 1, 18:51

Um die Geschichte und die Charakterdetails zu priorisieren und Ablenkungen zu reduzieren wird oftmals der *Medium Close Up Shot* genutzt. Damit werden Reaktionen im

Abbildung 27: Medium Close Up Shot. Quelle: Breaking Bad Staffel 1 Folge 1, 18:20

Gesicht dargestellt. Dies führt dazu, dass die zuschauende Person sich intim mit dem Charakter fühlt.

Um noch mehr Details darzustellen, wird der *Close Up Shot* eingesetzt. Hierbei können Gedanken und Gefühle beispielsweise von den Augen des Charakters abgelesen werden. Dies vermittelt der zuschauenden Person das Gefühl der Empathie (Lannom, 2020).

Abbildung 28: Close Up Shot. Quelle: Breaking Bad Staffel 1 Folge 1, 15:04

5. Herangehensweise Fluid Gas Simulation für Bewegtbild

Simulationen im Bewegtbild können mit verschiedenen Methoden erstellt werden. Im folgenden Kapitel werden vier verschiedene Methoden erläutert und Anwendungsbeispiele aufgezeigt. Um den industriellen Standard abzubilden, wurde im Vorfeld dieser Arbeit ein Interview verfasst und an Personen aus der Filmindustrie gegeben. Die Grundlage der Anwendungsbeispiele beruht auf den Interviewergebnissen.

5.1 Bespoke Simulationen

Der Begriff “bespoke” kommt aus dem englischen und bedeutet maßgeschneidert. Bespoke Simulationen sind auf die jeweilige Situation ausgelegt. Das Verhalten der Simulation kann nach Belieben angepasst werden (Bolt). Sobald eine Simulation mit einem Charakter oder einem bestimmten Objekt interagieren muss, wird die Bespoke Simulation angewandt (Busco). In der Abbildung 29 wird eine Bespoke Simulation veranschaulicht. Hierbei ist ein kleiner Teil der Feuer-Simulation als Bespoke erstellt worden. Dort interagiert das Feuer mit einer Person, die sich hindurchbewegt und dadurch einen „Feuerschweif” hinter sich zieht. Die benötigte Rechenleistung ist hierbei deutlich höher als bei den anderen Simulationsarten. Aus diesem Grund wird sie oftmals mit anderen Simulationsarten kombiniert. In der Abbildung ist nur der Teil, der mit der Person interagiert, eine Bespoke Simulation. Der Rest des Feuers ist eine Pre-Cached Simulation. Diese Art wird im Folgenden erläutert.

Abbildung 29: Effekt einer Bespoke Simulation Quelle: Fantastic Beasts (Crimes of Grindelwald) 1:51:56

5.2 Pre-cached Simulationen

Das Programm „Houdini” speichert Simulationsdaten in den Arbeitsspeicher, um eine schnelle Wiedergabe zu ermöglichen. Dieser Vorgang wird „cachen” genannt. Sollte der Arbeitsspeicher nicht ausreichen, löscht das Programm immer den ersten Frame, um einen neuen Frame zu speichern. Wie viele Frames in den Arbeitsspeicher geladen werden können, hängt von den Einstellungen in dem dazugehörigen DOP Node ab. Dort kann die maximale Speicherplatzgröße bestimmt werden, welche das Programm benutzen kann. Zudem gibt es die Möglichkeit, die gesamten Simulationsdaten auf eine Festplatte zu speichern. Hierzu speichert Houdini eine Simulationsdatei, die alle Frames beinhaltet und dann als Datei von einer Festplatte gelesen wird. Das Abspielen der Simulation von einer Festplatte ist langsamer als das Abspielen aus dem Arbeitsspeicher, jedoch ermöglicht es das Speichern von gesamten komplexen Simulationen. Das Speichern in eine Simulationsdatei (pre-cachen) erleichtert den Einsatz in einer dreidimensionalen Szene, da die Simulation bereits vollständig gespeichert ist.

Eingesetzt werden solche Simulationen in Bereichen, bei denen keine spezielle Interaktion mit der Simulation nötig ist. Oftmals im Mittelgrund oder Hintergrund der Szene (Bolt). Zudem ist es üblich, eine Bibliothek von pre-cached Simulationen in der Vorproduktion anzufertigen. Dies spart zum einen Zeit während der Produktion, zum anderen können die caches auch schnell und leicht platziert werden. Dadurch kann sich die FX Abteilung mehr auf bespoke Simulationen konzentrieren (Busco). In Filmen mit niedrigem Budget kommt es vor, dass die pre-caches mehrmals verwendet werden. So wurden bei dem Film „Geostorm“ Rauch- und Feuersimulationen sowie Explosionen öfter genutzt (Nixon).

Der Nachteil an den pre-cached Simulationen ist, dass das Verhalten der Simulation nicht mehr beeinflusst werden kann. Beispielsweise kann eine Feuer-Simulation nicht mit einer Person interagieren, wenn diese durch das Feuer läuft. Lediglich das Aussehen kann noch verändert werden. Hierzu zählen unter anderem die Skalierung oder ein veränderter Startzeitpunkt des Feuers.

5.3 Simulationen im Upresing Verfahren

Bei dem Upres-Verfahren wird die jeweilige Simulation zunächst in einer niedrigen Auflösung erstellt. Nach der Fertigstellung der Simulation wird das Upres-Verfahren angewandt. Die niedrige Auflösung der Haupt-Simulation hat zur Folge, dass sie nach dem Upresing weniger Detail besitzt als eine Simulation mit anfänglich hoher Auflösung. In der Abbildung 30 werden verschiedene Auflösungen dargestellt. Auf der linken Seite befindet sich die Rauchsimulation mit der Voxelgröße von 0.2. In der Mitte hat die Simulation eine Voxelgröße von 0.5 und auf der rechten Seite wurde das Upres Verfahren angewandt. Vergleicht man das Aussehen der beiden äußeren Simulation, erkennt man den Detailverlust durch das Upresing-Verfahren deutlich, da sie in der Ausgangs-Simulation nicht berücksichtigt worden sind.

Abbildung 30: Upresing Darstellung

In der Praxis wird das Upresing Verfahren eher selten eingesetzt. Es verkompliziert den Arbeitsablauf, da ein weiterer Arbeitsschritt hinzugefügt wird. Jedoch können dadurch bei hoch aufgelösten Simulationen sogenannte "Cross hatches" (Kreuzschraffuren) beseitigt werden. (Nixon). Wird es angewandt, werden die Feuersimulationen in der Regel in mittlerer Auflösung vorgefertigt. Anschließend wird durch das Upresing lediglich Detail hinzugefügt und es ändert nicht die allgemeine Ästhetik der Simulation (Bolt).

5.4 Vorgefertigte 2D Elemente

Bei vorgefertigten 2D-Elementen handelt es sich um zweidimensionale Simulationen beziehungsweise Animationen. Diese Art der Simulation benötigt im Vergleich zu einer dreidimensionalen Simulation deutlich weniger Rechenleistung. Aufgrund der Zweidimensionalität sind Kamerabewegungen beziehungsweise Kamerafahrten nur parallel zu dem 2D-Element möglich. Sobald eine dreidimensionale Fahrt vorhanden ist, verzerren sich die vorgefertigten 2D-Elemente beispielsweise ein Feuer. Bei Objekten, die nicht lange zu sehen sind, fällt dieser Effekt kaum auf. Ein gutes Beispiel hierfür ist das Mündungsfeuer einer ausgelösten Waffe. Dieses Vorgehen wurde beispielsweise in dem Film „Jupiter Ascending“ (2015) genutzt, um die Mündungsfeuer der futuristischen Waffen abzubilden (Nixon), siehe Abbildung 23. Vorzugsweise werden sie jedoch in den Hintergrund gestellt oder in Einstellungen genutzt, wo es wenig Parallaxe gibt (Busco). Bei dieser Simulationsart ist eine Veränderung des Verhaltens nicht möglich. Lediglich das Aussehen kann bedingt verändert werden, wie bei der Pre-cached Simulation.

Abbildung 31: Mündungsfeuer Quelle: Jupiter Ascending, 29:18

6. Praktischer Teil

6.1 Einleitung

In diesem Kapitel werden die verschiedenen Schritte des von mir angefertigten praktischen Projekts erläutert. Für den folgenden Teil werden Grundkenntnisse im Bereich 3D-Computergrafik vorausgesetzt, da es sonst den Umfang der Arbeit überschreiten würde. Der Arbeitsablauf in jedem genutzten Programm wird jeweils zusammengefasst und kurz erläutert. Beginnend mit der Idee des Projektes. Anschließend wird auf die Umsetzung der verschiedenen Bereiche eingegangen. Hierzu zählen die Dreharbeiten, die Pyro-Simulation, das Kameratracking, das Erstellen einer 3D-Szene und die abschließende Videonachbearbeitung. Die genutzten Modelle wurden als 3D-Objekte heruntergeladen und sind frei nutzbar (cgtrader).

6.2 Idee

Es werden 3D-Modelle von Möbeln als Grundlage für verschiedene Pyro-Simulationen genutzt. Hierbei dienen die vorab genannten Simulationstypen als Ausgangspunkt. Da es sonst den Umfang des Projektes überschreiten würde, werden lediglich die Bespoke- und Pre-cached Simulationen berücksichtig und jeweils in halber Auflösung gerendert, was in diesem Fall eine Auflösung von 960x540 Pixel bedeutet. Die Simulationen werden abschließend in das angefertigte Realfilmmaterial integriert und farblich der Umgebung angeglichen.

6.3 Durchführung des Projektes

6.3.1 Dreharbeiten

Als Ort für das Realfilm-Material diente die Lutterburg in Bodenwerder. Neben den Videoaufnahmen wurden Aufnahmen eines Colorcheckers angefertigt. Diese dienen als Grundlage für das farbliche Anpassen im abschließenden Videonachbearbeitungsprogramm. Dort werden die Farben der Simulation mit den Farben des Videomaterials abgeglichen und gegebenenfalls angeglichen. Zudem wurde ein LiDAR-Scan der verschiedenen Räume erstellt. Der Scan dient als Referenz für die Erstellung der 3D-Szene.

Abbildung 32: Drehort Original

Abbildung 33: Drehort als Scan

6.3.2 Pyro-Simulation in Houdini

In diesem Abschnitt wird eine Pyro-Simulation an dem Beispiel von einem brennenden Stuhl erläutert. Erstellt wurde die Simulation in „Houdini“ von SideFX. Die Herangehensweise kann in der Regel in drei Hauptschritte unterteilt werden: die Erstellung der Ausgangsgeometrie, die Erzeugung der Simulation mittels eines Pyro Solvers und das Importieren beziehungsweise Umwandeln in ein VDB, falls nötig.

Als erster Schritt wird die Ausgangsgeometrie erstellt. In diesem Fall ist es ein fertig modellierter Stuhl. Dieser wird mittels eines *transform* Nodes in die nötige Position gebracht. Danach wird er mittels eines *iso-offset* und *scatter* Nodes in Punkte umgewandelt, wie bereits bei dem Kapitel 4.2.3 Emitter beschrieben, und anschließend in den pyro-source Node weitergegeben. Dieser erzeugt das *fuel*-, *temperature*- und

density-Feld für die Simulation. Infolgedessen wird *ein attribute noise* erzeugt. Dieser Node erzeugt auf den im Node ausgewählten Simulationsfeldern Unregelmäßigkeiten, um den Feldern Bewegung zuzufügen. Um diese Unregelmäßigkeiten zu animieren, können bei Bedarf Formeln in die jeweiligen Felder eingefügt werden. In diesem Fall handelt es sich um die Formel *„$T*2“.* Das bedeutet, dass die aktuelle Zeit in Sekunden mal zwei genommen wird, um die Bewegung zu erzeugen. Dies dient der Dynamik in der späteren Simulation, da die Ausgangsgeometrie durch dieses Vorgehen mit der Zeit pulsiert und dadurch immer wieder die Form leicht verändert. Im Anschluss auf den ersten Schritt folgt ein *volume rasterize attributes* Node. Dieser wandelt die erzeugten und animierten Felder in ein Volumen um. Das Volumen wird benötigt, um die Simulation im Pyro Solver darzustellen. Abschließend wird ein *null* Node erzeugt, der das Ergebnis aller vorherigen Nodes repräsentiert.

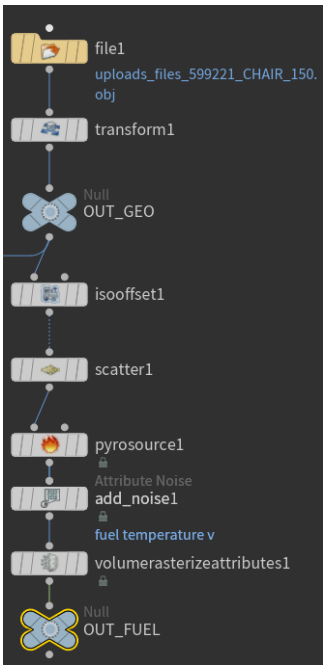

Abbildung 34: Nodebaum Geometrie

Als zweiter Schritt wird ein *pyro solver* Node erstellt. In diesem Node wird die Simulation erstellt. Der *pyro solver* Node benötigt mindestens zwei Variablen, um die Simulation berechnen zu können. Zum einen eine Volume Source und zum anderen ein *smoke objekt* Node. Als *volume source* Node wird der zuvor erzeugte *null* Node genutzt und dient als Grundlage der Simulation. Der *smoke object* Node definiert die Voxelgröße und die anzuzeigenden Felder während der Simulation. Diese beiden Nodes werden in den *pyro solver* Node gegeben und ermöglichen die Berechnung der Simulation wie in Kapitel 4.2.2 Pyro Solver beschrieben. Nachdem die Berechnung der Simulation erfolgt ist, wird die Erdanziehung, hier der *gravity* Node, angewandt. Anschließend wird das Ergebnis mittels eines *output* Nodes ausgegeben. Ab diesem Zeitpunkt ist die Simulation exportfähig.

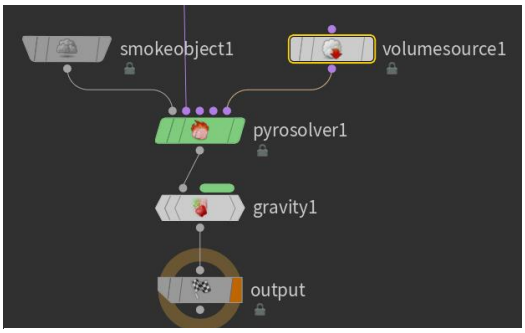

Abbildung 35: Nodebaum Pyro Simulation

Als dritter Schritt erfolgt das Darstellen des Ergebnisses des pyro solvers mittels eines *dop i/o* Nodes. Dieser dient zur Veranschaulichung des Ergebnisses des *pyro solver* Nodes. Er lädt das Ergebnis der Simulation und kann es divers weiterverarbeiten. In diesem Fall wandelt er das Ergebnis der Simulation mittels *convert to VDB* Nodes in ein VDB um. Das wird dann durch einen *file cache* Node auf einer ausgewählten Festplatte gespeichert. Alternativ kann mithilfe des VDB File Caches eine Loop Simulation erstellt werden. Ein Loop ist in diesem Fall eine ständige Wiederholung von einer festgesetzten Anzahl von Frames. Wie ein solcher erstellt wird, wird im Folgenden kurz erläutert.

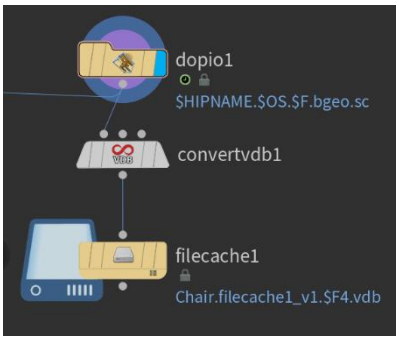

Abbildung 36: Nodebaum Dop I/O

Um einen Loop zu erzeugen, wird im Vorfeld eine gecachte Simulation benötigt. Dieser Cache beinhaltet die Anzahl der Frames, die geloopt werden sollen. In diesem Beispiel sind es 50 Frames, beginnend bei Frame 50 und endend bei Frame 100. Der Cache wird dann in zwei *timeshift* Nodes gegeben. In dem ersten *timeshift* Node wird bei dem Frame Feld die Formel „($F % 50) + 50“ eingegeben. Das Frame Feld beinhaltet den aktuellen Frame der Simulation. Die Formel innerhalb der Klammer bedeutet, dass das Programm die Simulation bei Frame 1 startet und sobald es Frame 50 erreicht hat, wieder auf den ersten Frame zurückspringt. Durch das „+ 50“ hinter der Klammer wird der erste Frame zu Frame 50. Somit werden mit dieser Formel die Frames von 50 bis

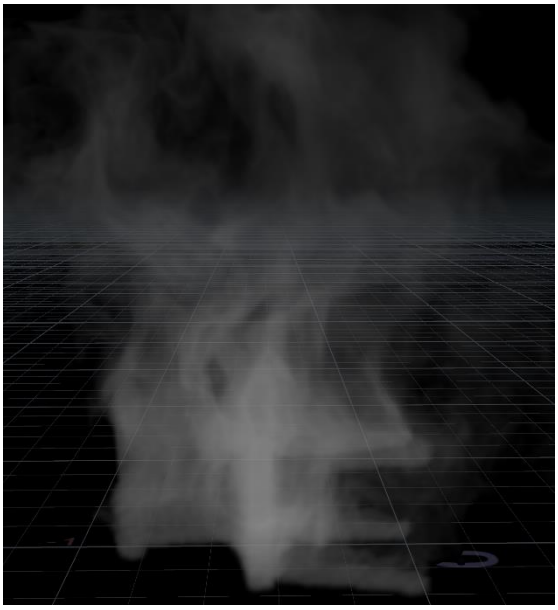
Abbildung 37: Simulation Frame 49


Abbildung 38: Simulation Frame 50

100 wiederholt. Der *timeshift* Node „springt“ somit von Frame 100 auf Frame 50. Dies ist deutlich in der Simulation zu sehen. Um einen offensichtlichen Beginn des Loops zu vermeiden, wird ein zweiter *timeshift* Node erstellt. In diesem wird bei dem Frame Feld die Formel „(($F + 25) % 50) + 50“ eigetragen. Durch das Hinzufügen des „+ 25“ innerhalb der ersten Klammer, wird der „Sprung“ verschoben. Die Formel wiederholt ebenfalls 50 Frames, nur beginnt es nicht bei Frame 50, sondern bei Frame 75. Sobald Frame 100 erreicht ist, springt die Simulation ebenfalls auf Frame 50 zurück. Somit ist der „Sprung“ in der Mitte der Simulation. Um diesen Effekt unsichtbar zu machen, werden die beiden erzeugten *timeshift* Nodes in ein *blend shapes* Node eingespeist. In diesem werden mittels Keyframes die beiden *timeshift* Nodes überblendet. Dabei ist am Anfang der Simulation der zweite *timeshift* Node ganz zu sehen. Dies hat den Grund, dass in diesem Node die Simulation bereits zur Hälfte durchgelaufen ist. In der Mitte der Simulation ist der erste *timeshift* Node zu sehen. Zu diesem Zeitpunkt ist dieser zur Hälfte durchgelaufen. Bis zum Ende der Simulation wird wieder auf den zweiten Node zurückgeblendet. Daraus resultiert eine flüssige Simulation, die beliebig lange simuliert werden kann. Abschließend wird diese Simulation in ein VDB umgewandelt und kann als VDB Cache platziert werden.

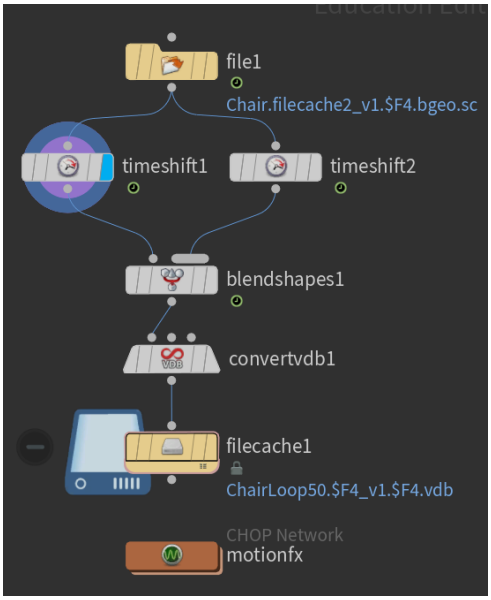

Abbildung 39: Nodebaum Loop Simulation

6.3.3 Kameratracking

Das Kameratracking wurde in dem Programm „After Effects“ von Adobe erstellt. Hierfür wurden in dem Videomaterial zunächst Features erstellt, die dann getrackt wurden. Anschließend wurde eine virtuelle 3D-Kamera und sogenannte „Nullebenen“ erstellt. Um eine 3D-Szene einrichten zu können, wurde die 3D-Kamera und die Nullebenen mithilfe eines Scripts in eine „alembic“ Datei exportiert. Das Skript wandelt die Nullebenen in sogenannte „Locator“ für Maya um. Das Einrichten der Szene fand in dem Programm „Maya“ von Autodesk statt.

Abbildung 40: getrackte Nullebenen

6.3.4 3D-Szene

Die 3D-Szene wurde in dem Programm Maya von Autodesk erstellt. Als Grundlage der Szene dienen die 3D-Kamera und die exportierten Locator. Die Locator sind feste Punkte in einem 3D-Raum, an dem der erstellte LiDAR-Scan ausgerichtet werden kann. Dieser wiederum dient als Referenz für das Erstellen von einer Set-Geometrie. Die Set-Geometrie beinhaltet in diesem Fall die in Maya nachmodellierte 3D-Umgebung. Der Umgebung werden Eigenschaften zugewiesen, die denen des Drehortes entsprechen. Die Set-Geometrie wird genutzt, um Reflektionen von Licht und Schatten darzustellen. Anschließend wurden die erstellten Pyro-Simulationen in die Szene importiert. Daraufhin wurde die Szene mit zwei Renderlayern als jeweils eine JPEG Sequenz exportiert. Der erste Layer beinhaltet die Simulation und der zweite Layer die Set-Geometrie. Die Sequenzen werden dann in einem Videonachbearbeitungsprogramm zusammengefügt und nachbearbeitet.

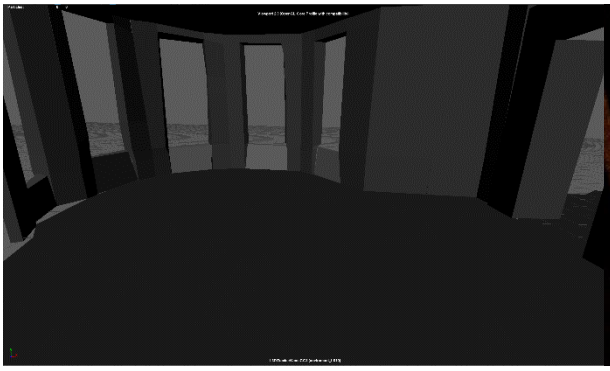

Abbildung 41: Set Geometrie nachmodelliert

Abbildung 42: Set Geometrie mit Reflektionen

6.3.5 Videonachbearbeitung

In diesem Schritt wurden die verschiedenen Videosequenzen der Simulation mit der Videosequenz des Realfilm-Materials zusammengefügt. Dafür werden die verschiedenen Ausgangssequenzen mittels *Merge* Nodes verbunden (in Abbildung 43 rot markiert). Die *Merge* Nodes verbinden die beiden Eingänge miteinander. Das können sie auf verschiedene Weise. In diesem Fall wurde eine „over“ Operation ausgeführt. Dabei wird das Videomaterial „A“ über das Videomaterial „B“ gelegt und ausgegeben. Als Programm hierfür diente „Nuke“ von Foundry. Zudem wurde mithilfe des Colorcheckers und Color Correction Nodes (in Abbildung 43 weiß markiert) die Farbe der Simulation an das Realfilm-Material angepasst und abschließend als Video ausgegeben.

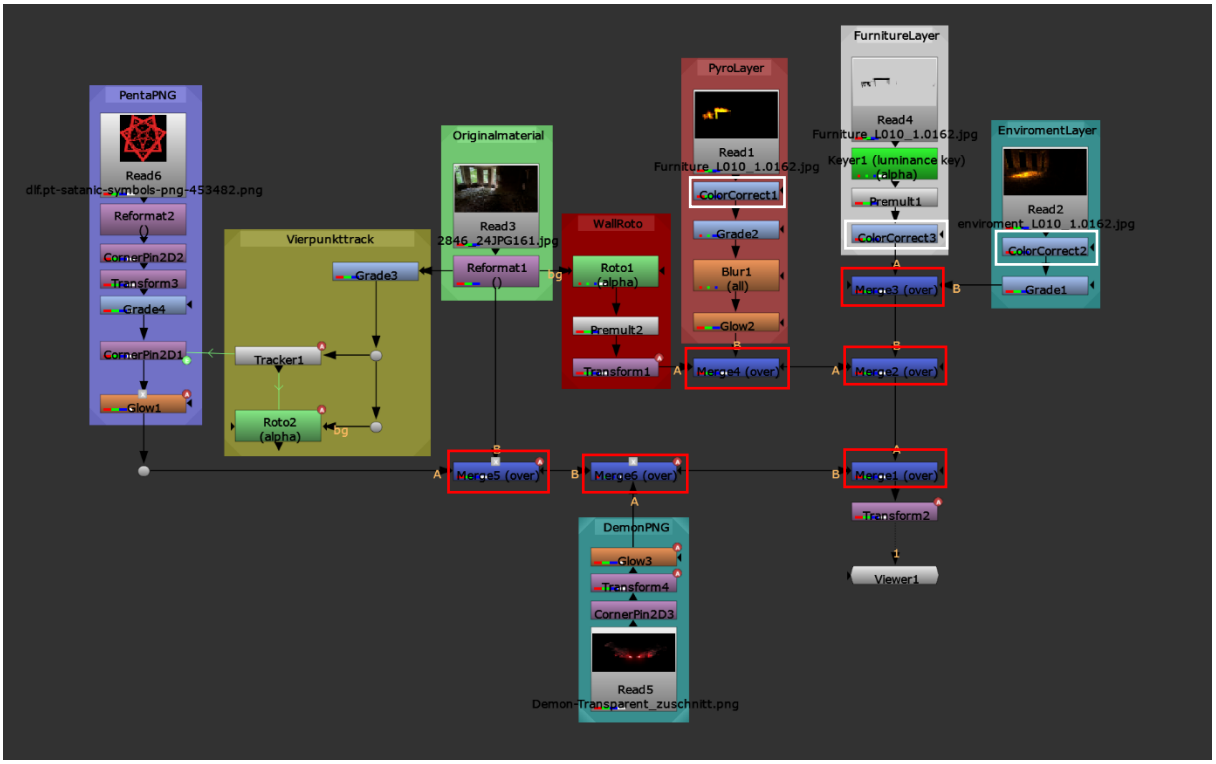

Abbildung 43: Nodebaum in Nuke

6.4 Zeitrechnung

In diesem Abschnitt wird der Zeitfaktor der Simulationen erläutert und Möglichkeiten aufgezeigt, Zeit einzusparen. Ausgegangen wird von den mir zur Verfügung stehenden Ressourcen. Entgegen dem Industriestandard wird hier lokal simuliert und gerendert. Zudem werden für die Berechnung bestimmte Bedingungen festgesetzt, um Daten erheben zu können. In diesem Fall bedeutet das, dass sich die zu simulierenden Feuer in mittlerer bis weiter Distanz zu einer Kamera befinden und keine Interaktion mit Objekten oder Personen nötig sind. Darauf resultierend bezieht sich diese hypothetische Rechnung ausschließlich auf Pre-cached Simulationen. Als Beispiel dient die das Zusammentreffen der Avengers 1:50:40- 1:51:03 in dem Film „Marvel Studio´s The Avengers“ aus dem Jahr 2012.

Abbildung 44: Zusammentrefefn der Avengers. Quelle: Marvel Studio´s The Avengers, 1:50:40

Wird angenommen, dass in der oben genannten Szene alle 11 sich am Boden befindenden Pyro-Simulationen angefertigt werden müssen und keine Bibliothek bereits erstellter Pyro-Simulationen vorhanden ist, würde es pro Simulation 164:15 Minuten, also circa 2:30 Stunden dauern. Diese Zahlen resultieren aus einer Hochrechnung des von mir zur Verfügung stehenden Systems. Dieses benötigt zur Simulation von 48 Frames 14:17 Minuten. Bei 23 Sekunden ergeben sich 552 Frames. Dadurch resultiert eine Simulationszeit der 11 Simulationen von 27:30 Stunden. Die Zeit bezogen auf Arbeitstage mit einer durchschnittlichen Arbeitszeit von 9 Stunden ergibt drei

Arbeitstage. Das bei einem durchschnittlichen wöchentlichen Gehalt von rund 1600 € als VFX Artist (*Gagenkompass*, o. D.) ergeben sich 800 €, bei regulären 6 Arbeitstagen pro Woche. Hierbei ergibt sich die Möglichkeit lediglich eine Pyro-Simulation zu simulieren und sie anschließend mittels eines erstellten VDB File Caches mehrfach zu platzieren. Die platzierten Caches können anschließend im Aussehen angepasst werden, um sie sich optisch deutlicher unterscheiden, beispielsweise durch das Anpassen der Skalierung, der Rotation oder der Geschwindigkeit. Das Abspeichern in ein VDB dauert in diesem Beispiel 9:24 Stunden. Das geschieht einmalig. Hinzu kommt das einmalige Simulieren. Dieser Schritt würde eine Zeitersparnis in der Simulationszeit von circa 18:06 Stunden bewirken, also eine Ersparnis von 533 €.

Eine weitere Möglichkeit Simulationszeit einzusparen, ist die Erstellung beziehungsweise das Verwenden von VDB Loops. Hierbei ist das einmalige Berechnen eines Loops nötig. Dieser kann dann, ähnlich wie der VDB File Cache, beliebig oft in einer 3D-Szene platziert werden. In diesem Beispiel wird für eine Loop Simulation von 50 Frames eine Zeit von 14:57 Minuten und für das anschließende Berechnen und das Speichern noch einmal 39 Minuten. Dieser Vorgang würde in dem oben genannten Beispiel 26:37 Stunden einsparen. Auf den finanziellen Aspekt bezogen, wäre es eine Einsparung von 774 €.

Anschließend können die VDBs in der Szene gerendert werden. Bei dem Rendern kann ohne Qualitätsverlust nur bedingt Zeit eingespart werden. Zum Beispiel, indem nicht jede Pyro-Simulation einzeln, sondern alle Simulationen zusammen gerendert werden.

7. Fazit

Die Arbeit setzte sich mit der Frage auseinander, welche Möglichkeiten es gibt eine Simulationszeit ohne Qualitätsverlust zu beschleunigen und ob es eine Universallösung hierfür gibt. Es gibt verschiedenste Möglichkeiten, die Simulation zu beschleunigen. Allerdings ist es nicht möglich, eine Universallösung dafür festzusetzen. Es ist eher Situationsabhängig. Jede Szene und jede Einstellung müssen einzeln betrachtet werden, um dann entscheiden zu können, welche Art der Simulation einzusetzen ist. Zudem ist es in der Industrie bei größeren Effekt-Studios üblich, Zugriff eine Bibliothek mit Pre-cached Simulation zu haben. Dadurch allein kann schon Zeit eingespart werden. Außerdem kommt es auf die Häufigkeit der Pyro-Simulationen im gesamten Film an. Dabei rentiert sich eine Erstellung einer Bibliothek im Vorfeld nur, wenn es klar ist, dass viele Simulationen nötig sind, die sich im Hintergrund oder mittelweiter Entfernung zu der Kamera vorhanden befinden. Existieren zum Beispiel im gesamten Film nur drei Sequenzen mit jeweils zwei bis drei Pyro-Simulationen, würde sich der Arbeitsaufwand nicht lohnen, eine Bibliothek hierfür anzufertigen.

Während der Erstellung des Praxisprojekts ist mir bewusst geworden, dass die Zeitersparnis relativ zu der Anzahl der Einstellungen steigt. Somit ist die Simulationsbeschleunigung bei nur einer Szene mit drei Pyro-Simulationen kaum bemerkbar. Wohingegen es bei mehreren Szenen und Einstellungen eine große Zeitersparnis zu Folge hat, da die Simulationszeiten nicht mehr berücksichtigt werden müssen. Im Gegensatz zu meinem System, werden in der Industrie leistungsstärkere Systeme verwendet. Es ist zudem üblich, das Rendern auf sogenannte „Renderfarms“ auszulagern. Es gibt somit keine Universallösung zur Simulationsbeschleunigung ohne den Verlust von Qualität.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Texte und Abbildungen, die wörtlich, bildlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Lemgo, 18.08.2022

Patrick Schmidtmeyer

Quellenverzeichnis

- *Cineversity Fluid Simulation Wiki*. (o. D.). Cineversity. Abgerufen am 18. Februar 2022, von https://www.cineversity.com/wiki/Fluid_Simulation/
- *Understanding how pyro works*. (o. D.). How Pyro Works. Abgerufen am 10. Februar 2022, von https://www.sidefx.com/docs/houdini/pyro/background.html
- *Gas Wind*. (o. D.). Gas Wind Node. https://www.sidefx.com/docs/houdini/nodes/dop/gaswind.html
- *Gas Turbulence*. (o. D.). Gas Turbulence Node. Abgerufen am 19. Februar 2022, von https://www.sidefx.com/docs/houdini/nodes/dop/gasturbulence.html
- Museth, K. (2013). VDB: High-Resolution Sparse Volumes with Dynamic Topology. ACM Transactions on Graphics, 32(3), 1–22
- Tarjan, R. E. (1984). *Data Structures and Network Algorithms (CBMS-NSF Regional Conference Series in Applied Mathematics, Band 44)*. Society for Industrial and Applied Mathematics
- Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2009). *Introduction to Algorithms, third edition (Mit Press)* (third edition). The MIT Press
- *Pyro for Beginners: Smoke Emitters in Houdini*. (2020, 1. September). [Video]. YouTube. https://www.youtube.com/watch?v=dQkAisr2_3E
- Maio, A. (2021, 26. Januar). *What is an Establishing Shot? Creative Examples that Set the Tone*. StudioBinder. Abgerufen am 26. Februar 2022, von https://www.studiobinder.com/blog/what-is-an-establishing-shot-definition-examples/?utm_source=youtube&utm_medium=video&utm_campaign=content-marketing-promotion&utm_term=establishing-shot&utm_content=theshotlist-shotsizes-video
- *Voxel*. (o. D.). Autodesk Help. Abgerufen am 30. Januar 2022, von https://help.autodesk.com/view/MAYAUL/2022/ENU/?query=voxel

- *Flames*. (o. D.). Flames. Abgerufen am 30. Januar 2022, von https://www.sidefx.com/docs/houdini/pyro/flames.html
- Lannom, S. C. (2020, 1. Oktober). *Guide to Camera Shots: Every Shot Size Explained*. StudioBinder. Abgerufen am 1. März 2022, von https://www.studiobinder.com/blog/types-of-camera-shots-sizes-in-film/
- Bolt, T. (2022, 30. März). Interview.
- Nixon, M. (2022, 31. März). Interview.
- Busco, M. (2022, 1. April). Interview.
- Dutre, P., Bekaert, P. & Bala, K. (2018). *Advanced Global Illumination*. Amsterdam University Press.
- *Gagenkompass*. (o. D.). Bundesverband Filmschnitt Editor e.V. Abgerufen am 10. August 2022, von https://bfs-filmeditor.de/gagenkompass/
- Statista. (2022, 1. Februar). *Erfolgreichste Filmreihen und -franchises nach Einspielergebnis bis 2022*. Abgerufen am 14. August 2022, von https://de.statista.com/statistik/daten/studie/932924/umfrage/erfolgreichste-filmreihen-und-franchises-nach-einspielergebnis/
- cgtrader, 3D-Modell Stuhl, von https://www.cgtrader.com/items/599221/download-page
- cgtrader, 3D-Modell Kerzen, von https://www.cgtrader.com/free-3d-models/architectural/lighting/crate-and-barrel-denby-candle-holders
- cgtrader, 3D-Modell Tisch, von https://www.cgtrader.com/free-3d-models/furniture/table/table-model-textures
- Dämon, PNG, von https://www.pngall.com/de/demon-png/download/9428
- Pentagram, PNG, von https://www.dlf.pt/ddetail/TbmJbx_crowley-7-pointed-star-hd-png-download/

Abbildungsverzeichnis

Interview Michael Nixon, *31.03.2022, 10:16*

Patrick Schmidtmeyer
mail: patrick.schmidtmeyer@stud.th-owl.de

Good day,

In the context of my bachelor thesis, I deal with the topic of speeding up simulations in relation to visual quality. For this I would like to get some insight into industry workflows and techniques. Below you will find a few questions about the topic Pyro FX. I would be grateful if you could answer them and send them back to me.
Many thanks in advance.

Patrick Schmidtmeyer

What is your name, surname?
Michael Nixon

What is your profession and how long have you been working in it?
FX supervisor/ VFX artist 19 years.

Were there, or are there, situations where you used one or more of the following techniques?
(bespoke simulations per shot, pre-cached simulation elements, pre-rendered 2D elements, simulations with upresing procedure)

bespoke simulations per shot
Traditionally we try and sim for every shot, but in the past as simulations were longer and more painful to get right, we would re-use caches quite a lot. Even between shows. On GREYHOUND we simmed 500 fully interactive water shots, and we were able to do this by running a very well tested water pipeline. Some shots ran through the first time, with very little setup time.

pre-cached simulation elements
On films like Geostorm where the budget was low, we would work hard to reuse caches and have robust setups. Smoke columns, fires and explosions all tended to be re-used. On larger scale shows like Pacific Rim 2 we would simulate everything from scratch as the interaction of falling buildings wouldn't allow for re-use.

pre-rendered 2D elements
We have pre-rendered 2d elements for previs, and sometimes for comp. Muzzle flashes are a good example, you can render them from each angle and have a script in Nuke which will allow you to align them quickly, without a new render. We used this approach on Jupiter Ascending for the futuristic weapons fire.

simulations with upresing procedure
Upresing sims is a tricky one, a lot of the time it's easier just to sim higher. Adding another stage does complicate the pipeline. Ignoring the extra effort, upresing can be useful to help smooth out artifacts in the simulation, it can remove the nasty cross hatching and that's very useful.

Are there any examples you can give with a justification why exactly this technique was or is right for this situation?
Probably my favourite example of re-use has been the wand FX in Harry Potter, we used the same caches for 3 films. The explosions from Captain America were re-used a lot too . As were the smoke stacks from Greenzone. The most rewarding shots tend to be special and unique. I really enjoyed helping setup the Sentinel Swarms on the latest Matrix, which was made using a custom crowd network, which was super fun and fast to use.

31.03.2022, 10:16

Interview Maria Busco, *01.04.2022, 00:23*

Patrick Schmidtmeyer
mail: patrick.schmidtmeyer@stud.th-owl.de

Good day,

In the context of my bachelor thesis, I deal with the topic of speeding up simulations in relation to visual quality. For this I would like to get some insight into industry workflows and techniques. Below you will find a few questions about the topic Pyro FX. I would be grateful if you could answer them and send them back to me.
Many thanks in advance.

Patrick Schmidtmeyer

What is your name, surname?
Maria Busco

What is your profession and how long have you been working in it?
FX lead/supervisor - VFX artist 11 years.

Were there, or are there, situations where you used one or more of the following techniques?
(bespoke simulations per shot, pre-cached simulation elements, pre-rendered 2D elements, simulations with upresing procedure)

bespoke simulations per shot
Whenever the sims need to be emitted or collide with a specific asset or character we tend to run bespoke sims per shot.

If the length of the project allows it, one technique is to spend some time at the beginning of the show to develop setups that procedurally process an input, for example an animated character, and create a specific effect leaving per shot adjustments to tweaking a limited set of parameters promoted by who created the setup.

This maximizes efficiency and consistency across shots.

pre-cached simulation elements
In a situation similar to the one above, but in which custom interactions are not needed, some time can be spent to create a library of caches that not only save time, but can also then be be placed or scattered easily, automatically or by another department (usually either Layout or Lighting) to dress shots, leaving the fx department more time/people to focus on more demanding/bespoke fx

pre-rendered 2D elements,
A further change in scenario from the one above is that the fx is needed in the background or in cases in which there's minimal parallax. In this case the library can be either pre-rendered fx or comp elements. Lights contribution AOVs can be used to allow some relighting in comp.

simulations with upresing procedure
From my experience upresing hasn't been all that successful when trying to get "more details" out of sims so in cases like a pyro sim that is going to picture in a very close up shot, is often a good idea to advance the sim resolution progressively to optimize iteration speed at the beginning with lower/mid-res and narrowing down the tweaks when getting to the higher resolutions until the desired final result (or time runs out :))

Are there any examples you can give with a justification why exactly this technique
was or is right for this situation?

01.04.2022, 00:23

Interview Tom Bolt, 30.03.2022, 15:23

Patrick Schmidtmeyer
mail: patrick.schmidtmeyer@stud.th-owl.de

Good day,

In the context of my bachelor thesis, I deal with the topic of speeding up simulations in relation to visual quality. For this I would like to get some insight into industry workflows and techniques. Below you will find a few questions about the topic Pyro FX. I would be grateful if you could answer them and send them back to me.
Many thanks in advance.

Patrick Schmidtmeyer

What is your name, surname?
Tom Bolt

What is your profession and how long have you been working in it?
CG Supervisor / FX Supervisor 8years at DNEG

Were there, or are there, situations where you used one or more of the following techniques?
(bespoke simulations per shot, pre-cached simulation elements, pre-rendered 2D elements, simulations with upresing procedure)

bespoke simulations per shot - *reserved for hero highly art direcable moments often with interaction from other assets*

pre-cached simulation elements - *cost saving exercise, used for highly repetitive appearances of assets in shots. often reserved for mg-bg use but not exclusively. It's time better spent up front in build to generate a library of caches rather than the bespoke every shot approach. Also help with consistency.*

pre-rendered 2D elements - *often the above is also rendered out on cards for use in comp as well. This works particularly well on anything that relies less on lighting conditions such as naked flames, muzzle flash etc*

simulations with upresing procedure - *A point of contention, upres "look" has limitations depending on workflow. I prefer to sim high res from scratch once I have a solid mid res look where I know the high res will only add detail and not change the overall aesthetic.*

Are there any examples you can give with a justification why exactly this technique
was or is right for this situation?
How high is high enough in resolution - *when you can no longer see individual voxels. I would advise making a hda which is essentially a 2d image plane. If you add a texture to this and place this at a given depth you will see how many voxels you require at least in the axis that is perpendicular to the camera. As you increase the division size of the card, you essentially start to visualise the required voxel resolution for that depth.*

30.03.2022, 15:23