

BACHELOR-ARBEIT

vorgelegt an der
Technischen Hochschule Würzburg-Schweinfurt
in der Fakultät Informatik und Wirtschaftsinformatik
zum Abschluss eines Studiums im Studiengang Informatik

Rendering klassischer handgezeichneter Zeichentrickanimationen

Angefertigt an der Fakultät für Informatik und Wirtschaftsinformatik der Technischen
Hochschule Würzburg-Schweinfurt

Erstprüfer: Prof. Dr.-Ing. Frank Deinzer

Zweitprüfer: Max Werner

Abgabetermin: 23. 12. 2024

Eingereicht von: Simon Arapoglu aus Fürth



Hiermit versichere ich, dass ich die vorgelegte Bachelorarbeit/Masterarbeit selbstständig verfasst und noch nicht anderweitig zu Prüfungszwecken vorgelegt habe. Alle benutzten Quellen und Hilfsmittel sind angegeben, wörtliche und sinngemäße Zitate wurden als solche gekennzeichnet.

Würzburg, den

(Unterschrift)

Hiermit willige ich ein, dass zum Zwecke der Überprüfung auf Plagiate meine vorgelegte Arbeit in digitaler Form an PlagScan übermittelt und diese vorübergehend (max. 5 Jahre) in der von PlagScan geführten Datenbank gespeichert wird, sowie persönliche Daten, die Teil dieser Arbeit sind, dort hinterlegt werden.

Die Einwilligung ist freiwillig. Ohne diese Einwilligung kann unter Entfernung aller persönlichen Angaben und Wahrung der urheberrechtlichen Vorgaben die Plagiatsüberprüfung nicht verhindert werden. Die Einwilligung zur Speicherung und Verwendung der persönlichen Daten kann jederzeit durch Erklärung gegenüber der Fakultät widerrufen werden.

Würzburg, den

(ggf. Unterschrift)

Übersicht

In dieser Arbeit wird untersucht, wie die Ästhetik klassischer handgezeichneter Zeichentrickanimation in einer dreidimensionalen Umgebung nachgebildet werden kann. Das Ziel ist es, herauszufinden, ob und wie der Stil ikonischer Werke mithilfe moderner Rendering-Techniken erfasst und repliziert werden kann. Dafür werden exemplarisch Animationsfilme von Winsor McCay, den Disney-Studios und Studio Ghibli auf stilistische und technologische Merkmale analysiert, dies bietet die Grundlage der Entwicklung interaktiver 3D-Nachbildungen. Moderne Tools wie Unity und Blender werden vorgestellt und bei ihrer Nutzung wird deutlich, wie hilfreich sie bei der Umsetzung klassischer Techniken sein können. Darüber hinaus werden eigene Animationen erstellt und Shader sowie Post-Processing-Effekte entwickelt und vorgestellt, unter anderem der anisotrope Kuwahara-Filter und ein Kantenerkennungsshader. Originale und entwickelte Szenen werden verglichen und zeigen, dass eine erfolgreiche Nachbildung klassischer Stile möglich ist. Daneben wird auch ihre Performance betrachtet. Moderne Technologien müssen die traditionellen Methoden nicht ersetzen, vielmehr kann durch gezieltes Nachbilden von Stilmerkmalen der Charme klassischer handgezeichneter Zeichentrickanimation bewahrt werden. Darüber hinaus bieten interaktive Nachbildungen neue Perspektiven für kreative Einsätze.

Abstract

This thesis examines how the aesthetics of classic hand-drawn cartoon animation can be re-created in a three-dimensional environment. The aim is to find out whether and how the style of iconic works can be captured and replicated using modern rendering techniques. Animated films by Winsor McCay, Disney Studios and Studio Ghibli will be analyzed for stylistic and technological characteristics, this provides the basis for the development of interactive 3D replicas. Modern tools such as Unity and Blender are introduced and their use shows how helpful they can be in the implementation of classic techniques. In addition, animations are created and shaders and post-processing effects are developed, including the anisotropic Kuwahara filter and an edge detection shader. Original and developed scenes are compared and show that it is possible to successfully recreate classic styles. Their performance is also considered. Modern technologies do not have to replace traditional methods; rather, the charm of classic hand-drawn animation can be preserved through the targeted recreation of stylistic features. In addition, interactive recreations offer new perspectives for creative applications.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Hintergrund und Motivation	1
1.2 Zielsetzung der Arbeit	3
1.3 Aufbau der Arbeit	4
2 Geschichte und Analyse klassischer Animationsstile	7
2.1 Eine kurze Geschichte der 2D-Animation	7
2.2 „Gertie the Dinosaur“ von Winsor McCay	9
2.2.1 Historischer Kontext	9
2.2.2 Stilmerkmale und technische Innovationen	11
2.3 Disney und Mickey Mouse in den 1930er und 1940er Jahren	13
2.3.1 Fokus auf Charakteranimation	13
2.3.2 Visueller Stil	15
2.4 Studio Ghibli und Prinzessin Mononoke	16
2.4.1 Unterschiede zur westlichen Animation	16
2.4.2 Visueller Stil	17
3 Animation und Rendering	21
3.1 Blender als Animations-Framework	21
3.1.1 Allgemeines	21
3.1.2 Animation in 3D	22
3.2 Unity Engine als Rendering-Framework	23
3.2.1 Shader Pipeline	23
3.2.2 Scriptable Render Pipeline	25
4 Umsetzung	27
4.1 Entwickelte Szenen und Shader	27

4.1.1	Gertie-Szene	27
4.1.2	Disney-Szene	32
4.1.3	Ghibli-Szene	39
5	Evaluation	43
5.0	Umgebungsparameter	43
5.1	Authentizität und visuelle Unterschiede	43
5.1.1	Gertie-Szene	43
5.1.2	Disney-Szene	45
5.1.3	Ghibli-Szene	47
5.2	Rechenzeit der Post-Processing-Shader	49
5.2.1	Rechenzeit der Gertie-Szene	50
5.2.2	Rechenzeit der Disney-Szene	50
5.2.3	Rechenzeit der Ghibli-Szene	51
5.2.4	Durchgeführte Experimente	52
6	Zusammenfassung	55
7	Ausblick	59
Literatur		63
Abbildungsverzeichnis		67
Tabellenverzeichnis		69

Kapitel 1

Einleitung

1.1 Hintergrund und Motivation



Abbildung 1.1: Nachbildung eines Zoetrops aus dem viktorianischen Zeitalter [Dun04]

Die Animation ist eine der einflussreichsten Kunstformen des 20. und 21. Jahrhunderts. Ihre Ursprünge reichen zurück in eine Zeit vor dem Film, als bewegte Bilder nur mit eigens entwickelten Maschinen wie dem in Abb. 1.1 abgebildeten Zoetrop abgespielt werden konnten. Mit der Zeit wurde Animation immer beliebter und das Publikum größer. Vorreiter wie Winsor McCay entwickelten Techniken, welche zu Standards wurden und Männer wie Walt Disney machten die Produktion animierter Filme zu einem lukrativen Geschäft. Ihre Animationen leg-

ten den Grundstein für eine Kunstform, welche nicht nur unterhält, sondern auch heute noch Menschen jeden Alters inspiriert und begeistert.

Im Laufe der Zeit wurde die handgezeichnete Zeichentrickanimation aufgrund der sich stetig weiterentwickelnden Technologie immer mehr von computergenerierten Verfahren abgelöst. Diese erlaubten komplexere Szenen und vereinfachten Prozesse, allerdings geht dabei oft die handgefertigte Ästhetik verloren, welche der traditionellen Animation so viel Charme gegeben hat. Computergenerierte Animationsfilme haben oft einen sehr einheitlichen Stil. Dies steht im starken Kontrast zu der individuellen Handschrift des Künstlers, die bei handgezeichneten Animationen deutlich zu sehen ist. Diese Bachelorarbeit setzt es sich zum Ziel, genau diesen Charme mit Hilfe moderner Rendering-Technologien zu replizieren, ohne die visuelle oder emotionale Essenz zu verlieren.



Abbildung 1.2: „Spider-Man: Into the Spider-Verse“ arbeitet Stilmerkmale von 2D-Animation und Comic-Büchern erfolgreich in seine 3D-Animation ein [18]

Eine zentrale Motivation der Arbeit ist es zu erforschen, ob moderne Technologien als Ergänzung für klassische Methoden und Stile genutzt werden können, anstatt diese komplett zu ersetzen. Aktuelle Animationsfilme wie „Spider-Man: Into the Spider-Verse“ (1.2) sind ein Beweis dafür, dass die Kombination verschiedener Stilmerkmale einen einzigartigen Stil hervorbringen kann, der bei Zuschauern und Kritikern gleichermaßen gut ankommt.

Hinzu kommen die erweiterten Möglichkeiten, welche sich bieten, wenn die Emotionalität und Lebendigkeit klassischer handgezeichneter Animation in einer interaktiven Umgebung präsentiert wird. Hierdurch kann der Betrachter sich noch stärker an die ihm präsentierte Welt binden. Diese Möglichkeiten werden einem erst eröffnet, wenn die Stile in der dritten Dimension präsentiert werden.

Weiterhin besteht ein persönliches Interesse an Animation und den Prozessen, die zur Produktion eines animierten Filmes nötig sind. Da es für eine überzeugende Nachbildung unumgänglich ist, sich kritisch mit dem Original zu beschäftigen und dieses zu analysieren, hatte dies den Effekt, dass viele geschichtliche und produktionstechnische Einzelheiten in Erfahrung gebracht werden konnten. So wurde neben einem wissenschaftlichen Beitrag auch ein persönlicher Wunsch befriedigt.

Zusammenfassend lässt sich sagen, dass diese Arbeit nicht nur einen wissenschaftlichen Beitrag leisten möchte, sondern auch eine Brücke zwischen Vergangenheit und Gegenwart schlagen möchte. Sie soll zeigen, dass handgezeichnete Animation weiterhin als Inspirationsquelle dienen kann und dass es sich lohnt, deren kulturelles Erbe zu ehren.

1.2 Zielsetzung der Arbeit

Das Hauptziel der Arbeit besteht darin, die Möglichkeiten und Herausforderungen zu untersuchen, klassische handgezeichnete Zeichentrickanimationen mit Hilfe moderner Rendering-Techniken authentisch in eine dreidimensionale Umgebung zu übertragen. Dabei soll ein ganzheitliches Verständnis für den Prozess aufgebaut werden. Um dies zu ermöglichen, werden Beispiele der Kunstform aus verschiedenen Zeitperioden in Hinsicht auf genutzte Stilmittel analysiert. Hierbei wird sich nicht nur für dynamische Objekte interessiert, auch die Hintergründe, welche einen großen Beitrag zu der Atmosphäre eines animierten Werkes leisten, werden in Betracht gezogen. Diese Informationen werden anschließend genutzt, um den Stil der Zeichentrickanimation authentisch mithilfe von Computergrafik zu rekonstruieren.

Ein zentrales Anliegen ist die Analyse und Rekreation stilistischer und technischer Eigenschaften ikonischer Werke unterschiedlicher Epochen. Dazu gehören der von der Art-Nouveau inspirierten, detailverliebte Stil von Winsor McCays „Gertie the Dinosaur“, die dynamischen Charakteranimationen in Disneys Animationen der 1930er und 1940er Jahre und die atmosphärischen Hintergründe und interessanten Charaktere der Filme von Studio Ghibli. Diese Werke werden auf visuelle und narrative Merkmale untersucht, aber auch der technologische Kontext wird beachtet.

Die gewonnenen Informationen werden dazu genutzt, die Atmosphäre der handgezeichneten Animation so originalgetreu wie möglich innerhalb einer dreidimensionalen Umgebung zu replizieren. Hierzu wird ein Verständnis für aktuelle Tools, Frameworks und Technologien aufgebaut. Dieses Verständnis wird praktisch demonstriert, indem zu jeder der analysierten Zeichentrickanimationen eine 3D-Szene mit Hilfe von Shadern, Lichteffekten und Animationen gebaut wird, welche Stilmerkmale und Atmosphäre des Originals einfangen.

Letztendlich soll auch eine Reflexion der gewonnenen Ergebnisse und Informationen stattfinden. Hierfür wird untersucht, wie weit die erstellten 3D-Reproduktionen den Originalwerken visuell und atmosphärisch nahekommen. Dabei soll auch untersucht werden, in welchen Fällen moderne Technologien hilfreich sind und in welchen sie an ihre Grenzen stoßen.

Die Arbeit soll letztendlich eine Grundlage für zukünftige Projekte darstellen, welche eine Schnittstelle zwischen klassischer Zeichentrickanimation und computergenerierter Animation darstellen. Sie bietet Empfehlungen an für Entwickler und Künstler, die klassische Animationsstile in einer 3D-Umgebung nachbilden möchten. Es soll das Verständnis für die Schnittstelle zwischen traditionellen und modernen Animationstechniken vertieft werden.

1.3 Aufbau der Arbeit

Anfangs werden klassische Animationsstile analysiert. Zunächst wird ein kurzer Einblick in die Geschichte der 2D-Animation geworfen. So wird theoretisches Wissen gesammelt, welches dabei hilft, exemplarisch drei Animationsstile zu analysieren. Diese basieren auf "Gertie the Dinosaur" von Winsor McCay, klassischen Disney-Animationen der 1930er und 1940er Jahre sowie Produktionen von Studio Ghibli. Es werden stilistische Merkmale analysiert, doch auch der historische Kontext und technische Innovationen werden betrachtet.

Das darauf folgende Kapitel soll theoretische Grundlagen für Animationen und Rendering bieten. Zuerst wird Blender als Animations-Framework betrachtet. Besonderer Fokus wird hier auf Animation innerhalb einer dreidimensionalen Umgebung gelegt. Die Unity-Engine wird daraufhin als Rendering-Framework vorgestellt. Es findet eine kurze Erklärung einer Shader-Pipeline statt, danach werden das Konzept und die Vorteile von Unitys Scriptable Render Pipeline dargestellt. Diese Punkte sind essenziell für die Nachbildung klassischer handgezeichneter Animation in 3D, da sie erlauben, das Aussehen der Szene beliebig anzupassen und damit so nah wie möglich an das Original heranzukommen.

Danach werden die gewonnenen theoretischen Erkenntnisse praktisch umgesetzt, in dem die zuvor analysierten 2D-Animationen in dreidimensionale Szenen verwandelt werden. Es wird beschrieben, wie diese aufgebaut wurden und wie die Stilmerkmale der klassischen Animationen übertragen werden konnten.

Daraufhin werden die Authentizität und visuelle Unterschiede der zweidimensionalen Inspiration und ihrer 3D-Nachbildungen evaluiert. Anschließend wird die Performance der zur Nachbildung genutzten Shader analysiert. Es werden Experimente durchgeführt, welche die Shader in verschiedenen Umgebungen testen.

Im Zuge der Arbeit Gewonnene Erkenntnisse werden für den Leser zusammengefasst, daneben wird reflektiert und bewertet, ob die Zielsetzungen der Arbeit erfüllt wurden.

Abschließend gibt das letzte Kapitel einen Ausblick auf weiterführende Forschungsfelder. Es werden Verbesserungsmöglichkeiten für Szenen und Shader formuliert sowie Ideen für potentielle neue Werkzeuge und Technologien, die 3D-Nachbildungen erleichtern könnten. Des Weiteren diskutiert das Kapitel Anwendungspotentiale der Arbeit im Bereich Animation und Game Design.

Kapitel 2

Geschichte und Analyse klassischer Animationsstile

Hier soll ein kurzer Überblick der Geschichte der handgezeichneten 2D-Animation stattfinden. Diese Informationen dienen als theoretische Grundlage für die darauf folgende Analyse von drei verschiedenen Animationsstilen. Die Analyse soll den gefundenen Stilmitteln einen historischen und kulturellen Kontext geben und es uns damit erleichtern, die Elemente der Szenen zu identifizieren, welche zur Erfüllung der Zielsetzung implementiert werden sollen.

2.1 Eine kurze Geschichte der 2D-Animation

Animation wird definiert als eine Abfolge von „Einzelbilder[n], die durch eine Art Mechanismus in schneller Folge betrachtet werden, um die Illusion von Bewegung zu erzeugen“¹ [Cav11, S. 35]. Animationen nutzen das Prinzip der Persistenz des Sehens. Gesehene Bilder verweilen eine kurze Zeit auf der Netzhaut. Werden sie schnell genug nacheinander angezeigt, scheint es dem Zuschauer so, als würde sich vor seinen Augen etwas bewegen.

Bevor Animationen auf Papier verwirklicht wurden oder computergeneriert stattfanden, verlangten die ersten Experimente mit bewegenden Bildern eigens gebaute Maschinen. Eine dieser Maschinen war das Phenakistiskop, welches 1832 vom belgischen Physiker und Mathematiker Joseph Plateau entwickelt wurde. Das Gerät besteht aus einer Pappscheibe, an deren Rand der Reihe nach Bilder angeordnet sind. Der Zuschauer betrachtet die sich drehende Scheibe und konzentriert sich dabei auf einen Punkt. Durch die sich wechselnde Abfolge der Bilder wird die Illusion von Bewegung erzeugt. [Ben16, S. 12-13]

¹Aus dem Englischen übersetzt mit DeepL Translate, Originaltext: „single-frame images viewed in rapid succession by some form of mechanism, to create an illusion of movement“

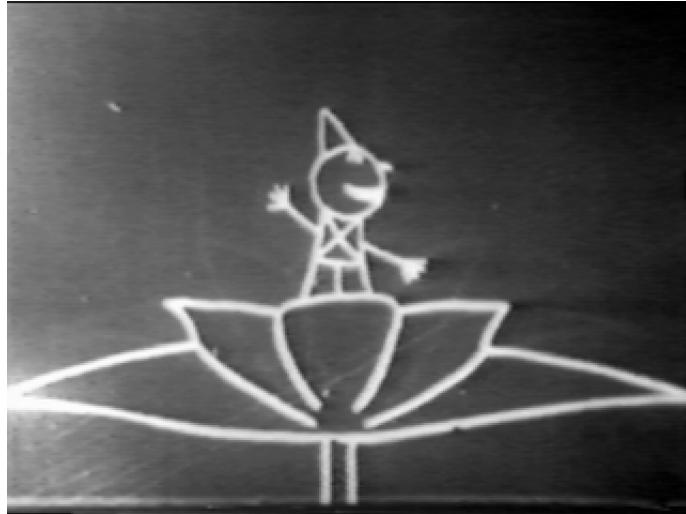


Abbildung 2.1: Émile Cohls Fantasmagorie [Coh08]

Es würden noch 70 Jahre vergehen, bis 1908 Émile Cohls *Fantasmagorie*, siehe Abb. 2.1, veröffentlicht wurde. Das Werk gilt als einer der ersten animierten Filme. *Fantasmagorie* folgt keiner traditionellen Geschichte, es zeigt stattdessen eine Reihe von Transformationen und Bewegungsabläufen und nutzt dafür die Frame-by-Frame Animationstechnik, die auch heutzutage ein grundlegendes Werkzeug der Animation ist. Das Ergebnis war ein zweiminütiger Film, welcher Zuschauer faszinierte und Produzenten dazu veranlasste, Émile mit der Kreation weiterer Filme zu beauftragen [Ben16, S. 31].

In den 1910er Jahren wurde in den USA ein Verfahren entwickelt, welches weltweit zum Standard wurde. Earl Hurd beschrieb in seinem Patent die Idee, Hintergründe auf Papier zu zeichnen, während zu animierende Elemente auf durchsichtigen Celluloidfolien gezeichnet wurden [Cav11, S. 62]. So konnten Hintergründe immer wieder verwendet werden, was Zeit und Kosten sparte.

Das goldene Zeitalter der Cartoons (1928-1951) wurde stark von Walt Disney geprägt. Sein Produktionsstudio revolutionierte auf technischer Seite unter anderem mit der Einführung von synchronisiertem Sound in „*Steamboat Willie*“ (1928) [Ben16, S. 96] und der Erfindung der Multiplan-Kamera, welche eine größere Tiefenwirkung in animierten Szenen ermöglichte. [Ben16, S. 100]. Zum anderen legte Disney einen starken Fokus auf Charakteranimation. In dem Buch „*The Illusion of Life*“ [TJ81, Kapitel 3] werden 12 Animationsprinzipien identifiziert, welche die Disney-Studios nutzten, um ihren Charakteren Persönlichkeit zu geben und authentische Bewegungen zu animieren. Durch diesen Fokus, zusammen mit Storytelling, welches den Zuschauer emotional mitnahm, half Walt Disney dabei mit, Animationen einem breiteren Publikum zugänglich zu machen [Ben16, S. 106].

Disney inspirierte Animatoren weltweit, unter anderem in Japan. Einer der Pioniere japanischer Animation, Tezuka Osamu, wurde stark von Disneys Werken inspiriert und nahm dessen Stilelemente in seinen eigenen Stil auf [Ben17a, S. 106]. Das japanische Animationsstudio Toei, welches in den 1950er Jahren gegründet wurde, startete mit der Ambition, das Disney des Ostens zu werden [Ben17a, S. 86]. Japanische Animation, oder auch Anime, wurde im Laufe der 1960er Jahre immer populärer. Fernsehserien und Filme entstanden, die ab Mitte der 1970er Jahre auch in den Westen exportiert wurden [Ben17a, S. 361]. Dies führte zu einem Austausch zwischen westlicher und östlicher Animation. Heutzutage verschwimmen die Grenzen der Stile und es ist im Gegensatz zu früher schwerer zu erkennen, ob ein Film aus Japan oder Amerika stammt.

Die 2D-Animation ist heutzutage weiter relevant, auch wenn sich 3D-Animation seit den 1990er Jahren, spätestens aber mit der Veröffentlichung von „Toy Story“ (1995) von Pixar, immer größerer Beliebtheit erfreut. Auch diese greifen jedoch auf klassische Techniken und Theorien wie die 12 Charakteranimationsprinzipien von Disney oder die Frame-by-Frame-Animation zurück, der Einfluss von 2D-Animation bleibt also unbestreitbar.

2.2 „Gertie the Dinosaur“ von Winsor McCay

2.2.1 Historischer Kontext



Abbildung 2.2: Winsor McCays Gertie the Dinosaur, 1914 [McC14]

Zu einer Zeit, in der Animation oft nur Bewegung um der Bewegung willen bot, setzten sich Winsor McCays Werke ab, indem sie auf Realismus und Charakteranimation setzten. Er startete als Comic-Autor für Zeitungen, doch selbst diese Werke waren dynamisch und fingen Bewegungen gut ein. [Can05, S. 165]. Charaktere eines seiner Comicstrips namens „Little Nemo in Slumberland“ wurden schließlich zu Hauptfiguren seines ersten Animationsfilms. Er übernahm von Émile Cohl und George B. Kleine, doch innovierte und differenzierte sich auch: Seine Bewegungen waren flüssig und hatten Gewicht, waren also realistischer. Zu diesem Zweck versuchte er auch als erster Animator, seinen Charakteren individuelle Persönlichkeitsmerkmale zu geben. [Can05, S. 168].

Die Produktion von „Gertie the Dinosaur“ startete im Sommer 1913. McCay zeichnete auf Reispapier. Er übernahm die Animation des titelgebenden Dinosauriers, während sein Assistent die immer wieder gleichen Hintergründe im Nachhinein von einer Referenzzeichnung kopierte. Dieser Prozess wurde tausende Male wiederholt [Can05, S. 177], was für die damalige Zeit eine außergewöhnliche Leistung war.

Winsor McCay nutzte das Vaudeville-Theater als Plattform für seine Animationen. Hierbei handelt es sich um eine zu der Zeit beliebte Form der Unterhaltung, welche aus einer Mischung von Kunstformen, unter anderem Musik, Komödie und Tierdressur, bestand. Mit diesem Hintergrund ist es keine Überraschung, dass die Erstvorführung von Gertie the Dinosaur keine einfache Filmvorführung war. Es handelte sich um einen Live-Auftritt, bei dem sich McCay als Zirkusdirektor verkleidete und mit einer Peitsche in der Hand vor eine Leinwand stellte. Als er diese schwang, begann der Film, in dem sich der Diplodocus Gertie mal mehr und mal weniger wie ein gut dressiertes Zirkustier verhielt. Die Videos, welche von dem Animationsfilm heute im Umlauf sind, sind keine Aufnahmen des Vaudeville-Auftrittes, sondern eine von McCay produzierte Version, in der seine Interaktionen mit Gertie durch Live-Action-Aufnahmen und die Dialoge mit Zwischentiteln ersetzt wurden. Diese Version erschien im September 1914 und versicherte, dass Gertie und McCay auch in Städten bekannt wurden, in denen er nicht persönlich auftreten konnte. [Can05, S. 190]

Nicht nur die Zuschauer waren begeistert, auch andere Animatoren nahmen Notiz. Seine Filme wurden weltweit exportiert und inspirierten so ausländische Animatoren, aber auch Amerikaner wie Walt Disney oder die Fleischer Brüder wurden von McCays Aufführung inspiriert [Can05, S. 265]. Er war seiner Zeit weit voraus und setzte Maßstäbe, welche qualitativ erst Jahrzehnte später von den Walt-Disney-Studios wiederholt werden konnten [Can05, S. 265].

2.2.2 Stilmerkmale und technische Innovationen

Stilmittel

McCays Animationsstil, wie auch der Stil seiner Comicstrips, war inspiriert von Art Nouveau. Dieser Stil nutzte fließende, geschwungene Linien, um das Werk natürlich und lebendig wirken zu lassen. Des Weiteren standen die Natur und natürliche Wesen im Vordergrund [Enc24], zu sehen bei Gertie und der Umgebung, in der sie sich wiederfindet. Ein Tal, in dem das Wasser schimmert, Steine und Berge umranden die Landschaft und Gertie steht im Mittelpunkt. Gertie liegt auf ihrer Seite und atmet langsam ein und aus. Hier nutzte McCay eine große Uhr, um zu messen, dass das Einatmen länger dauerte als das Ausatmen. Das Timing der Animation war ihm sehr wichtig und er experimentierte viel [Can05, S. 177]. Art Nouveau war auch eine Antwort auf die industriellen Produktionen, die schnell und günstig hergestellt wurden, dafür aber qualitativ minderwertig waren. Art Nouveau entgegnete mit handgemachten Werken. [Wol12] Dies kann McCays Liebe für die Details erklären und seine Bereitschaft, tausende Zeichnungen anzufertigen, um ein Werk abzuliefern, das am Ende nicht nur ein Produkt ist. „Animation sollte eine Kunst sein, so habe ich es verstanden.“² [Can05, S. 297] sagte McCay 1927 bei einer zu seinen Ehren veranstalteten Dinner-Party. Die Eleganz und Liebe für das Detail von Art Nouveau und McCays Vision für Animation waren wie füreinander gemacht.

Da der Hintergrund immer wieder von McCays Assistenten nachgezeichnet werden musste, scheint es im fertigen Werk so, als würden statische Elemente vibrieren. McCay hatte damit kein Problem, er fand, dass diese Vibration den Hintergrundelementen Leben gab. [Can05, S. 177]

Gertie hatte eine starke Persönlichkeit. Sie weigerte sich, Befehle auszuführen, tanzte, wenn sie sich freute und weinte, wenn McCay sie für ihren Ungehorsam ermahnte. [Can05, S. 21]. Dies, zusammen mit Gerties realistischen Bewegungen und der Kombination des Filmes mit McCays Live-Auftritt [Can05, S. 184], sollte den Zuschauern das Gefühl vermitteln, dass Gertie ein lebendiges Wesen ist. Um dieses Gefühl zu finalisieren, verschwand McCay am Ende des Auftrittes von der Bühne und erschien im Film vor Gertie, die ihn anhob und anschließend davontrug [Can05, S. 185].

Technische Innovationen

McCay nutzte eine Technik, die später Cycling genannt werden würde. Wenn ein Charakter mehrmals ein- und wieder ausatmete, zeichnete er diese Kette von Ereignissen nur einmal auf

²Aus dem Englischen übersetzt mit DeepL Translate, Originaltext: „Animation should be an art. That is how I conceived it“

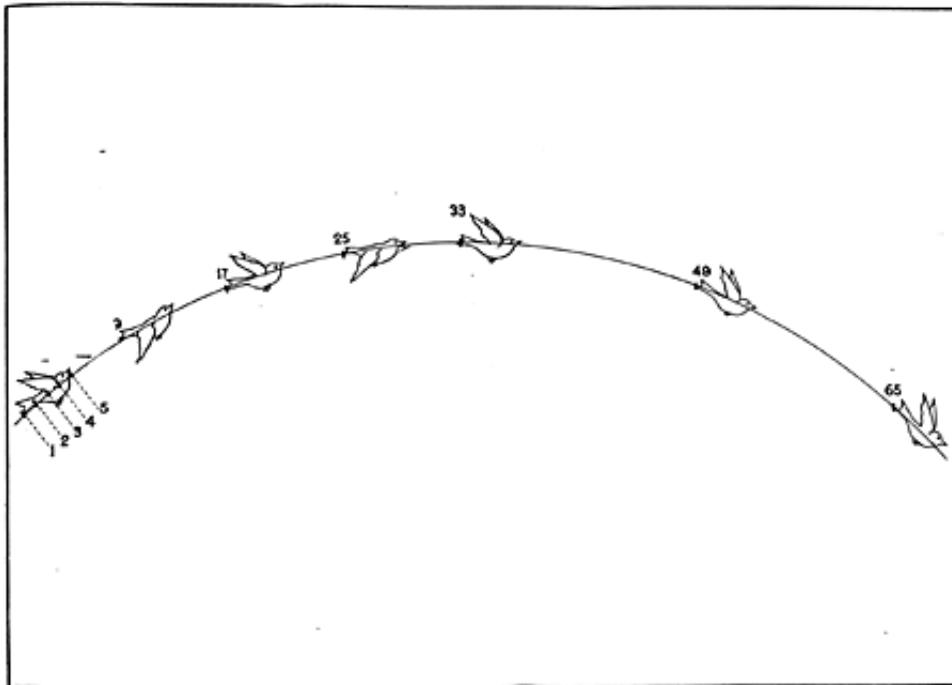


Abbildung 2.3: Beispiel der Verwendung des McCay Split Systems [McC19]

und fotografierte sie beim letztendlichen Zusammensetzen des Filmes einige Male nacheinander. Dies sparte Zeit und damit Geld [Can05, S. 179].

Eine weitere Innovation McCays war das „McCay Split System“, ein Vorläufer der Animationstechnik Inbetweening. Zu dieser Zeit war „Straight-Ahead-Animation“ der Standard, bei dem ein Animationsfilm von Anfang bis Ende durch animiert wurde. Die Funktionsweise von McCays System ist in Abb. 2.3 zu sehen. Zuerst werden ausdrucksstarke Posen zu bestimmten Zeitpunkten gezeichnet, in Abb. 2.3 sind das unter anderem die Frames 25, 33 und 49. Die Flügel des Vogels und dessen Position werden Pose für Pose angepasst und dienen dem Animatoren als „Zielposten“ [Can05, S. 179]. Dabei wird auch auf das Timing geachtet, also wann die Pose wo zu sehen ist. Erst danach werden die verbleibenden Zeichnungen, in 2.3 beispielhaft die Frames zwischen 25 und 33, in denen die Flügel des Vogels sich relativ schnell von einer mittleren Position nach oben bewegen, oder zwischen 33 und 49, in der sich für fast das Doppelte der Zeitspanne von 25 bis 33 fast nichts außer der Position des Vogels verändert, ausgefüllt. Dies hatte mehrere Vorteile: Zum einen half das Konzentrieren auf Schlüsselposen es McCay, starke, erzählerische Posen zu kreieren, die dem Zuschauer im Gedächtnis blieben. Zum anderen bietet diese Art der Animation dem Animator die Möglichkeit, das Timing und den Fluss der Bewegung präzise zu steuern [Can05, S. 179].

Visuelle Artefakte

Sieht man sich heutzutage die Filmversion von „Gertie the Dinosaur“, abgebildet in Abb. 2.2, an, fallen einem einige visuelle Artefakte auf, die typisch für Filme dieser Zeit sind. Sie entstanden aufgrund des damaligen Standes der Technik und des alternden Materials, auf dem sie aufgenommen wurden.

Zum einen gibt es die Vignettierung, durch welche die Ecken des Bildes dunkler als dessen Mitte wirken. Es gibt dafür verschiedene technische Gründe, die mit den damals verwendeten Kameras und deren Linsen zu tun hatten. Natürliche Vignettierung entsteht durch den schrägen Lichtfall auf den Kamerasensor, mechanische Vignettierung entsteht, wenn externe Objekte das Licht blockieren, und optische Vignettierung entsteht durch die Bauweise des Objektivs, bei der der Objektivtubus oder die internen Linsen das Licht abschatten [Com]. Heutzutage sind diese Arten von Vignettierung bei Wahl der richtigen Kamera und des richtigen Objektives sowie für die Situation korrekter Brennweite und Blendenöffnung kaum noch zu sehen. Vignetten können sich durch Abnutzung des Films im Laufe der Zeit verstärken.

Weiterhin leidet das Werk an zufällig wirkenden Veränderungen in der Lichtintensität, die das komplette Bild betreffen. Das sieht man besonders gut am Hintergrund, der weiß sein soll, manchmal aber in das Graue übergeht. Eine Erklärung dafür ist, dass bei der Aufnahme aller gezeichneten Bilder eine Lichtquelle genutzt wurde, die flackerte. Der Film „Little Nemo“ musste aufgrund dieses Problems sogar neu aufgenommen werden [Can05, S. 172].

Ein weiteres beobachtetes Artefakt sind vertikale, dunkle Linien, die zu wandern scheinen und während des gesamten Films zu sehen sind. Dies könnten Abnutzungsspuren sein, die durch das Abspielen des Filmes zustande gekommen sind.

Zuletzt sind schwarze Punkte zu erkennen, welche auftauchen und genauso schnell wieder verschwinden. Hierbei handelt es sich um Schmutz, welcher sich zum Zeitpunkt der Aufnahme auf dem Filmband befand.

2.3 Disney und Mickey Mouse in den 1930er und 1940er Jahren

2.3.1 Fokus auf Charakteranimation

Disney bezeichnete seine Figuren als „gezeichnete Schauspieler“ und ihm war es wichtig, ihnen individuelle Persönlichkeiten zu geben [Ben16, S. 100]. Da es sich hier aber nur um Zeichnungen handelte, mussten diese Persönlichkeiten dem Zuschauer hauptsächlich visuell mitgeteilt

werden, durch ihr Design und die Art und Weise, in der sie sich bewegen. Aus den Erfahrungen der Disney-Studios in den 1930er Jahren wurden die 12 Prinzipien der Animation abgeleitet. Sie werden in dem Buch „The Illusion of Life: Disney Animation“ von den Disney-Animatoren Ollie Johnston und Frank Thomas beschrieben. Hierbei handelt es sich nicht um feste Regeln, die in jeder Situation angewendet werden können. Sie helfen einem lediglich, glaubhafte Charaktere zu animieren [Ben16, S. 109].

Hier aufgelistet ist eine kurze Zusammenfassung der 12 Prinzipien [TJ81, S. 47-70]:

1. Squash and stretch: Objekte können sich, wie im echten Leben auch, verformen. Sie dehnen sich aus und ziehen sich zusammen.

2. Anticipation: Bewegungen sollten nicht aus dem Nichts kommen, der Zuschauer muss darauf vorbereitet werden.

3. Staging: Die Szene sollte so gestaltet sein, dass die Handlung, die Emotionen und die Persönlichkeit der Figur klar und deutlich zum Ausdruck kommen. Der Fokus sollte immer auf einem Element der Szene liegen, nicht auf mehreren gleichzeitig.

4. Straight-ahead action and pose to pose: Dieses Prinzip beschreibt zwei Animationsmethoden. Bei „Straight-ahead“ wird die Animation Bild für Bild erstellt. Da der Animator am Anfang keinen festen Plan für die Szene hat, bleibt der Prozess sehr kreativ. Bei „Pose to Pose“ werden erst zunächst Extremposen gemalt. Erst danach füllt der Animator die Zeichnungen zwischen zwei Extremposen aus. Das gewährt eine Kontrolle über das Timing und den genauen Fluss der Bewegung.

5. Follow-through and overlapping action: „Follow-through“ bedeutet, dass, wenn ein Körper in Bewegung zum Stillstand kommt, sich bestimmte Körperteile aufgrund von Trägheit weiter bewegen können. Wenn jegliche Bewegung auf einmal stoppt, wirkt dies unnatürlich. „Overlapping Action“ bedeutet, dass verschiedene Körperteile nicht gleichzeitig starten und stoppen, deren Bewegungen sich also zeitlich überlappen. Wenn eine Figur läuft, macht der Arm eine Vorwärtsbewegung, während das Bein nach hinten schwingt.

6. Slow in and slow out: In der Natur müssen Objekte beschleunigen, bevor sie ihre Höchstgeschwindigkeit erreichen und bremsen langsam ab, bis sie zum Stillstand kommen.

7. Arcs: Die Bewegungen von Lebewesen folgen meistens runden oder ovalen Formen. So wirken sie weniger steif und natürlicher.

8. Secondary action: Um die Emotion eines Charakters zu vermitteln oder sie besonders hervorzuheben, ist manchmal eine sekundäre Aktion nötig. Dies ist visuell interessanter und es bleibt keine Frage um den Zustand des Charakters offen. Außerdem wird dadurch auch ein Realismus erzeugt, denn wenn ein echter Mensch auf einen starken emotionalen Reiz reagiert, sieht man ihm das am ganzen Körper an und nicht nur an einer Stelle.

9. Timing: Die Anzahl der Zeichnungen bestimmt die Dauer und das Timing einer Aktion in der Animation. Beispielsweise kann die Anzahl an „Inbetweens“ zwischen zwei Zeichnungen, in denen ein Charakter nach rechts und daraufhin nach links schaut, drastisch ändern, wie diese Aktion interpretiert wird. Für schnelle oder komplexe Bewegungen wird in „Ones“ animiert, also einer Zeichnung pro Frame, während für normale Aktionen in „Twos“ animiert wird, also eine Zeichnung alle 2 Frames. Dies spart Arbeit und wirkt oft lebendiger.

10. Exaggeration: Wie der Titel des Buches „The Illusion of Life: Disney Animation“ andeutet, wollen Disneys Animationen nur eine Approximation der Realität liefern und sie nicht perfekt einfangen. Disney wollte eine Karikatur der Realität, aber die Animationen sollten trotzdem überzeugend bleiben. Dies macht jede Aktion klarer und das Endprodukt unterhaltsamer.

11. Solid drawing: Zeichnungen sollten dreidimensional und anatomisch korrekt sein. Die linke und rechte Hälfte eines Charakters sollten einander nicht spiegeln.

12. Appeal: Die zu animierende Figur soll den Betrachter ansprechen und interessant wirken. Charaktere, deren Designs zu kompliziert sind oder die sich unangenehm bewegen, fehlt diese Eigenschaft.

Diese Prinzipien waren es, die Disney-Figuren wie „Mickey Mouse“ oder „Donald Duck“ zu den bekanntesten und beliebtesten Figuren der Filmgeschichte machten. Die Zuschauer fingen natürlicherweise an, sich mit den realistisch wirkenden Karikaturen zu identifizieren und bauten so eine emotionale Bindung auf.

2.3.2 Visueller Stil

Charaktere

Charaktere waren glatt und minimalistisch, umrandet mit schwarzen Linien, um sich besonders vom Hintergrund abzuheben. Die Farben sind einfach und kontrastreich. Abb. 2.4 zeigt, dass der Fokus auf geschwungenen Linien lag, es ist keine einzige gerade Linie zu sehen. Dies liegt an der dynamischen Art und Weise, in der sich die Charaktere bewegen. Disney legte einen großen Fokus auf Charakteranimation, zu sehen an der starken Mimik von Mickey Mouse, die seine Gefühle dem Zuschauer ohne Zweifel vermittelt. All diese Elemente führen zu Charakteren, die ansprechend sind, einem ins Auge fallen und an die man sich leicht erinnern kann. Trotz der Kritik, dass Disneys Charaktere zu süß seien [Ben16, S. 105], änderte Disney nicht den Kurs. Damit beeinflusste er westliche Charakterdesigns bis in die heutige Zeit.



Abbildung 2.4: Screenshot aus einem Mickey Mouse Cartoon von 1937 [37]

Hintergründe

Hintergründe wurden mit einer großen Liebe zum Detail per Hand auf Papier gemalt. Sie lieferten einen großen Einfluss auf die Atmosphäre einer Szene und halfen, ihr den Realismus mitzugeben, den sich Walt Disney wünschte [Ben16, S. 100]. Der Hintergrund in 2.4 würde genauso gut zu „Schneewittchen“ oder „Peter Pan“ passen, also Filmen, deren Charaktere keine anthropomorphen Tiere sind. Hintergründe hatten eine malerische, leicht verschwommene Qualität, die den Fokus den Vordergrundelementen überlassen will.

Licht und Schatten

In 2.4 gibt es keine extremen Licht- oder Schattenkontraste. Des Weiteren ist Mickey frei von Schattierungen, die für den Hintergrund aufgehoben werden. Die Szene ist insgesamt ausgeleuchtet mit diffusem Licht. Lediglich Mickeys Handschuhe haben einen auffälligen, reflektierenden Effekt, um anzudeuten, dass es sich bei dem Material des Handschuhs um Plastik handelt.

2.4 Studio Ghibli und Prinzessin Mononoke

2.4.1 Unterschiede zur westlichen Animation

Studio Ghiblis Filme sind bekannt für tiefgründigere Themen, die sich an ein Publikum aller Altersgruppen richten. Dies steht im Kontrast zur westlichen Animation, die oft ein jüngeres

Publikum als Ziel hat. Ghiblis Filme handeln von Kriegen und stellen sich, wie im Falle von Prinzessin Mononoke, der Frage, welche Rolle der Mensch in der Natur einnimmt. [Ben17b, S. 219]

Auch im Bezug auf Kinematographie gibt es Unterschiede. Animes „strotzen nur so von Kamerafahrten, langen Einstellungen, ausgefallenen Schwenks, ungewöhnlichen Kameraperspektiven und extremen Nahaufnahmen“³ [Cav15, S.16], während in Cartoons eher eine statische oder parallel fahrende Kamera genutzt wird. Im Bezug auf Einstellungsgrößen bleibt es meist bei der herkömmlichen Totale oder einem Medium Shot. Auch findet man in Animes dynamische Hintergründe, die sich im Laufe einer Szene verändern, im Gegensatz zu westlicher Animation, die mit eher statischen Hintergründen arbeitet. [Cav15, S.17]

Es ist zu beachten, dass es sich hier um allgemeine Tendenzen handelt und sich westliche und östliche Animation mit der Zeit voneinander inspirieren haben lassen. Hier soll lediglich ein grober Überblick gegeben werden über dokumentierte und beobachtete Unterschiede.

2.4.2 Visueller Stil



Abbildung 2.5: San und Ashitaka, die Hauptcharaktere von Prinzessin Mononoke [97a]

³Aus dem Englischen übersetzt mit DeepL Translate, Originaltext: „absolutely overflow with tracking shots, long-view establishing shots, fancy pans, unusual point-of-view ‘camera angles’ and extreme close-ups“



Abbildung 2.6: Ashitaka vor einer grünen Gebirgslandschaft, bei dem Versuch, seinen Bogen zu spannen [97b]

Charaktere

Die Klamotten und das Design der Charaktere verraten viel über ihre Persönlichkeiten und Rollen. In 2.5 trägt Ashitaka sehr einfache, funktionale Kleidung, was ihn als Reisenden kennzeichnet. Sans Rock hingegen ist leicht zerrissen, sie trägt ein Stirnband und Farbe im Gesicht, was sie als Kriegerin kennzeichnet.

Die Figuren haben keine für Animes typischen riesigen Augen, stattdessen wirken Gesichter realistischer. Die Mimik spielt hier eine große Rolle, wie man in Abb. 2.6 sehen kann. Obwohl das halbe Gesicht von einem Bogen verdeckt wird, kann man die Entschlossenheit Ashitakas alleine durch seine Augen erkennen. Miyazaki will Mimik nutzen, um den aktuellen Zustand und die emotionale Entwicklung im Laufe der Geschichte zu verdeutlichen [Cav15, S.10].

Hintergründe

Prinzessin Mononoke und andere Studio Ghibli Filme benutzen natürliche, intensive Farben. Figuren, sowie die Landschaft sind detailliert, aber nicht überfüllt mit Details. In 2.5 ist der Hintergrund genauso detailreich wie die Hauptcharaktere und die Tiere, auf denen sie reiten. Der Film bietet gerade am Anfang der Geschichte viele solcher offenen Landschaften, wie auch in 2.6 zu sehen, in denen Grün bis auf die Farbe des Himmels und der Kleidung der Charaktere den ganzen Bildschirm füllt. Dies soll die Schönheit der Natur betonen, welche im Laufe des Filmes von den angreifenden Menschen verunreinigt wird.

Licht und Schatten

Licht und Schatten spielen eine wichtige Rolle dabei, die Atmosphäre der Umgebung widerzuspiegeln. Die extremen Wechsel zwischen Licht und Schatten im Wald von 2.6 lassen ihn dicht und mysteriös wirken, was aufgrund seiner Wichtigkeit im Verlauf der Geschichte nicht verwundert. Diese Szene und 2.5 sind erfüllt von natürlichem Licht, Charaktere werfen leichte Schatten. Des Weiteren haben Gesichter und Klamotten keine komplexen Farbverläufe, stattdessen haben sie eine helle und eine dunkle Seite, was typisch für Cartoons und Animes ist.

Kapitel 3

Animation und Rendering

3.1 Blender als Animations-Framework

3.1.1 Allgemeines

Blender ist eine kostenlose Software, welche „Open Source“ entwickelt wird, d.h., der Quellcode ist frei verfügbar und es ist jedem möglich, ihn zu bearbeiten. Diese Eigenschaft macht Blender zu einem vielseitigen Werkzeug, welches Modellierung, Rigging, Animation und viele weitere nützliche Funktionen unterstützt [Foua]. Blender wird von der Blender Foundation, einer gemeinnützigen Organisation, zur Verfügung gestellt. Das Ziel dieser Organisation ist es laut eigenen Worten, die weltbeste 3D-CG-Technologie Künstlern und Entwicklern kostenlos zur Verfügung zu stellen. [Foub]

Der Wunsch, ein Allesköninger im Bereich 3D zu sein, kann für den unerfahrenen Nutzer auch einen Nachteil darstellen, da die Komplexität von Blender es einem Anfänger schwer macht, sich in der Benutzeroberfläche zurechtzufinden. Es werden einem eine Vielzahl von Modi geboten, die gefüllt sind mit ihren eigenen Menüs und Optionen. Es ist einiges an Einarbeitungszeit nötig, bis man sich damit vertraut gemacht hat. Natürlich wird auch ein leistungsstarker Computer benötigt, um in dreidimensionalen Szenen zu arbeiten. Dies ist allerdings ein Problem, das mit jeder 3D-Software einhergeht.

Viele Menschen, sowohl Anfänger als auch immer mehr Filmstudios, Animationsstudios und Videospiel-Entwickler, nehmen Blender in ihren Workflow auf. Beispiele hierfür sind Ubisoft Film & Television, Rovio Studios und Produktionen von Marvel und DC [Fouc]. Die Arbeit hat sich zum Ziel gesetzt, den Prozess der Umwandlung einer 2D-Szene in eine 3D-Szene zu untersuchen. Deswegen ist es sinnvoll, hier ein Tool in Betracht zu ziehen, welches in der realen

Welt von Entwicklern und Künstlern gleichermaßen benutzt wird, wodurch das gesammelte Wissen einer möglichst großen Menge an Personen von Nutzen ist.

Blender unterstützt eine Vielzahl von Formaten und bietet Import- und Exportmöglichkeiten für alle gängigen Grafikformate, sodass wir bei der Wahl von Tools oder Ressourcen nicht von Blender eingeschränkt werden.

Aufgrund der beschriebenen Vorteile wurde Blender als Animationsframework für dieses Projekt gewählt.

3.1.2 Animation in 3D

Zum Animieren von Charakteren wird in den meisten Fällen „Skelettanimation“ genutzt. Dies ist eine Animationstechnik, welche es dem Animator erlaubt, Modelle durch die Positionierung eines virtuellen Skeletts zu animieren. So lassen sich komplexe Bewegungen realistisch und effizient umsetzen.

Als Basis wird ein 3D-Modell benötigt. In diesem Modell werden dann „Knochen“ platziert, welche aus einem Startpunkt, einem Endpunkt und einer lokalen Transformation, also Translation, Rotation und Skalierung, bestehen. Bei humanoiden Charakteren orientiert man sich hier an einem menschlichen Skelett, siehe Abb. 3.1. Die Knochen werden hierarchisch angeordnet, beispielsweise sind die Fingerknochen in der Hierarchie unter dem Handknochen. Die Transformation eines Knochens beeinflusst alle Knochen, die in der Hierarchie unter ihm stehen. Die Gesamtheit dieser Knochen wird als „Armatur“ bezeichnet.

Anschließend wird die Armatur mit dem Mesh, also der sichtbaren Geometrie des 3D-Modells, verknüpft. Dieser Prozess heißt „Skinning“ und er entscheidet, welche Vertices des Meshnetzes von welchen Knochen beeinflusst werden. Danach kann für jeden Vertex des Modells ein Wert zwischen 0 und 1 gewählt werden, welcher beschreibt, wie stark dieser Vertex von einem bestimmten Knochen beeinflusst wird. Vertices können von mehreren Knochen beeinflusst werden, wodurch natürliche Übergänge zwischen Bereichen entstehen können. Dies wird „Gewichtszuweisung“ genannt.

Blender bietet hierfür auch eine automatisierte Option, welche das Skinning und die Gewichtszuweisung für den Nutzer übernimmt. Das Ergebnis kann daraufhin angepasst werden.

Nun kann mit der Animation begonnen werden. Hier steht es einem frei, ob man Keyframes erstellt und die Knochen zwischen diesen automatisch interpolieren lässt oder ob man in jedem Frame die Armatur manuell manipulieren will. In den meisten Fällen wird eine Kombination der Techniken genutzt, bei der die interpolierten Ergebnisse als Grundlage genommen und dann

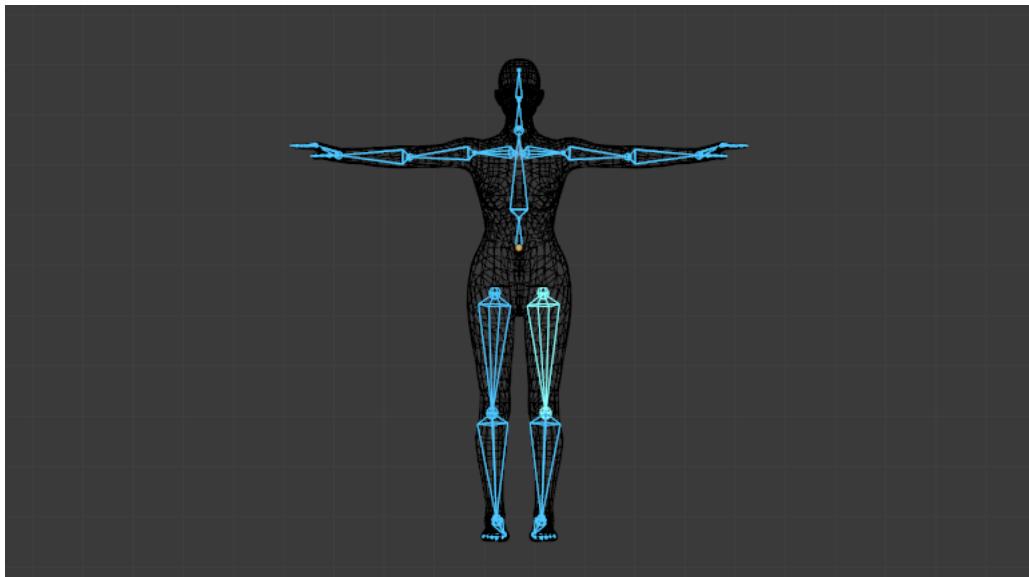


Abbildung 3.1: Humanoide Armatur, erstellt in Blender, zu sehen innerhalb eines durchsichtigen Meshes [Foud]

an die Wünsche des Animators angepasst werden, bis ein zufriedenstellendes Ergebnis erreicht wird.

Die zur Skelettautomation generierten Armaturen sind wiederverwendbar und bieten die Möglichkeit, viele verschiedene Animationen zu realisieren. Zusätzlich kann durch erweiterte Verfahren wie „Inverse Kinematik“ der Animationsprozess erheblich erleichtert werden. Aufgrund dessen ist diese Technik ein Standard für Charakteranimation in Filmen und Videospielen.

3.2 Unity Engine als Rendering-Framework

3.2.1 Shader Pipeline

Shader sind Programme, die auf der GPU ausgeführt werden. In der Computergrafik werden Shader genutzt, um die Darstellung von Oberflächen, Licht, Schatten und anderen visuellen Effekten zu steuern. In Unity werden Shader in der C-ähnlichen Programmiersprache „High Level Shader Language“ (HLSL) geschrieben. Mehrere Arten von Shadern, zum Beispiel Vertex- und Fragment-Shader, sowie alternative Versionen der beiden für andere Arten von Hardware werden in einer Programmiersprache namens ShaderLab in HLSL-Blöcken gespeichert. Dazu können hier Meta-Informationen mitgegeben werden, unter anderem welche Version des Shaders auf welcher Plattform zu laufen hat. Dadurch ist es in Unity möglich, eine gesamte Shaderpipeline für mehrere Systeme in einer einzigen Datei zu beschreiben. Daneben bietet Unity eine visuelle

Programmierumgebung namens „Shader Graph“ an, in der verschiedene „Nodes“ miteinander verbunden werden können, um Shader zu erstellen, ohne sich mit dem Syntax von HLSL oder ShaderLab vertraut zu machen.

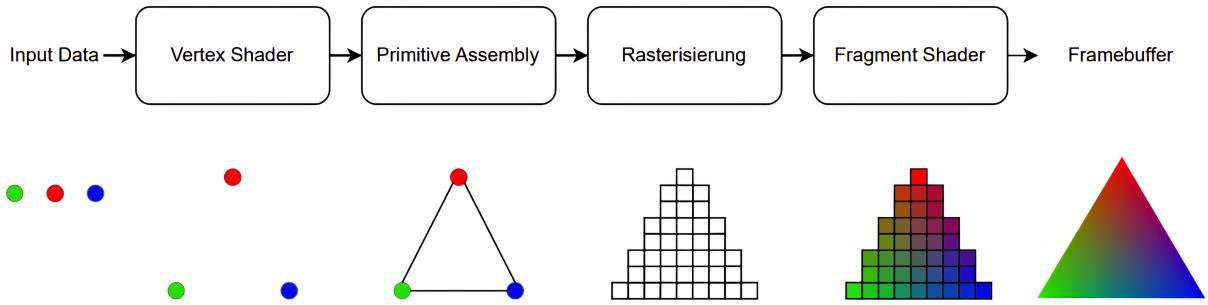


Abbildung 3.2: Eine simple Shader-Pipeline

Hier soll ein kurzer Überblick über die Stufen einer simplen Pipeline (Abb. 3.2) gegeben werden:

Vertex Shader

Ein Vertex ist ein Eckpunkt in einem Mesh. Vertex-Shader arbeiten mit einer Sammlung von Daten, die mit diesem Punkt assoziiert werden. Beispiele für solche Daten sind „World Space Coordinates“ (die Position des Vertex innerhalb einer Szene), Farben oder Normalen. Im Grunde können beliebige Daten übergeben werden. Der Vertex-Shader erhält diese Daten und nimmt Berechnungen an ihnen vor. Auch der Output kann beliebig gewählt werden, eine typische Berechnung ist die Position des Objektes in Kamerakoordinaten, also aus Sicht der Kamera. Die Outputdaten werden an die nächste Pipeline-Stufe weitergegeben.

Primitive Assembly

Primitive Assembly ist ein nicht programmierbarer Bestandteil der Pipeline. Die im Vertex-Shader transformierten Vertices werden zu Primitiven zusammengefügt, also Punkten, Linien und Dreiecken.

Rasterisierung

Rasterisierung ist ein nicht programmierbarer Bestandteil der Pipeline. Die im vorherigen Schritt geschaffenen Primitive werden in „Fragmente“ zerlegt, welche Pixel oder der Teilfläche eines Pixels auf dem Bildschirm entsprechen. Darauf hinaus werden Attribute wie Farben, Normalen und Texturkoordinaten interpoliert, um Werte für jedes Fragment zu finden. Das Resultat wird anschließend dem Fragment-Shader weitergereicht.

Fragment Shader

Der Fragment-Shader erhält die Fragmente und deren interpolierte Werte aus der Rasterisierung. Er führt Berechnungen wie das Anwenden von Texturen, das Berechnen von Licht- und Schatteneffekten oder die Berechnung von Spiegelungs- und Brechungseffekten durch. Am Ende der Berechnungen ergibt sich für jedes Fragment eine Farbe. Das Ergebnis dieser Stufe wird als Framebuffer ausgegeben.

Wenn ein Objekt mit einem Shader gerendert werden soll, muss ein „Material“ erstellt werden. Dieses kann dann dem Objekt zugewiesen werden. Im Shadercode wird dafür beschrieben, dass eine Variable eines bestimmten Datentyps von außen übergeben werden soll. Diese Variable kann dann in einem auf dem Shader basierenden Material ausgewählt werden.

3.2.2 Scriptable Render Pipeline

Da die Szenen, welche wir darstellen möchten, visuell sehr unterschiedlich zueinander sind, ist es wichtig, ein Rendering-Framework zu wählen, welches einfach an die Wünsche des Projektes angepasst werden kann. Zu diesem Zweck bietet die Unity-Engine die „Scriptable Render Pipeline“ (SRP) an.

Während die Shader-Pipeline die Berechnungen auf der GPU beschreibt, ermöglicht die SRP die flexible Organisation einer Menge an Shadern und Renderstufen. Die drei für Unity relevanten Stufen der Render-Pipeline sind laut [Doc]:

Occlusion: Diese Pipelinestufe besteht aus „Frustum Culling“, welches die Elemente außerhalb des Kamera-Frustums vom Rendering ausschließt, und „Occlusion Culling“, welches Objekte hinter anderen Objekten ausschließt, um „Overdraw“, also das mehrmalige Färben eines einzelnen Pixels innerhalb eines Frames, zu minimieren.

Rendering: Optische Eigenschaften aller Objekte in allen aktiven Kameras werden berechnet und in „Render-Targets“ geschrieben. Restriktionen wie die Anzahl der erlaubten Kameras oder Lichtquellen in einer Szene können spezifiziert werden.

Post-Processing: Der in der vorherigen Pipelinestufe generierte Pixel-Buffer wird bearbeitet und schlussendlich auf dem Bildschirm gerendert. Zu Post-Processing gehört unter anderem „Anti-Aliasing“, also das Glätten von Kanten, um Treppeneffekte zu vermeiden, und „Bloom“, welches Lichtquellen leuchtend erscheinen lässt.

Shader spielen in jeder Stufe der SRP eine wichtige Rolle. Beispielsweise werden die im Post-Processing erwähnten Effekte durch Shader implementiert, die im Clip-Space oder Screen-Space operieren.

Neben der Möglichkeit, eine komplett selbst definierte Pipeline zu kreieren, liefert Unity zwei Standard-Implementationen von Scriptable Render Pipelines, welche direkt genutzt oder mithilfe von C#-Code bearbeitet werden können. Zum einen die „High Definition Render Pipeline“, welche sich auf das Rendering möglichst realistischer Szenen konzentriert und als solches auch Raytracing als Feature bietet. Allerdings unterstützt diese Pipeline nur bestimmte Konsole und Desktop-PCs. Daneben existiert die „Universal Render Pipeline“, welche möglichst viele Plattformen unterstützen will, deshalb aber auch weniger grafische Fähigkeiten bietet. Die gewählte Pipeline wird im gesamten Projekt angewendet.

Zu beachten ist, dass die gewählte SRP für das gesamte Projekt gültig ist. Für noch mehr Flexibilität sorgen sogenannte „Render-Passes“, die als „Render-Pass Feature“ für jede Kamera einer Szene einzeln hinzugefügt werden können. Renderpässe rufen einen oder mehrere Shader auf, um das Aussehen einer Szene zu beeinflussen. Ein geschriebener Renderpass kann an voredefinierten Punkten in den Rendering-Prozess injiziert werden, zum Beispiel vor oder nach dem Rendern von Schatten oder dem Post-Processing. Der Output eines Renderpasses ist ein oder mehrere Render-Targets, zum Beispiel Farb- oder Tiefenbuffer oder auch Texturen, welche in späteren Renderpässen weiterverarbeitet werden können.

Diese Pipeline-Architektur erlaubt ein einfaches Zusammenspiel mit Shadern und bietet die Möglichkeit, komplexe visuelle Anforderungen flexibel zu erfüllen und dabei einen Überblick zu behalten.

Kapitel 4

Umsetzung

Für jede Rekreation, die im Laufe der Arbeit erstellt wurde, wurde eine separate, interaktive Szene in einem Unity-Projekt erstellt, welches die Universal Render Pipeline nutzt. Für jede dieser Szenen wurden daraufhin Renderpässe entwickelt, die eigens geschriebene und von Unity angebotene Shader miteinander vereinen.

Die fertigen 3D-Szenen sollen keine bestimmte Szene eines animierten Filmes direkt nachstellen, sondern den generellen Stil und die Atmosphäre von Werken, die zu einer ähnlichen Zeit von einem Produktionsteam kreiert wurden, erfassen. Animationsfilme werden in den meisten Fällen mit 24 Bildern pro Sekunde abgespielt, deswegen werden auch die 3D-Szenen mit 24 Bildern pro Sekunde gerendert.

Das Kapitel umfasst detaillierte Beschreibungen der implementierten Shader, damit diese reproduziert werden können.

4.1 Entwickelte Szenen und Shader

4.1.1 Gertie-Szene

Modellierung und Animation

Mit Hilfe der in Blender angebotenen Sculpting-Tools wurde ein bestehendes Dinosaurier-Modell an die Proportionen von Gertie angepasst. Daraufhin wurde der Charakter geriggt. Im Original brachte Dinosaurier viel Persönlichkeit zum Vorschein und dies sollte innerhalb der 3D-Szene nicht verloren gehen. Deswegen wurde Gerties Tanz, in dem sie ihre Vorderbeine auf- und abhebt, mit Hilfe von Blenders Animationswerkzeugen in die dritte Dimension gebracht.

Wie McCay es mit seinem Split-System vorgemacht hat, wurden zuerst Schlüsselposen aus der Originalanimation nachgebildet (Abb. 4.1). Diese wurden daraufhin jeweils 16 Frames aus-



Abbildung 4.1: Schlüsselposen für Gerties Tanzanimation in Blender

einander in der Timeline platziert. Blender interpoliert die Position der Knochen automatisch zwischen diesen Posen, allerdings stimmen die generierten Frames nicht immer mit der Vorlage überein. Hier wurde manuell nachgeholfen. Das Ergebnis wurde als .FBX-Datei exportiert und in Unity importiert. Hier wurden die nötigen Optionen gewählt, damit die Animation sich wiederholt, deswegen tanzt Gertie in der finalen Szene ununterbrochen.

Szenenaufbau

Die Szene ist an die Landschaft aus dem Film *Gertie the Dinosaur* angelehnt. Gertie tanzt in der Mitte der Szene, um sie herum wurden Steine verschiedener Form und verschiedener Größe platziert. Das Gebiet wird von Bergen umrundet.

Eine Kamera, welche sich frei in der Szene bewegen kann, erlaubt es, die gesamte Szene aus verschiedenen Perspektiven zu betrachten.

Die Originalszene ist zum großen Teil unbelichtet, deswegen wurde allen Elementen in der Szene ein Material gegeben, welches auf einem von Unity bereitgestellten, unbeleuchteten Shader basiert. Der Shader färbt das ganze Modell in einer einzigen Farbe, in diesem Fall ist in dem Material die Farbe Weiß ausgewählt worden.

Daraufhin wurde ein neuer Renderpass angelegt, welcher der Kamera innerhalb dieser Szene hinzugefügt wurde. Der Renderpass besteht aus zwei Shadern, die in der Post-Processing-Phase der Render-Pipeline angewandt werden.

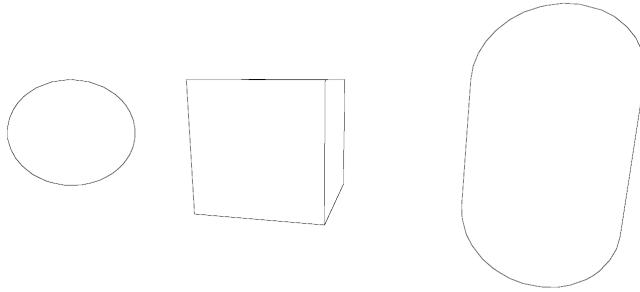


Abbildung 4.2: Erfasste Umrisse verschiedener primitiver Formen mit dem geschriebenen Post-Processing-Shader

Umriss-Shader als Post-Processing-Effekt

Der erste Shader zeichnet basierend auf dem Tiefenpuffer und dem Normalenpuffer der gerenderten Szene, einen Umriss um alle Objekte in der Szene.

Gegeben sind die UV-Koordinate des Fragments UV , die gewünschte Liniенstärke der Umrisslinie $LineThickness$ sowie die $TexelSize$, welche angibt, wie groß ein Texel ist. Ein Texel beschreibt die kleinste Einheit einer Textur, ähnlich wie ein Pixel in einem regulären Bild. Um eine Kante zu erkennen, soll die Bildtextur in einem X-Muster um die aktuelle UV-Koordinate aus dem Fragment abgetastet werden. Hierfür wird zuerst die Länge berechnet, um die UV verschoben werden soll.

$$\begin{aligned} offset_x &= (LineThickness/2) * TexelSize.x \\ offset_y &= (LineThickness/2) * TexelSize.y \end{aligned} \quad (4.1)$$

Daraus werden die UV-Koordinaten der diagonalen Nachbar-Texel errechnet. An diesen Punkten werden der Tiefenpuffer und der Normalenpuffer abgetastet. Dies resultiert in den Punkten $depth_i$ und $normal_i$ für $i \in 0, 1, 2, 3$. Daraufhin findet eine Kantendetektion mithilfe des Roberts-Operators [BHM95] statt, welcher weniger Rechenleistung als andere ähnliche Algorithmen erfordert. Es soll am Ende ein Wert von 1 erhalten werden, falls eine Kante gefunden wurde, und einen Wert von 0, wenn nicht. Dies geschieht für die Tiefenwerte und Normalenwerte. Um diesen Wert zu berechnen, benötigen wir einen den Betrag des Gradienten für das aktuelle Fragment. Gradienten beschreiben die Stärke und die Richtung der stärksten Änderung der Pixelwerte in einer bestimmten Region des Bildes. Zuerst werden die partiellen Ableitungen in x-Richtung G_x und y-Richtung G_y berechnet:

$$\begin{aligned} G_x &= depth_1 - depth_0 \\ G_y &= depth_3 - depth_2 \end{aligned} \quad (4.2)$$

Daraus lässt sich der Betrag des Gradienten $\nabla I(x, y)$ berechnen.

$$\nabla I(x, y) = \sqrt{G_x^2 + G_y^2} \quad (4.3)$$

Es folgt die Fallunterscheidung, welche uns den gewünschten Wert von 0 oder 1 gibt.

$$edgeDepth = \begin{cases} 1, & \text{wenn } \nabla I(x, y) > \text{Schwellenwert, an dem wir eine Kante erkennen möchten} \\ 0, & \text{ansonsten} \end{cases} \quad (4.4)$$

Dies geschieht analog für den Tiefenpuffer, mit einer Ausnahme:

$$\nabla I(x, y) = \sqrt{dot(normalGx, normalGx) + dot(normalGy, normalGy)} \quad (4.5)$$

Wir erkennen das aktuelle Fragment als Umriss an und färben es schwarz, wenn:

$$\max(edgeDepth, edgeNormal) = 1 \quad (4.6)$$

Visuelle Artefakte als Post-Processing-Effekt

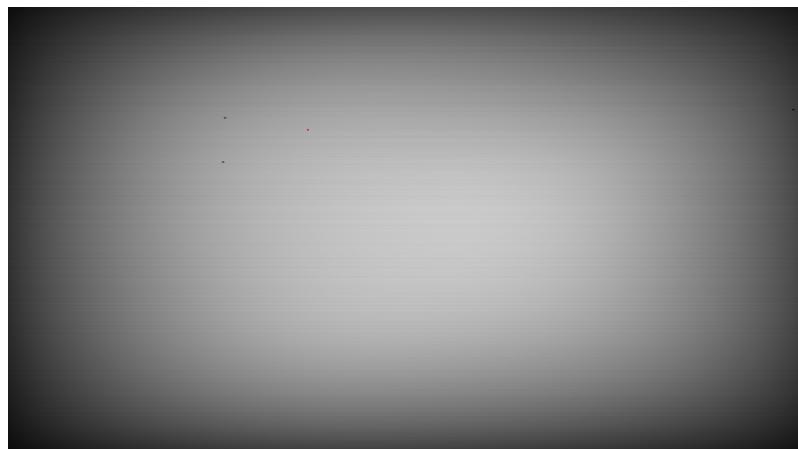


Abbildung 4.3: Der Post-Processing-Effekt simuliert die visuellen Artefakte eines auf einer alten Kamera aufgenommenen Bildes

Um die visuellen Artefakte zu simulieren, die in der Originalanimation zu sehen sind, wurde der Visuelle-Artefakte Post-Processing-Shader geschrieben. Die folgenden Berechnungen finden im Fragment-Shader statt und entstanden durch Experimentierung. Hier haben wir Zugriff auf das bis jetzt gerenderte Bild als Textur *screenTexture* und die UV-Koordinaten *uv* des Fragmentes.

In der Originalanimation ruckelt das gesamte Bild. Um dies zu simulieren, wird in die *screenTexture* mit einem Wert *zoomFactor* hineingezoomt. Wir zoomen immer in die Mitte der Textur, jedoch wird diese Mitte jeden Frame um einen zufälligen Vektor *scaleCenter* verschoben, um das Ruckeln nachzuahmen. So berechnen wir einen Wert *newUV*, an dem wir die *screenTexture* abtasten können.

$$\textit{newUV} = (\textit{uv} - \textit{scaleCenter}) * \textit{zoomFactor} + \textit{scaleCenter} \quad (4.7)$$

Daneben wird die Veränderung der Lichtverhältnisse nachgestellt. Dafür wird jeden Frame eine Simplex-Noise-Textur abgetastet. Die Laufzeit des Programms *t* beeinflusst den neuen uv-Wert, an dem wir abtasten. Der errechnete Wert wird mit *scale* skaliert, um langsamere Änderungen zu erreichen. Simplex-Noise stellt sicher, dass nebeneinanderliegende Pixel in der Textur keine großen Wertunterschiede haben. So wird versichert, dass zwei in aufeinander folgenden Frames abgetastete Werte ähnliche *brightnessModifier* ergeben, damit der Effekt kein wildes Flackern erzeugt. Der *brightnessModifier* wird von der Helligkeit der Fragmentfarbe subtrahiert, indem diese in den HSV-Farbraum übertragen wird.

$$\textit{brightnessModifier} = \textit{sample}(\textit{NoiseTexture}, \textit{vec2}(t, t) / \textit{scale}); \quad (4.8)$$

Mit einer ähnlichen Technik wird ein weiterer Effekt modelliert, welcher die *screenTexture* mit einem *localBrightnessModifier* an verschiedenen Stellen unterschiedlich abdunkelt. Dafür wird beim Abtasten auch *uv* beachtet. *uv* wird verschoben durch zufällige, von *t* abhängige Werte. In *screenTexture* sind Übergänge vom Hellen ins Dunkle mit dieser Technik nicht abrupt.

$$\textit{localBrightnessModifier} = \textit{sample}(\textit{NoiseTexture}, (\textit{uv} + \textit{vec2}(\textit{rand}(t), \textit{rand}(t))) / \textit{scale}); \quad (4.9)$$

Es folgt das Hinzufügen von horizontalen, grauen Linien, die sich über den gesamten gerenderten Bereich spannen. Die Positionen dieser Linien werden jeden Frame neu berechnet. Hier wird nicht darauf geachtet, dass sich die Linien glatt über den Bildschirm bewegen. Die

vollkommene Zufälligkeit der Positionen in jedem Frame hilft dem Effekt. Daneben werden an zufälligen Punkten im Bild alle paar Frames kleine schwarze und weiße Punkte hinzugefügt.

Letztendlich wird Vignettierung simuliert, indem das Bild zu den Rändern hin abgedunkelt wird. Dafür berechnen wir *vignetteUV*, so dass es in der Mitte der *screenTexture* einen höheren Wert hat als am Rand.

$$vignetteUV = uv * (1. - uv); \quad (4.10)$$

X- und Y-Werte von *vignetteUV* werden multipliziert und nach unseren Wünschen skaliert. So entsteht der Wert *vignetteIntensity*, welcher die Intensität der Vignettierung kontrolliert. Der Wert ist in der Bildmitte größer als am Rand. Die Potenz *vignetteRadius* wird auf *vignetteIntensity* angewandt. Bei großen Werten erstreckt sich der Vignette-Effekt weiter in die Mitte, wird aber auch schwächer. Hier muss experimentiert werden, damit ein idealer *vignetteRadius*-Wert gefunden wird. Der sich aus dieser Rechnung ergebende Wert *vignette* wird mit der aus den vorherigen Schritten berechneten Rückgabe-Farbe *tempColor* multipliziert, um die finale Farbe des Fragments *outputColor* zu erhalten.

$$outputColor = tempColor * vignette; \quad (4.11)$$

Modifizierter, unbeleuchteter Shader

Wie in 2.2.2 unter „Stilmittel“ beschrieben wurde, vibrieren die schwarzen Linien in der Originalszene. Auch hierfür wurde ein Shader implementiert, der auf dem von Unity gelieferten unbeleuchtenden Shader basiert. Der Unterschied besteht im Vertex-Shader. Jedes Vertex wird um einen zufälligen dreidimensionalen Vektor verschoben. Dies geschieht jeden Frame. Der Shader findet in der Render-Pipeline vor dem Post-Processing-Schritt statt. Es kommen Veränderungen im Tiefenpuffer und Normalenpuffer zustande, auf denen die Generierung des Umrisses basiert. So kommt eine Unregelmäßigkeit in die zuvor glatten Linien, die dem Original nahekommt.

4.1.2 Disney-Szene

Animationen

Da eines der wichtigsten Elemente von Disneys Animation der Fokus auf Charakteranimation ist, wurde ein Fokus darauf gerichtet, Animationen zu erstellen, welche inspiriert von den 12 Prinzipien der Charakteranimation, beschriebenen in 2.3.1, sind. Hiermit soll untersucht wer-

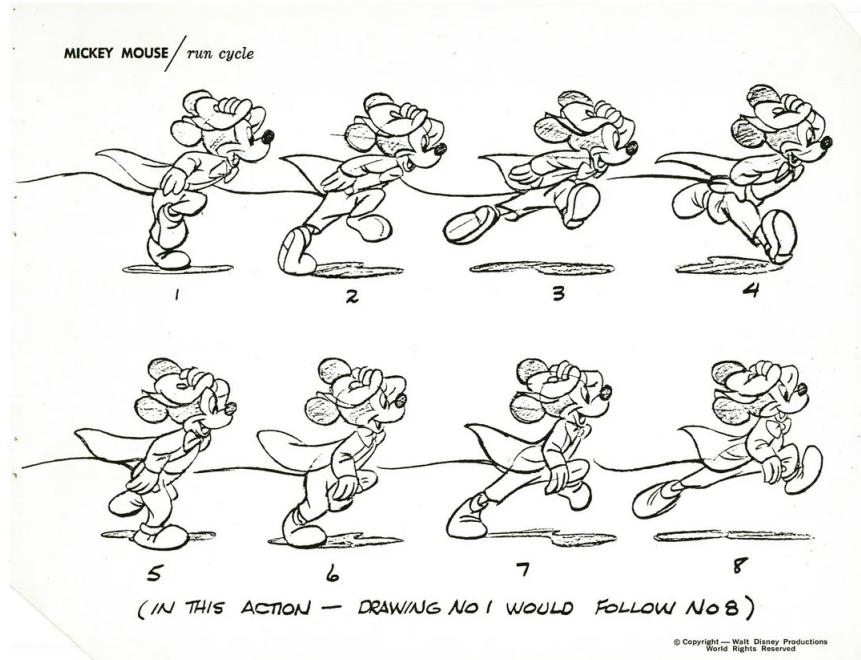


Abbildung 4.4: Frames eines „Run Cycles“ für Mickey Mouse [Pro]

den, ob und wie Computer dabei helfen können, diese Prinzipien umzusetzen und Unterschiede zwischen dem Animieren eines 3D-Modells und dem auf Papier zu erkennen.

Mit Hilfe eines geriggten Modells von Mickey Mouse, welches aus dem Videospiel „Epic Mickey“ stammt, wurden Animationen in Blender erstellt, zwischen deren Frames keine Interpolation stattfindet. Für die erste Animation wurde sich an Abb. 4.4 orientiert, wobei einige der in 2.3.1 beschriebenen Prinzipien angewandt wurden. Dazu gehört der „Appeal“ 2.3.1 (12) von Mickeys einfachem Design. Des Weiteren lässt sich die Bewegung der Arme und Beine als Ellipse beschreiben (2.3.1 (7)). Eine Besonderheit der Übertragung von Abb. 4.4 auf das 3D-Modell ergab sich durch das Prinzip „Squash and Stretch“. Für 2D-Animationen ist es relativ einfach, das Prinzip anzuwenden. Wenn die Beine in einem Frame länger sind als in einem anderen, kann dies im nächsten Bild leicht rückgängig gemacht werden. Dies mit einem 3D-Modell zu imitieren, erfordert etwas mehr Arbeit. In jedem Frame muss das Modell physisch verändert werden, um die gewünschte Verzerrung zu erhalten. Das Mesh muss verzogen werden oder es wird erforderlich, die Haut-Gewichtung des Rigs anzupassen. Im nächsten Frame muss diese Änderung entweder zurückgesetzt oder in einer anderen Art und Weise durchgeführt werden. Dies stellt zusätzlichen Aufwand dar, denn man muss konstant darauf achten, ob man mit einer unveränderten Version des Modells arbeitet, da ansonsten unschöne Artefakte entstehen können.

Eine weitere Animation wurde erstellt, die den ganzen Körper von Mickey Mouse von oben nach unten wippen lässt. Diese Animation dient dazu, dass der Charakter im Stillstand nicht einfach nur starr dasteht, was irritierend wirken könnte. Zu allen Zeiten soll der „Appeal“ 2.3.1 (12) des Charakters beibehalten werden.

Es wurde ein Schrank (Abb. 4.5 (i)) animiert, der sich von alleine öffnet. Dabei schießt das Fach zuerst aus dem Schrank und wird viel länger, als der Innenraum des Schrankes es erlaubt. Danach zieht er sich auf eine realistischere Länge zurück. Diese absurde Animation nutzt die Prinzipien „Timing“ 2.3.1 (9) und „Exaggeration“ 2.3.1 (10). Das Fach schießt mit einer solchen Geschwindigkeit hervor, dass es die Regeln der Realität bricht, bremst dann schlagartig ab und kehrt dadurch in seinen natürlichen Zustand zurück. Auch hier hat die 2D-Animation den Vorteil. In Blender musste mit der Skalierung des Faches gespielt werden, um den Effekt erfolgreich durchzuführen. Dabei muss darauf geachtet werden, dass die neue Größe des Faches nicht dazu führt, dass es durch die Rückseite des Schrankes geht. In solchen Situationen wird der Mehraufwand von 3D-Animation deutlich.



Abbildung 4.5: Animierte und interaktive Elemente in der 3D-Szene, Schrank (i), Auto (ii) und Flasche (iii)

Daneben wurden Animationen innerhalb der Unity-Engine geschaffen:

Es wurde ein fahrendes Auto (Abb. 4.5 (ii)) animiert, welches „Slow in and slow out“ (2.3.1 (6)) verdeutlichen soll. Durch sogenannte „Animation Curves“ können Beschleunigung und Abbremsung einfach umgesetzt werden. Die automatisch generierte Animationskurve, welche die Geschwindigkeit des Autos beschreibt, ist eine diagonale Linie. Um dem Animationsprinzip

gerecht zu werden, wurde diese gerade Linie ersetzt durch eine von Robert Penner als „easeInOutSine“ betitelte Übergangsfunktion [Pen02, S. 215]. So wirkt die Bewegung natürlicher.

Daneben wurde das Animationsprinzip „Squash and Stretch“, siehe 2.3.1 (1), durch eine hüpfende Colaflasche (4.5 (iii)) verdeutlicht. Die Flasche wird erst immer kleiner und dicker, während sie Energie aufbaut, was den Zuschauer auf den Sprung vorbereitet (ein Beispiel für „Anticipation“, siehe 2.3.1 (2)). Daraufhin schnellt sie nach oben, durch die Geschwindigkeit wird sie lang und dünn. Als sie auf dem Boden landet, wobbelt sie etwas, um die Wucht des Aufpralls zu kompensieren. Die Bewegung ist inspiriert von der Animation eines hüpfenden Balles. Bei „Squash and Stretch“ ist es wichtig, dass das zu animierende Objekt sein Volumen behält, während es sich verformt. Auch hier ist eine 3D-Animationssoftware hilfreich, da sie uns neben Position und Rotation eines Objektes auch dessen Skalierung zeigt. Diese kann für jeden Keyframe so berechnet werden, dass sie dem Volumen des Objektes vor der Verformung entspricht. Daneben können „Timing“, siehe 2.3.1 (9) und „Slow in and slow out“, siehe 2.3.1 (6), wieder durch Animationskurven beeinflusst werden.

Hierdurch werden die positiven Aspekte der Animation durch 3D-Animationssoftware dargestellt. Durch Funktionen wie Animationskurven und Rigging lassen sich die 12 Prinzipien der Charakteranimation erfolgreich in 3D umsetzen. Oftmals ist dies sogar simpler als in handgezeichnete 2D-Animation. Durch das einfache Ändern von Variablen wie der Position, Rotation und Skalierung von Objekten in Keyframes, die automatische Generierung von Inbetweens und die Nutzung von Animationskurven, welche Änderungen im Bezug auf Timing einfach ermöglichen, muss der Animator nicht den genauen Ablauf der Animation planen. Er kann experimentieren und iterieren, bis die Animation die ideale Beschaffenheit hat.

Szenenaufbau

Mit Hilfe der in dem vorherigen Schritt animierten Objekte sowie anderer 3D-Modelle wurde eine Szene entwickelt, die den Innenraum einer Wohnung darstellt.

Daraufhin wurde ein Objekt erstellt, welches Mickey Mouse in der Welt darstellt. Es wurde ein Script geschrieben, welches das Objekt mit Hilfe der Pfeiltasten bewegt. Dafür wird die Position des Objektes angepasst und ein fester Wert wird jeden Frame addiert oder subtrahiert, je nach Bewegungsrichtung. In diesem Beispiel bewegt sich Mickey nur auf einer zweidimensionalen Ebene. Steht er still, spielt eine „Idle“-Animation, ansonsten die „Run“-Animation, welche je nach Richtung der Bewegung gespiegelt wird. Dem Objekt wurde ein unsichtbarer „Capsule Collider“ hinzugefügt. Berührt dieser kapselartige Bereich eines der interaktiven Elemente der Szene, so starten dessen Animationen.

Die interaktiven Elemente aus Abb. 4.5 wurden an passenden Stellen innerhalb der Szene platziert. Hierbei wurde darauf geachtet, dass die Positionen für die Objekte logisch sind und der vom Betrachter gesteuerte Mickey sie erreichen kann, sowie dass die gespielten Animationen immer im Bild bleiben.

Kameras und implementierte Shader

Für die Szene wurden 2 Kameras in der Szene platziert. Beide Kameras nutzen eine orthographische Kameraperspektive, es tritt also keine perspektivische Verzerrung auf. Diese Projektion wurde der perspektivischen Projektion vorgezogen, weil sie dem Look klassischer Cartoons näher kommt. Dort bewegen sich Objekte nahe der Kamera nicht langsamer als Objekte im Hintergrund, was auch bei der orthographischen Projektion der Fall ist.

Sei ein 3D-Punkt (x, y, z) gegeben, welcher auf eine 2D-Ebene projiziert wird als (x', y', z') , so gilt:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

Um den Punkt in kartesischen Koordinaten in das Kamerakoordinatensystem zu transformieren, wird der Punkt mit folgender Matrix multipliziert:

$$P_{ortho} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Beide Kameras besitzen die gleichen Near- und Far-Clipping-Planes und rendern den gleichen Bereich der Welt. Sie folgen dem Mickey-Mouse-Objekt.

Unterschiede finden sich in den Objekten, welche die beiden Kameras rendern und in den ihnen zugeteilten Renderpass-Features.

Kamera 1 rendert Objekte, die zum „Hintergrund“ gehören, welche in einem alten Animationsfilm also auf einem Blatt Papier gemalt wurden, um immer wieder verwendet werden zu können und sich ohne perspektivische Verzerrung mit der Kamera bewegen.

Zuerst wurde der in 4.1.1 beschriebene Shader in einen separaten Renderpass gelegt. Das entsprechende Renderpass-Feature wurde daraufhin der Kamera hinzugefügt, um allen Elementen der Szene einen schwarzen Umriss zu geben.

Daraufhin wurden zwei Renderpässe kreiert, welche den Eindruck erwecken sollen, dass die Gesamtheit des Kamerabildes einem gemalten Bild nahe kommt.

Renderpass 1 besteht aus einem Shader, welcher den Kuwahara-Filter [KKD10] implementiert. Dies ist ein nichtlinearer Glättungsfilter, was bedeutet, dass neue Pixelwerte nicht aus linearen Kombinationen der umliegenden Pixel errechnet werden, sondern lokale Eigenschaften wie Varianz oder Median berücksichtigt werden. Das Ziel des Filters ist die Glättung des Bildes mit Erhaltung der Kanten. Für jeden Pixel wird ein quadratisches Fenster der Größe $2a + 1$ als Filterkern genutzt. Dieses Fenster wird in 4 einander überlappende Quadranten geteilt. Für jeden Quadranten Q_i mit $i \in 1, 2, 3, 4$ werden Mittelwert μ_i und Varianz σ_i^2 der Pixel innerhalb des Quadranten berechnet. Der Filter wählt den Mittelwert μ_k aus dem Quadranten mit der kleinsten Varianz, also des homogensten Quadranten, als neuen Pixelwert. Dies wird in [KKD10] formuliert als:

$$\mu_k := m_i, i = \operatorname{argmin}_k \sigma_k \quad (4.12)$$

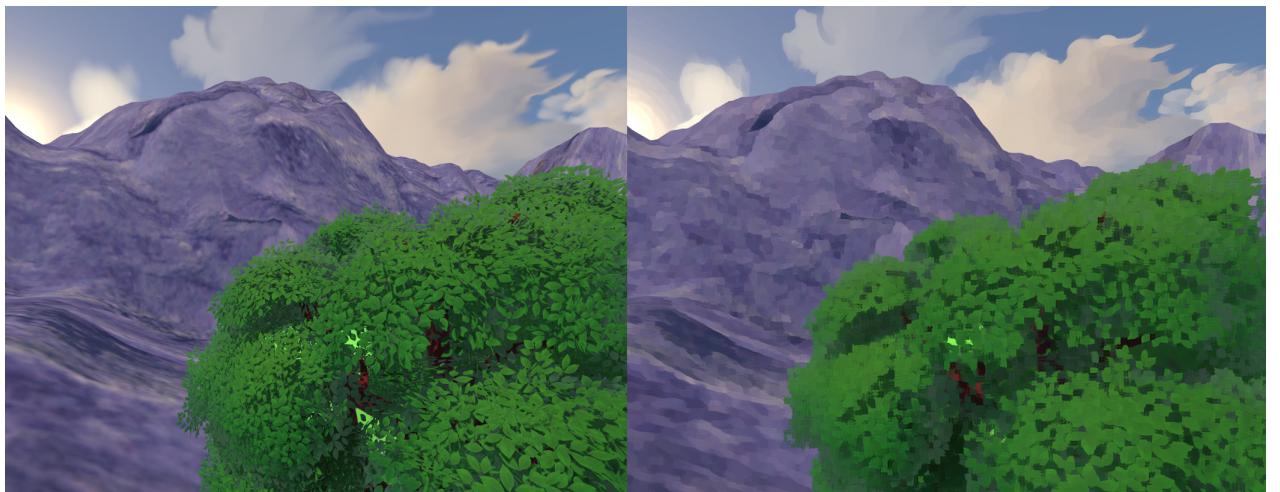


Abbildung 4.6: Beispielszene links vor Anwendung des Kuwahara-Filters (i) und rechts danach (ii).

Wie in Abb. 4.6 zu sehen, hat die Szene einen stilisierten Eindruck bekommen. Details gehen verloren, zu sehen bei den Blättern des Baumes, die Kante zwischen den verschiedenen Elementen des Bildes bleibt jedoch erhalten. Des Weiteren entsteht ein Verlust der Tiefenwirkung, da subtile Helligkeitsunterschiede, welche sonst die Tiefe andeuten, durch einheitliche Flächen ersetzt werden. In homogenen Bereichen von (i) entstehen in (ii) blockartige Segmente, welche für den malerischen Stil zwar von Vorteil sind, allerdings auch durch ihre ähnliche Form etwas unnatürlich wirken.

Renderpass 2 besteht aus zwei Post-Processing-Shadern. Um den in 2.3.2 beschriebenen gemalten, verschwommenen Effekt zu erzielen, wurde ein „Gaussian Blur“-Shader eingesetzt, welcher von Unity stammt. Dies soll die scharfen, blockartigen Artefakte des Kuwahara-Filters reduzieren und gibt uns die gewünschte, leicht verschwommene Qualität von alten Disney-Hintergründen. Der zweite Shader legt eine Textur über das gesamte Bild. Diese *waterColorTexture* hat ein unregelmäßiges Muster in Grautönen und soll dem Ergebnis das Aussehen eines mit Wasserfarben gemalten Bildes geben. Die Textur „klebt“ nicht an der Kamera, stattdessen wurde sie in einem absoluten Weltkoordinatensystem platziert. Dafür wurde die absolute Position im Weltkoordinatensystem *worldPosition* des Fragmentes mithilfe der Fragment-*clipPosition* berechnet.

$$\text{worldPosition} = \text{clipPosition} * \text{InverseViewMatrix} * \text{InverseProjectionMatrix} \quad (4.13)$$

Da die Szene orthografisch auf der YZ-Ebene dargestellt wird, tasten wir *waterColorTexture* an *worldPosition.yz* ab und multiplizieren das Ergebnis mit der im Fragment berechneten Farbe.

Kamera 2 rendert Objekte, welche animiert werden. Diese Trennung soll den Prozess der Cel-Animation imitieren, welcher zu dieser Zeit Stand der Technik war. Die auf Cels animierten Elemente hatten meist ein einfacheres Auftreten, die Schattierung wurde deutlich simpler gehalten. Hierfür wurde ein einfacher Cartoon-Shader geschrieben. Im Fragment-Shader wird dafür der Kosinus des Winkels zwischen der Lichtrichtung *L* und der normalisierten Oberflächen-Normale *N* im Weltkoordinatensystem berechnet.

$$NdotL = \text{dot}(N, L) \quad (4.14)$$

Wenn $NdotL > 0$ ist, wird die Oberfläche des Modells beleuchtet, ansonsten liegt sie im Schatten. Die Variable *lightIntensity* wird entsprechend auf 1 für beleuchtet oder 0 für unbeleuchtet gesetzt. Daraufhin wird die Farbe der Umgebungsbeleuchtung *ambientColor* berechnet. So kann mit der dem Fragment mitgegebenen interpolierten Farbe *color* und einer durch das Material mitgegebenen Farbe *baseColor* schließlich ein Output errechnet werden:

$$\text{outputColor} = \text{color} * \text{baseColor} * (\text{lightIntensity} + \text{ambientColor}) \quad (4.15)$$

Des Weiteren wurde Kamera 2 das Umriss-Render-Pass-Feature hinzugefügt. Umrisse sind zwar auch im Hintergrund erhalten, dort aber aufgrund des Blur-Effektes nicht so prominent sichtbar. Auch dies dient dazu, das Original zu mimen.

4.1.3 Ghibli-Szene

Animation

Diese Szene beinhaltet nur ein animiertes Element. Es handelt sich hierbei um ein 3D-Modell eines Menschen im Anime-Stil. Für diese bereits gerigchte Figur wurde eine Laufanimation in „Twos“ animiert, eine neue Pose wird also erst nach 2 Frames angezeigt. Dies ist eine Kosten-sparmaßnahme vieler Animes, die somit imitiert wurde.

Szenenaufbau

Die animierte Figur wurde in der Mitte der Welt platziert. Daraufhin wurden Skripte programmiert, um die Bewegung innerhalb einer 3D-Szene zu ermöglichen sowie Kamerakontrolle zu erlauben. Die Laufanimation spielt, wenn der Charakter sich mitbewegt. Es wurde sich hier für eine voll kontrollierbare Kamera entschieden, um die in 2.4.1 beschriebenen kinematografischen Unterschiede im Gegensatz zu der restriktiven Kamera von Cartoons fühlbar zu machen. Auch das Freiheitsgefühl, welches ein Bestandteil der Philosophie von Studio-Ghibli-Filmen ist, wird somit zum Ausdruck gebracht.

Die Szene sollte den starken Naturfokus von Studio-Ghibli-Werken hervorheben. In diesem Fall wurde sich für eine von dem Film „Prinzessin Mononoke“ inspirierte Szene entschieden, siehe Abb. 2.5 und 2.6. Dazu gehören ein blauer, bewölkter Himmel und eine Umgebung voller Gras und Bäume. Dazu wurde sich entschieden, die begehbarer Umgebung mit Bergen zu umrunden, was nicht nur die Figur davon abhält, von der Welt zu fallen, sondern der Szene auch einen natürlichen, nicht erzwungenen Endpunkt bietet.

Wieder wurden zwei Kameras genutzt, um die Szene zu realisieren. Die erste Kamera nimmt nur die spielbare Figur auf, welche mit dem Umriss-Shader noch einmal stärker von ihrer Umwelt abgehoben wird. Die zweite Kamera rendert die restliche Umgebung. Der Kamera wurde ein Renderpass hinzugefügt, welcher eine Erweiterung des in 4.1.2 beschriebenen Kuwahara-Filters darstellt.

Anisotroper Kuwahara-Filter

Der in 4.1.2 beschriebene Kuwahara-Filter ist aufgrund der blockartigen Artefakte, die trotz des Gaussian Blurs nicht ganz verloren gehen, nicht für die Studio-Ghibli-inspirierte Szene ausreichend. Besonders die Blätter der Bäume leiden unter den Artefakten, sie wirken unnatürlich, siehe Abb. 4.6. Da für diese Szene ein Fokus auf der Natur als Umgebung steht, ist dies nicht zu verkraften. Deswegen wurde eine erweiterte Variante, der anisotrope Kuwahara-Filter implementiert. Auch dieser soll eine natürlichere, stilisierte Szene erzeugen, allerdings frei von blockartigen Artefakten. Der Hauptunterschied des anisotropen Kuwahara-Filters besteht darin, dass der um den aktuellen Pixel gelegte Filter nicht mehr quadratisch ist. Stattdessen kann er seine Form an die Beschaffenheit des Bildes anpassen (Abb. 4.7). Existiert im Original eine Kante, so verformt sich der Filter, bis er so viel der Kante wie möglich zur Berechnung der neuen Pixelfarbe nutzt. Dadurch werden die Formen des Inputs besser übernommen. Die genutzte Implementierung und die Formeln stammen aus [KKD10].

Zur Berechnung des Bildes werden zuerst eine Reihe von Filterkernen auf den Input angewandt. Das Ergebnis dieser Schritte wird als Textur gespeichert, die später der Input für den anisotropen Kuwahara-Filter ist.

Zuerst wird ein „Strukturtensor“ mithilfe der horizontalen und vertikalen Faltungskerne des Sobel-Filters aufgebaut.

$$S_x = 1/4 \begin{pmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{pmatrix}$$

$$S_y = 1/4 \begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Die durch Faltung des Bildes mit diesen Kernen entstehenden f_x und f_y , die partiellen Ableitungen von f , dem gesuchten Strukturtensor.

$$g_{ij} = \begin{pmatrix} f_x \cdot f_x & f_x \cdot f_y \\ f_x \cdot f_y & f_y \cdot f_y \end{pmatrix} =: \begin{pmatrix} E & F \\ F & G \end{pmatrix} \quad (4.16)$$

Der Strukturtensor kann dazu genutzt werden, Kanten zu erkennen, indem die Eigenwerte und Eigenvektoren der Matrix analysiert werden. Dafür wird er zuerst mit einem Gauß-Filter geglättet, um Rauschen zu reduzieren und eine robustere Schätzung der lokalen Bildstruktur zu

erhalten. Danach werden lokale Orientierung und Anisotropie aus den Eigenwerten und Eigenvektoren des Strukturtensors berechnet.

Mit Hilfe von 4.16 werden Eigenwerte $\lambda_{1,2}$, Eigenvektor in Richtung der minimalen Veränderungsrate t , lokale Orientierung φ und Anisotropie A berechnet. Diese werden als Input für den anisotropen Kuwahara-Filter genutzt.

$$t = \begin{pmatrix} \lambda_1 - E \\ -F \end{pmatrix} \quad (4.17)$$

$$A = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \quad (4.18)$$

$$\varphi = \arg t \quad (4.19)$$

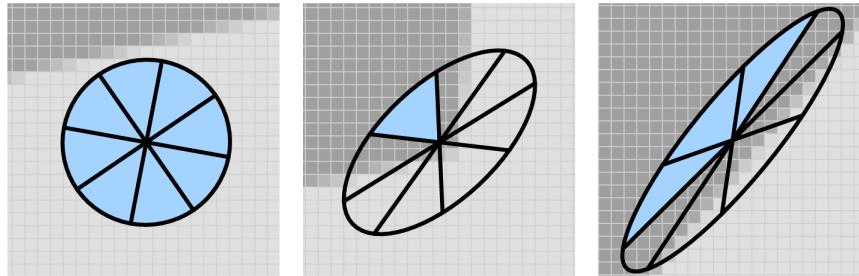


Abbildung 4.7: Der Kernel passt sich an die Beschaffenheit des Bildes an. [Kyp10]

Lokale Orientierung und Anisotropie entscheiden über die Form des Filters. In homogenen Bereichen ist der Filter kreisförmig, in anisotropen dagegen eine Ellipse, deren Hauptachse an der Hauptrichtung der Bildmerkmale ausgerichtet ist, siehe Abb. 4.7. Für den Radius r und $\alpha > 0$ berechnen wir die Achsen a und b der Ellipse.

$$a = \frac{\alpha + A}{\alpha} r, \quad b = \frac{\alpha}{\alpha + A} r, \quad (4.20)$$

$$(4.21)$$

Die Ellipse wird in Sektoren unterteilt und mit SR_φ auf einen Einheitskreis abgebildet. Dies vereinfacht die Berechnungen und garantiert konsistente Verarbeitung, das Ergebnis wird also weniger anfällig für numerische Fehler. S und R_φ werden definiert als:

$$S = \begin{pmatrix} \frac{1}{2a} & 0 \\ 0 & \frac{1}{2b} \end{pmatrix}, \quad R_\varphi = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix} \quad (4.22)$$

Für jeden der Bereiche wird nun eine Gewichtsfunktion definiert, die angibt, wie stark ein Pixel den Sektor beeinflusst. Hier nutzen wir nicht die in [KKD10] definierten Gewichtsfunktionen. Stattdessen werden die aus [Kyp10] verwendet. Diese erfordern keine Faltung und können deswegen effizient in Echtzeit berechnet werden. Für den ersten Sektor ist die Gewichtungsfunktion als Polynom definiert, andere entstehen durch Rotation.

$$Q_{k0}(x, y) = \begin{cases} (x + \eta - \zeta y^2)^2 & \text{wenn } x^2 + y^2 \leq r^2 \\ 0 & \text{sonst} \end{cases} \quad (4.23)$$

$$Q_{ki} = Q_{k0} \circ R_{2\pi i/N} \quad (4.24)$$

Danach werden für jeden Sektor die gewichteten lokalen Mittelwerte m_i und die quadrierten Standardabweichungen σ_i^2 berechnet. Das Ergebnis des Filters wird als gewichtete Summe der lokalen Mittelwerte der Sektoren definiert, homogene Sektoren werden dabei stärker gewichtet.

$$F(x_0, y_0) = \frac{\sum_{i=1}^N \alpha_i m_i}{\sum_{i=1}^N \alpha_i}, \quad \alpha_i = \frac{1}{1 + (255 \cdot \sigma_i^2)^{q/2}} \quad (4.25)$$

Um diesen Filter weiter auf die Bedürfnisse der Szene anzupassen, wird auch der Tiefenpuffer in Betracht gezogen. Je weiter entfernt das Fragment von der Kamera ist, desto stärker wirkt der Filter. Dies hat zur Folge, dass Elemente mehr Details verlieren, je weiter sie von der Kamera entfernt sind.

Kapitel 5

Evaluation

5.0 Umgebungsparameter

Die Szenen wurden auf einem System mit folgenden Komponenten entwickelt: 32 GB RAM, AMD Ryzen 5 5600 6-Core CPU, Nvidia RTX 2080 GPU. Die Szenen werden in einer Auflösung von 1920 x 1080 Pixeln gerendert, wenn nicht anders angegeben. Tabelle 5.1 zeigt eine Auflistung der genutzten Shader.

5.1 Authentizität und visuelle Unterschiede der 2D-Inspiration und ihrer 3D-Nachbildung

5.1.1 Gertie-Szene

Visueller Stil

Die Nachbildung des visuellen Stils ist größtenteils gelungen. Sie hält sich an einfarbige Materialien für den Dinosaurier und die Umgebung, während dessen Umrisse in tiefem Schwarz gezeichnet wurden. Wie im Original vibrieren die Umrisse, was besonders in Bewegung einen authentischen, handgezeichneten Effekt bietet.

Daneben wird die Atmosphäre der Filmversion mit Hilfe der Simulation visueller Artefakte eingefangen. Dazu gehören die bewusste Einführung von Verschiebungen in jedem Frame, welche das Ruckeln der frühen Filmaufnahmen nachahmt. Daneben werden mit Hilfe von Noise-Funktionen Lichtschwankungen authentisch simuliert. Auch die Vignettierung hilft dabei, den Betrachter in die Zeit zurückzuversetzen. Dies ist auch ein Beispiel dafür, wie sich historisch bedingte technologische Einschränkungen als kreative Mittel in der Nachbildung nutzen lassen.

Szene	Kamera	Shader
Gertie	1	Umriss Shader Wackelnde Vertices Shader Visuelle Artefakte Shader
Disney	1	Umriss Shader Cartoon Shader Kuwahara Filter Gaussian Blur Shader Texture Overlay Shader
	2	Umriss Shader Cartoon Shader
Ghibli	1	Umriss Shader Cartoon Shader
	2	Anisotroper Kuwahara Filter

Tabelle 5.1: Genutzte Shader sortiert nach Szene und Kamera innerhalb der Szene

Dennoch gibt es einige Schwächen, welche die Gesamtwirkung beeinträchtigen. Die Szene im Original wirkt insgesamt in sich schlüssiger, was an McCays Liebe zum Detail liegt. Elemente wie das schimmernde Wasser und die organische Anordnung der Steine um Gertie herum konnten nicht eingefangen werden. Die Umgebung hat an Charakter verloren, wirkt willkürlicher und repräsentiert nicht die in 2.2 beschriebene Kunstform „Art Nouveau“, welche genau diese Liebe zum Detail zelebriert.

Daneben gibt es Unstimmigkeiten in Form der Schatten. Die Schatten unter Gerties Füßen beim Tanzen haben nicht die Form ihrer Füße, sondern sind simple Kreise. Unrealistischerweise sieht man die Schatten, wenn ihr Fuß auf dem Boden aufliegt. Einige Schattenbereiche wurden nicht eingefangen, dazu gehören die unter Gerties Füßen und die an den zum Wasser zeigenden Klippen.

Animation

Die gewählte Tanzanimation wurde mit Schlüsselposen und Interpolation in Blender umgesetzt, wobei McCays „Pose-to-Pose“-Ansatz erfolgreich nachgeahmt wurde. Die Animation greift Gerties im Original hervorgehobene verspielte Persönlichkeit auf. Die Übernahme der Schlüsselposen half dabei, die Bewegungen des Tanzes authentisch zu übernehmen.



Abbildung 5.1: Finale Nachbildung von Winsor McCays „Gertie the Dinosaur“

Zusammenfassung

Die Nachbildung von Winsor McCays „Gertie the Dinosaur“ innerhalb von Unity ist insgesamt ein gelungenes Beispiel des Renderings klassischer handgezeichneter Zeichentrickanimation. Die Essenz des Originals wird sowohl visuell als auch in Bezug auf die Darstellung der titelgebenden Figur erfolgreich bewahrt. Die Implementierung der visuellen Artefakte wie Vignettierung und das Flackern geben einem den Eindruck, in eine virtuelle Welt einzutauchen, welche aus dem Jahr 1914 stammen könnte.

Daneben zeigt die Nachbildung die Möglichkeiten neuer Technologien wie Blender, historische Animationsprozesse zu adaptieren. Es wird verdeutlicht, wie alte Techniken, wie das McCay-Split-System, mit moderner Software verfeinert und erweitert wurden.

Dennoch gibt es einige Schwächen, wie die Detailgenauigkeit der Umgebung. Diese halten die Nachbildung davon ab, das Original in allen Aspekten zu replizieren.

5.1.2 Disney-Szene

Visueller Stil

Obwohl einige Elemente von Disney klassischem Stil übernommen werden konnten, schafft es die Nachbildung nicht, den visuellen Stil ausreichend zu replizieren.



Abbildung 5.2: Finale Nachbildung von Disneys Stil der 1930er und 1940er Jahre

Zwar helfen die schwarzen Umrandungen und kontrastreichen Farben sowie der leicht verschwommene Hintergrund dabei, Elemente wie Mickey Mouse und die hüpfende Flasche vom Hintergrund abzuheben.

Negativ zu bemerken sind jedoch die Hintergründe selbst. In der Nachbildung wirken sie eher schlicht und funktional. Ihnen fehlt die malerische Qualität der Hintergründe des Originals. Auch ist der Hintergrund weniger detailliert, was letztendlich zu einem Verlust der Atmosphäre führt.

Auch Licht und Schatten wurden nicht genügend Aufmerksamkeit geschenkt. Zwar werfen die Objekte lange, dramatische Schatten wie in den Originalen, es mangelt aber vor allem bei den Hintergrundobjekten an überzeugendem Shading.

Animation

Im Gegensatz dazu wurde sich viele Mühe gegeben, Prinzipien der Charakteranimation innerhalb der Szene einzubauen. Mehrere der 12 Prinzipien der Animation wurden in verschiedene Animationen eingebaut.

Mickeys „Run“-Animation wurde direkt von Disney übernommen und ist als solche originalgetreu. Daneben stellen weitere animierte Gegenstände andere Prinzipien der Animation vor, was Ihnen viel Persönlichkeit gibt und den Eindruck steigert, es handele sich um eine spielbare Animation aus dem goldenen Zeitalter der Animation.

Auch die Art und Weise, in der die Kamera Mickey verfolgt, während er durch die Szene läuft, ist passend für die Animationen der Zeit.

Zusammenfassung

Der Fokus der Nachbildung von Disneys Stil wurde auf die Aufarbeitung der Animationen gelegt. Diese sind überzeugend übernommen worden und zeigen, dass klassische Animationsprinzipien in den meisten Fällen sehr gut mit Hilfe neuer technologischer Möglichkeiten adaptiert werden können und wie hilfreich diese dabei sind.

Die Animationen sind zwar ein wichtiger und ikonischer Teil von Disneys Erbe, allerdings können sie die fehlende Liebe zum Detail in Bezug auf den Hintergrund nicht ganz entschädigen. Hier hätte es mehr Zeit gebraucht, um mehr mit Shading und Texturen zu experimentieren und näher an den visuellen Stil zu kommen.

5.1.3 Ghibli-Szene



Abbildung 5.3: Finale Nachbildung von Studio Ghiblis Stil

Visueller Stil

Die als Inspiration gewählte Szene beeindruckt durch die realistische Darstellung der Natur, die malerische Qualität, die intensiven Farben und die Harmonie zwischen Charakteren und Landschaft. Die Nachbildung setzt viele der gesehenen Elemente gelungen um:

Die gewählten Farben sind natürlich und ausdrucksstark. Üppige Vegetation, weitläufige Wiesen und Berge im Hintergrund zeigen den Fokus auf die Natur. Die Atmosphäre des Originals konnte gut nachgestellt werden.

Auch die malerische Qualität der Hintergründe wurde überzeugend umgesetzt. Durch den verwendeten Filter hat die gesamte Umgebung einen stilisierten Eindruck. Genau wie in den echten Hintergründen von Studio Ghibli verlieren die am weitesten von der Kamera entfernten Elemente, die Berge und der Himmel, fast komplett an Details.

Nur zum Teil übernommen wurden dagegen die dynamischen Lichteffekte, die im Original den Wald so mystisch machen. Zwar werden Schatten realistisch geworfen, sie haben jedoch nicht genug Kontrast und Komplexität, um an das Original heranzukommen.

Des Weiteren wirkt die Figur leicht abgehoben von ihrer Umgebung. Dies liegt an der unpassenden Kleidung, die farblich und stilistisch nicht so gut passt wie die zerrissenen Klamotten im Original.

Ein weiterer Unterschied liegt darin, wie Licht und Schatten mit dem Charakter interagieren. Betrachtet man das Gesicht der Charaktere in 2.5, so stellt man fest, dass die Gesichter klar in zwei Seiten aufgeteilt werden, eine dunkle und eine helle. Der verwendete Cartoon-Shader nutzt die Normalen der Vertices, um diese Unterscheidung zu treffen. Allerdings können die Normalen in den Modellen eines Gesichts im Normalfall in einem Bereich in viele Richtungen zeigen, zum Beispiel in der Augenhöhle. Hier fehlt die Feinkontrolle, um zu entscheiden, welcher Bereich komplett im Schatten liegen soll.

Animation

Die Laufanimation des Charakters ist flüssig und natürlich, jedoch nicht so ausdrucksstark wie ähnliche Animationen aus dem Original. Besonders Mimik und Körpersprache könnten weiter ausgebaut werden.

Daneben wurde sich dazu entschieden, statt der dynamischen Kameraposen eine frei kontrollierbare Kamera aus der dritten Person zu nutzen. So wird es ermöglicht, den nachgebildeten Fokus auf die Natur zu erforschen. Die damit erreichbaren Kameraeinstellungen kommen aber nicht an die Dramatik und Stimmung der für jede Szene manuell platzierten Kamera des Originals heran.

Zusammenfassung

Diese Nachbildung steht im Gegensatz zu den Disney-Werken, bei denen der Fokus auf den Animationen lag. Stattdessen konnten hier die Atmosphäre und visuelle Merkmale gut übernommen werden, auch wenn die Komplexität von Licht und Schatten an manchen Stellen nicht ausreichend ist und der Charakter visuell nicht ins Bild passt. Es hätte mehr Fokus auf Gestik oder Mimik gelegt werden müssen, um die Ausdrucksstärke von Studio Ghiblis Charakteren zu erreichen. Trotzdem konnte der Stil des Originals zufriedenstellend übernommen werden.

Bemerkung: Auf dem Testsystem führt der Versuch, in 24 FPS zu rendern, zu einer extrem hohen CPU-Framezeit. Dies hat zur Folge, dass die Zahl der FPS einbricht. Dieser Bug betrifft nur die Ghibli-Szene und tritt auf anderen Systemen und auf dem Testsystem ohne die Limitierung auf 24 FPS nicht auf.

5.2 Rechenzeit der Post-Processing-Shader

Im folgenden Kapitel sollen die Rechenzeiten der nachgebildeten Szenen untersucht werden. Es wurden 3 identische Szenen aufgebaut, welche jeweils die Kameras aus den Nachbildungen nutzen. Einige der Szenen rendern zwei Kameras. In diesen Fällen wird vom Worst-Case-Szenario ausgegangen, in dem beide Kameras die gesamte Szene rendern müssen. Die Testszenen bestehen aus 4,6 Millionen Vertices und haben eine geringe visuelle Komplexität. Sie zeigen lediglich eine Ansammlung von 3D-Objekten, welche mit einem einfachen, roten Material ohne Beleuchtung dargestellt werden.

Für jede Szene wurde eine Basismessung vorgenommen, welche die Renderzeit der Szene ohne zusätzliche Post-Processing-Effekte erfasst. Um die effektive Renderzeit der Post-Processing-Shader zu ermitteln, wird die Basismessung von der gemessenen Renderzeit der Shader abgezogen. Für Szenen mit zwei Kameras wurden separate Basismessungen pro Kamera und eine Messung mit beiden aktiven Kameras durchgeführt, um die Ergebnisse korrekt berechnen zu können. Zu beachten ist, dass die Renderzeit ohne Post-Processing-Shader für jede Testszene neu gemessen wurde. Dies liegt daran, dass zur Implementierung der Shader jede Kamera unterschiedliche Renderpässe nutzt, ohne welche die gewünschten Shader nicht in Unity implementiert werden können. Durch eine Basismessung pro Szene wird sichergestellt, dass die Ergebnisse nicht durch Unity-spezifische Faktoren verfälscht werden.

Es soll untersucht werden, wie gut die Shader sich für Echtzeit-Rendering eignen. Dabei wird ein Fokus auf Rendering mit 24 Frames pro Sekunde gelegt, was die Vorgabe für die in

Renderzeit (RZ) ohne Post-Processing-Shader in ms: 1,80 ms

Kamera	RZ Kamera in ms	Shader	RZ Shader in ms	Eff. RZ in ms
1	2,62			0,82
		Umriss-Shader	2,31	0,51
		Visuelle-Artefakte-Shader	2,47	0,67

Tabelle 5.2: Gemessene Renderzeiten der Post-Processing-Shader der Gertie-Szene

der Bachelorarbeit beschriebene Situation ist. Daneben wird besprochen, ob sich die Shader für Echtzeit-Rendering mit 60 FPS eignen.

5.2.1 Rechenzeit der Gertie-Szene

Hier werden die in der Gertie-Szene genutzten Post-Processing-Shader untersucht. Es wird erwartet, dass der Umriss-Shader langsamer ist als der Visuelle-Artefakte-Shader, da er pro Fragment 11 Texturen abtasten muss. Solche Texture-Lookups werden oft als performancekritisch betrachtet. Die Messergebnisse sind zu finden in Tabelle 5.2.

Analyse der Shader-Performance

Wie in 5.2 zu sehen, sind die genutzten Shader mit jeweils ungefähr 0,5 ms Laufzeit gut für Echtzeit-Rendering nutzbar. Für 60 FPS nehmen sie pro Shader gerade mal 3% der verfügbaren Zeit in Anspruch. Für unseren Verwendungszweck mit 24 FPS liegen wir sogar unter 1%.

Der Umriss-Shader arbeitet etwas effizienter als der Visuelle-Artefakt-Shader. Dies scheint zunächst überraschend, betrachtet man allerdings die Anzahl an mathematischen Operationen, welche die beiden Shader ausführen müssen, wird das Bild klarer: Der Visuelle-Artefakte-Shader muss für ein Fragment 194 mathematische Operationen durchführen, das liegt an der Vielzahl der zu berechnenden, zufälligen Werte. Diese werden benötigt, damit sich das Ergebnis glaubhaft willkürlich verhält. Doch genau diese Berechnungen sind arithmetisch aufwendig. Der Umriss-Shader nimmt dagegen nur 34 mathematische Operationen vor. Außerdem haben moderne Grafikkarten sogenannte Textureinheiten (TMU), die unter anderem aus Texture Addressing Units (TAU) bestehen. Diese machen Texture-Lookups weniger teuer und beeinflussen die Performance der Shader dadurch weniger als anfangs vermutet.

5.2.2 Rechenzeit der Disney-Szene

Hier werden die in der Disney-Szene genutzten Post-Processing-Shader untersucht. Die Messergebnisse sind zu finden in Tabelle 5.3.

Renderzeit (RZ) ohne Post-Processing-Shader in ms. 2 Kameras: 1,88, Kam. 1: 1,21, Kam. 2: 1,20

Kamera	RZ Kamera in ms	Shader	RZ Shader in ms	Eff. RZ in ms
1+2	16,54			14,66
1	14,93			13,72
		Kuwahara Filter	12,07	10,86
		Umriss-Shader	1,57	0,36
		Gaussian Blur	4,28	3,07
		Step Function	1,36	0,15
		Texture Overlay	1,38	0,17
2	1,58			0,38
		Umriss-Shader	1,58	0,38

Tabelle 5.3: Gemessene Renderzeiten der Post-Processing-Shader der Disney-Szene

Analyse der Shader-Performance

Der teuerste Shader in dieser Szene ist mit Abstand der Kuwahara-Filter. Er nimmt mit 10,86 ms von 13,72 ms etwa 79% der Renderzeit in Anspruch.

Er ist also durchaus für Echtzeit-Rendering mit 24 FPS geeignet, ist aber gleichzeitig die erste Anlaufstelle für Optimierungen, sollte es letztendlich knapp mit der Zeit werden. Der Shader kann nicht für ein Echtzeit-Rendering mit 60 FPS empfohlen werden. Ein Shader sollte nicht mehr als die Hälfte der verfügbaren Zeit in Anspruch nehmen. Es ist unwahrscheinlich, dass weitere Aufgaben, wie Lichtberechnungen, in den verbleibenden 6 ms erledigt werden können. Für diesen Zweck müssten also Optimierungen vor allem im Kuwahara-Algorithmus vorgenommen werden.

Die anderen Shader sind günstig und nehmen jeweils effektiv unter 1 ms in Anspruch. Dies macht sie perfekt für die Nutzung im Echtzeit-Rendering.

5.2.3 Rechenzeit der Ghibli-Szene

Hier werden die in der Ghibli-Szene genutzten Post-Processing-Shader untersucht. Es soll besonders der anisotrope Kuwahara-Filter (AKF) untersucht werden, da dem Umriss-Shader schon in der Analyse der Rechenzeit der Gertie-Szene Aufmerksamkeit gewidmet wurde. Der AKF benötigt zur Berechnung die Ergebnisse einiger anderer Post-Processing-Shader. Da diese Shaders als Input mit dem Output der vorangegangenen Shader arbeiten, ist es nicht möglich, diese einzeln zu testen. Deswegen werden hier akkumulierte Werte gemessen. Durch die Differenz der nacheinander durchgeführten Messungen lassen sich die effektiven Renderzeiten für einen einzelnen Shader akkurat berechnen. Die Messergebnisse sind zu finden in Tabelle 5.4.

RZ ohne Post-Processing-Shader in ms: 2 Kameras: 2,32, Kam. 1: 1,65, Kam. 2: 1,21

Kamera	RZ Kamera in ms	Shader	RZ Shader in ms	Eff. RZ in ms
1+2	21,79			19,47
1	21,00			19,35
		Anis. Kuwa. Filter	21,00	19,35
		->Ber. Struktur Tensor	2,01 (akk.)	0,36
		->Ber. Gaussian Blur	2,55 (akk.)	0,54
		->Ber. Lok. Orient. & Anis.	2,57 (akk.)	0,02
		->Ber. AKF	12,24 (akk.)	9,67
		->Ber. AKF	21,00 (akk.)	8,76
2	1,60			0,39
		Umriss Shader	1,60	0,39

Tabelle 5.4: Gemessene Renderzeiten der Post-Processing-Shader der Ghibli-Szene

Analysen der Shader-Performance

Aus Tabelle 5.4 ist zu erkennen, dass die Post-Processing-Effekte die Renderzeit der Szene dominieren. Die Gesamtrenderzeit der Szene beträgt 21,79 ms, davon sind 19,47 die effektiven Kosten für das Post-Processing. Die für diese Szene genutzten Post-Processing-Effekte sind nur bedingt dafür geeignet, in Echtzeit mit 24 Frames pro Sekunde gerendert zu werden. Sie nehmen fast 50% der zur Verfügung stehenden Renderzeit in Anspruch. Es muss also unbedingt darauf geachtet werden, wie teuer andere Berechnungen der Szene sind, um zu garantieren, dass es nicht zu Einbrüchen der Performance kommt. Auch um 60 Frames pro Sekunde zu rendern, lässt sich diese Kombination an Shadern also im aktuellen Zustand nicht nutzen. Denkbar wäre, den anisotropen Kuwahara-Filter nur einmal über das Bild laufen zu lassen. Somit würden 8,76 ms gespart werden. Die gesamte Renderzeit wäre mit 10,71 ms aber weiter sehr hoch und die Qualität des Ergebnisses würde durch das einmalige Anwenden des Filters leiden. Es müsste hier also abgewogen werden, ob die visuelle Qualität in diesem Fall ausreichend ist. Selbst dann bleiben nur noch ca. 6 ms für weitere Berechnungen übrig. Die Szene müsste also sehr einfach aufgebaut sein. Es gibt nur wenige Situationen, in denen diese Kombination an Shadern für Echtzeit-Rendering empfohlen werden kann.

5.2.4 Durchgeführte Experimente

Hier soll untersucht werden, ob es Situationen gibt, in denen die Shader mehr oder weniger Renderzeit benötigen, als in anderen. In jedem Test wird ein anderer Fokus gelegt.

Renderzeit (RZ) ohne Post-Processing-Shader in ms: 2,90

Kamera	RZ Kamera in ms	Shader	RZ Shader in ms	Eff. RZ in ms
1	3,73			0,83
		Umriss-Shader	3,50	0,60
		Visuelle-Artefakte-Shader	3,52	0,62

Tabelle 5.5: Gemessene Renderzeiten einer visuell komplexeren Szene unter Nutzung der Post-Processing-Shader der Gertie-Szene

Experiment 1

Für das erste Experiment wurde eine neue Szene kreiert, sie stellt eine große Innenstadt voller Wolkenkratzer dar. Sie besteht aus der gleichen Anzahl von Vertices wie die gewohnte Testszene. Die Materialien der Szene basieren auf dem standardmäßigen, belichteten Shader, den Unity mit der URP zur Verfügung stellt. Diese Szene soll eine Situation aus der echten Welt darstellen, in der Szenen visuell viel komplexer sind. Das Ziel des Experimentes ist es, herauszufinden, ob die visuelle Komplexität der Szene einen Einfluss auf die Rechenzeit der Post-Processing-Shader hat.

Die Messergebnisse in Tabelle 5.5 zeigen, dass die Renderzeit der getesteten Post-Processing-Shader nicht durch die visuelle Komplexität der Szene beeinflusst wird. Da die Anzahl der Fragmente pro Szene gleich bleibt, muss der Shader für jeden Frame die gleiche Anzahl an Berechnungen durchführen. Die Framezeiten unterscheiden sich deswegen nur um einige 10 µs von denen in Tabelle 5.2, wobei es sich wahrscheinlich um Messungenauigkeiten handelt.

Dies bedeutet jedoch nicht, dass die visuelle Beschaffenheit einer Szene in keinem Fall Einfluss auf die Renderzeit hat. Sollte ein Shader Branches nutzen, um die Berechnung von bestimmten Fragmenten vorzeitig zu beenden, könnte dies einen enormen Unterschied machen. Man stelle sich einen Greenscreen-Shader vor, welcher nur mit einem Fragment arbeitet, wenn dieses eine grüne Fragmentfarbe mitgeliefert bekommt. In diesem Fall kann die Beschaffenheit der Szene einen großen Unterschied in der Performance machen.

Experiment 2

Für Experiment zwei wurde die Renderauflösung erhöht. Statt in Full HD rendern wir nun in QHD, das sind 2560 x 1440 Pixel. Dadurch wurde die Anzahl der Fragmente auf das 1,78-Fache erhöht. Nun soll erforscht werden, ob die Rechenzeit der Post-Processing-Shader linear mit der Menge der Fragmente steigt.

Die Messungen in Tabelle 5.6 zeigen, dass die Renderzeit der Shader nicht linear mit der Menge der Fragmente steigt. Betrachten wir die effektive Renderzeit der Post-Processing-Shader

RZ ohne Post-Processing-Shader in ms: 2 Kameras: 2,97, Kam. 1: 1,38, Kam. 2: 1,35

Kamera	RZ Kamera in ms	Shader	RZ Shader in ms	Eff. RZ in ms
1+2	34,41			31,44
1	32,96			31,58
		Anis. Kuwa. Filter	32,96	31,58
		->Ber. Struktur Tensor	2,80 (akk)	1,42
		->Ber. Gaussian Blur	3,75 (akk)	0,95
		->Ber. Lok. Orient. & Anis.	3,77 (akk)	0,02
		->Ber. AKF	20,24 (akk)	16,67
		->Ber. AKF	32,96 (akk)	12,72
2	2,02			0,64
		Umriss Shader	2,02	0,67

Tabelle 5.6: Gemessene Renderzeiten einer Testszene in einer Auflösung von 2560 x 1440 Pixeln unter Nutzung der Post-Processing-Shader der Gertie-Szene

in Tabelle 5.4 mit 19,47 ms für die gesamte Szene und denselben Wert aus Tabelle 5.6 mit 31,44 ms, so können wir den tatsächlichen Skalierungsfaktor für die in der Ghibli-Szene genutzten Shader berechnen:

$$\frac{31,44ms}{19,47ms} \approx 1,61$$

Der Wert liegt unter dem erwarteten Faktor von 1,78. Dafür gibt es ein paar mögliche Gründe:

Eigenschaften der genutzten Shader: Der für den anisotropen Kuwahara-Filter implementierte Shader beinhaltet Schleifen, welche in homogenen Bereichen vorzeitig aus der Schleife ausbrechen. Bei höherer Auflösung gibt es entsprechend mehr Fragmente, die in solchen Bereichen liegen. Die Anzahl der Schleifeniterationen steigt also nicht proportional mit der Auflösung.

GPU-Caching: Moderne GPUs nutzen Caches, um häufig benutzte Daten schneller verfügbar zu machen. Der Shader muss zwar öfter Texturen abtasten, dies bietet aber eine bessere Chance, dass ein abzutastender Wert gerade im Cache liegt. Höhere Auflösungen erhöhen die Cache-Trefferquote, da die Dichte benachbarter Texel mit steigender Auflösung zunimmt.

GPU-Parallelisierung: GPUs arbeiten parallel, die genutzte GPU hat Beispielsweise 2944 CUDA-Kerne. Geht man davon aus, dass bei der Full-HD-Szene die Kerne nicht voll ausgelastet waren, bei der Berechnung der QHD-Szene allerdings schon, so haben die Kerne in diesem Fall effizienter gearbeitet. Durch die bessere Verteilung sinkt die durchschnittliche Renderzeit pro Pixel.

Kapitel 6

Zusammenfassung

Die Bachelorarbeit untersucht die Herausforderungen und Potentiale moderner Rendering-Techniken zur Nachbildung klassischer handgezeichneter Zeichentrickanimation. Ziel ist es, ein umfassendes Verständnis der stilistischen und technischen Anforderungen zu entwickeln, die eine authentische Übertragung von Animationen in die dritte Dimension ermöglichen. Dabei soll der Charme klassischer Animationen behalten werden, um emotionale und ästhetische Erfahrungen zu schaffen. Zu diesem Zweck werden ikonische Werke aus verschiedenen Epochen betrachtet, darunter fallen Winsor McCays „Gertie the Dinosaur“, Disney-Animationen der 1930er und 1940er Jahre und Produktionen des Studio Ghibli. Hier soll letztendlich die Atmosphäre dieser Werke in einem dreidimensionalen Kontext eingefangen werden.

Zuerst wurden die gewählten Beispiele auf den historischen Kontext und die stilistischen Merkmale untersucht. „Gertie the Dinosaur“ zeichnet sich dabei durch seine von „Art Nouveau“ geprägte Liebe zum Detail aus, aber auch die realistische Tiefenwirkung der Animation und die authentischen Bewegungen und das natürliche Verhalten des Hauptcharakters machten das Werk zu einem Liebling von Kritikern und Zuschauern. Die Liebe zur Charakteranimation, zu sehen bei McCays Werk, wird in Disneys Werken noch mehr in den Fokus gerückt. Ein wichtiger Bestandteil der Analyse sind die 12 Prinzipien der Charakteranimation, die sich im goldenen Animationszeitalter herausgebildet haben. Studio Ghibli integriert detailreiche Hintergründe, erwachsene Themen und komplexe Charaktere, um visuell und narrativ beeindruckende Werke zu schaffen.

Danach wurden moderne Tools vorgestellt, welche notwendig sind, um die Rekreation in die Tat umzusetzen. Hierzu gehört Blender, welches zur Modellierung und Animation der Charaktere genutzt wurde. Darüber hinaus wird ein kurzer Einblick in die Funktionsweise der Skelettautomation gewährt. Anschließend wird Unity als Rendering-Framework vorgestellt. Das Konzept der Scriptable Render Pipeline bietet eine ideale Umgebung, um Shader in den verschiedenen

Stufen des Rendering-Prozesses einzusetzen und somit die gewünschten Stilmittel so akkurat wie möglich nachzubilden.

Es wurden 3 Szenen innerhalb von Unity erstellt, eine für jede der drei Szenen.

Dabei wurde chronologisch vorgegangen. Als Erstes wurde die Szene für die Nachbildung von „Gertie the Dinosaur“ aufgebaut. Hierfür wurde ein 3D-Modell für Gertie in Blender entworfen. Erst wurde ein Rig erstellt. Eine auf Gerties Tanz im Original basierende Animation wurde daraufhin animiert, in Anlehnung an das McCay-Split-System wurden erst Schlüsselposen kreiert und danach die fehlenden Zwischenposen durch automatische Interpolation der Schlüsselposen. Die Animation soll Gerties starke Persönlichkeit zeigen. Die Szene wurde auf Basis der Umgebung im Originalfilm gestaltet. Jedes Objekt hat ein unbeleuchtetes, weißes Material zugewiesen bekommen. Der dazugehörige Shader verschiebt die Vertices der Modelle jeden Frame zufällig, um das Zittern der Linien im Original nach zu bilden. Es wurde ein Renderpass programmiert, welcher aus zwei Shadern besteht. Der erste Shader generiert anhand des Tiefenpuffers und Normalenpuffers schwarze Umrisse für die Objekte in der Szene. Der zweite Shader bildet die visuellen Artefakte eines aus dieser Zeit stammenden Filmes wieder, beispielsweise Vignettierung.

Die nächste Szene basiert auf den alten Disney-Filmen der 1930er und 1940er Jahre. Hier wurde ein Fokus auf Animationen gelegt, welche einige von Disneys 12 Prinzipien der Animation umsetzen. Besonderer Fokus lag hierbei auf einem 3D-Modell von Mickey Maus. Es wurden für ihn zwei Animationen erstellt. Die erste war eine Renn-Animation, diese wurde direkt aus Disneys Archiven übernommen. Hierdurch wurde vor allem das Prinzip „Squash and Stretch“ in einem dreidimensionalen Raum betrachtet. Danach wurde eine Steh- oder „Idle“ Animation entworfen, damit Mickey, auch wenn er stehen bleibt, das Prinzip des „Appeals“ bewahrt. Daneben wurden weitere Objekte animiert. Es wurde eine Szene aufgebaut, welche das Innere einer Wohnung zeigt. Die animierten Objekte wurden darin platziert. Wenn der kontrollierbare Mickey an ihnen vorbeiläuft, beginnen deren Animationen. Es wurden zwei Kameras in der Szene platziert, die orthographisch projizieren und den gleichen Bereich rendern, allerdings andere Objekte. Die erste Kamera rendert Hintergrundelemente mithilfe des Kuwahara-Filters und soll einen stilisierten, gemalten Eindruck vermitteln. Der Filter betrachtet einen quadratischen Teil um den aktuellen Pixel, teilt diesen in vier Quadranten auf und wählt als Ergebnis den Mittelwert des Quadranten mit der niedrigsten Varianz. In homogenen Bereichen entstehen dadurch quadratische Artefakte. Die zweite Kamera rendert animierte Elemente mit einem unbelichteten Cartoon-Shader. Diese Zwei-Kamera-Technik ist von Cel-Animation inspiriert, die Hintergründe und animierte Objekte ähnlich voneinander trennt, um sie wiederverwendbar zu machen. In unserem Fall dient die Technik dazu, die animierten Objekte in den Fokus zu setzen.

Die letzte Szene basiert auf Studio Ghiblis Werken. Es wurde eine von der in „Prinzessin Mononoke“ zu sehenden Umgebung inspirierte Szene erstellt. Zu sehen ist ein von Bergen umrandeter Wald, dessen Boden zugewachsen mit Gras ist. Darüber befindet sich ein blauer, wolriger Himmel. Damit wird der Naturfokus, zu sehen in vielen von Studio Ghiblis Filmen, fühlbar gemacht. Ein Anime-Modell lässt sich mit einer voll kontrollierbaren Kamera in der dritten Person durch das Gebiet bewegen. Dies soll dem Benutzer des Programmes die Möglichkeit geben, die Umgebung aus diversen, interessanten Kameraperspektiven zu betrachten. Wieder wurden zwei Kameras genutzt. Die erste Kamera rendert das 3D-Modell der Anime-Figur mit Hilfe der in den vorherigen Szenen entwickelten Umriss- und Cartoon-Shader. Die zweite Kamera implementiert einen Post-Processing-Effekt mit dem Namen „Anisotroper Kuwahara-Filter“. Dieser stellt eine Weiterentwicklung des in der vorherigen Szene genutzten Kuwahara-Filters dar. Statt eines quadratischen Filterkernels wird ein elliptischer genutzt, welcher sich verformen kann, um sich an die lokalen Merkmale des Bildes anzupassen. Hierfür werden lokale Orientierung und Anisotropie aus Eigenwerten und Eigenvektoren des Strukturtensors berechnet. Diese weiteren Berechnungen werden durchgeführt, um die quadratischen Artefakte des originalen Kuwahara-Filters zu vermeiden. Der Effekt wird mit der Nähe des Pixels zur Kamera schwächer, so dass weit entfernte Elemente stärker stilisiert werden.

Dann findet eine Analyse der Authentizität der Nachbildungen statt.

Dabei wurde festgestellt, dass die Nachbildung der Gertie-Szene als gelungen betrachtet wird. Dies liegt einerseits an dem akkurat eingefangenem visuellen Stil, welcher durch die Simulation der visuellen Artefakte von Filmaufnahmen noch authentischer wird. Zum anderen wurde eine Animation gewählt, welche nicht nur den Charakter von Gertie trifft, sondern uns auch die Möglichkeit bietet, mit dem McCay-Split-System in einer dreidimensionalen Umgebung zu arbeiten. So konnte erforscht werden, wie moderne Technik helfen kann, ältere Techniken zu benutzen und zu vereinfachen. Lediglich einige fehlende Details halten die Nachbildung davon ab, als perfekt bezeichnet werden zu können.

Das Gleiche kann nicht über die Disney-Szene gesagt werden. Hier wurde der visuelle Stil nicht genau genug eingefangen, was vor allem an der Nachbildung des Hintergrundes liegt. Es fehlen Details und Licht- und Schatteneffekte wirken nicht authentisch. Stattdessen wurde der Fokus auf Animation gelegt. Sie sind das Akkurateste an der Nachbildung und wirken fast so, als würden sie aus klassischen Animationen stammen. Auch wenn die Animationen wichtig sind, so können sie leider nicht komplett von der fehlenden Komplexität der Hintergründe ablenken.

In der Ghibli-Szene konnten die kontrastreichen Farben und die Liebe zur Natur erfolgreich eingefangen werden. Die genutzten Shader replizieren die gemalte Ästhetik des Originals sehr gut. Dem kontrollierbaren Charakter hätte mehr Zeit gewidmet werden müssen, denn er passt

mit seinen Klamotten nicht in seine Umgebung. Die größte Schwäche hier sind Ungenauigkeiten bei der Schattierung. Auf Seite der Animationen besteht Möglichkeit zur Verbesserung, die ausdrucksstarke Gestik und Mimik von Ghiblis Werken konnte nicht erfasst werden. Trotzdem stellt die Nachbildung größtenteils zufrieden.

Daraufhin wurden die Rechenzeiten der Shader analysiert. Hierfür wurden identische Szenen entwickelt, welche die Kameras und Shader der nachgebildeten Szenen übernehmen. Es wurde diskutiert, ob sich die Shader für Anwendungen in Echtzeit eignen.

Die Messungen der Gertie-Testszene zeigen, dass die beiden genutzten Shader, der Umriss-Shader und der Visuelle-Artefakte-Shader, jeweils nur ca. 0,5 ms kosten und sich deshalb sehr gut für Echtzeit-Rendering eignen. Daneben wurden die beiden Shader miteinander verglichen. Obwohl die Vermutung bestand, dass der Umriss-Shader aufgrund der vielen Texturabtastungen langsamer sein würde, hat sich das Gegenteil herausgestellt. Der Grund dafür liegt an Hardwareoptimierungen für Texturabtastungen und an der Menge der mathematischen Operationen, die der Visuelle-Artefakte-Shader durchführen muss.

Der Kuwahara-Filter, welcher den größten Teil der Renderzeit in der Disney-Szene einnimmt, ist im aktuellen Zustand nicht für den Einsatz von Echtzeit-Renderings mit 60 FPS zu empfehlen. Für unseren Einsatzfall, in dem 24 FPS angestrebt werden, kann der Shader zwar genutzt werden, ist aber ein offensichtlicher Kandidat, wenn Optimierungen stattfinden müssen.

Ähnliches gilt für den Anisotropen Kuwahara-Filter, welcher den größten Teil der Renderzeit in der Ghibli-Szene einnimmt. Hier werden Ideen geboten, um den Shader in 60 FPS laufbar zu machen. Zum Beispiel kann der Anisotropen Kuwahara-Filter nur einmal auf das Bild angewandt werden, anstatt wie aktuell zweimal. Doch selbst dies ist nicht genug, um eine Nutzung ohne weitere Optimierungen empfehlen zu können.

Daneben wurden zwei Experimente durchgeführt, welche Post-Processing-Shader in verschiedenen Situationen getestet haben. Es wurde herausgefunden, dass die Shader der Gertie-Szene dieselbe Zeit zum Berechnen brauchen, wenn die Szene visuell komplexer aufgebaut ist. Danach wurde anhand der Shader der Ghibli-Szene getestet, ob die Renderzeit linear mit der Anzahl zu berechnender Fragmente steigt. Die Messergebnisse zeigen, dass dies nicht der Fall ist. Dafür können GPU-Optimierungen verantwortlich gemacht werden.

Die Bachelorarbeit zeigt, dass die Nachbildung klassischer handgezeichneter Zeichentrickanimation in 3D eine anspruchsvolle, aber machbare Aufgabe ist. Sie erfordert ein tiefes Verständnis für technische und künstlerische Prozesse, wobei auch ein historischer Kontext dabei helfen kann, authentische Ergebnisse zu erhalten.

Kapitel 7

Ausblick

Zuletzt wird eine Sammlung aus Ideen für die Zukunft präsentiert. Hier werden Verbesserungsvorschläge für die vorgestellten Szenen und Shader sowie Anwendungsbereiche der Forschungsresultate oder weitere Forschungsbereiche aufgelistet.

Die Elemente, welche den für die Disney-Szene kreierten Hintergrund ausmachen, haben wie in 5.1.2, im Absatz „Visueller Stil“ erwähnt, einen zu simplen Eindruck. Hier kann mit texturbasierten Details wie Alterungseffekten oder handgezeichneten Unregelmäßigkeiten experimentiert werden, welche dem Hintergrund mehr Komplexität geben und die Szene somit näher an das Original kommen lassen.

Während der Evaluation wurde im Kapitel 5.1.3 im Absatz „Visueller Stil“ festgestellt, dass die Feinkontrolle beim Rendern der Schattenseite des Gesichtes fehlt. Da Animationsfilme oder Zwischensequenzen für Videospiele direkt in Unity animiert werden können, macht es Sinn, diese Möglichkeit anzubieten. Je nach Lichtsituation der Szene oder für dramatische Zwecke soll ein Gesicht vielleicht komplett in Dunkelheit gehüllt sein. Hier könnte der Cartoon-Shader erweitert werden. Beispielsweise kann der Entwickler über das Material eine Normal Map mitgeben, die für Berechnungen abgetastet wird, anstatt sich auf die Vertex-Normale zu verlassen.

Auch der im gleichen Absatz festgestellte fehlende Fokus auf die Mimik der Charaktere ist ein Gebiet, in dem Verbesserungsbedarf besteht. Hierzu sollten bestehende Verfahren analysiert werden, welche Animationsstudios nutzen. Neben der Möglichkeit, die Mimik per Hand zu animieren, kann mit „Facial Motion Capture“ experimentiert werden, um Animationen interessanter wirken zu lassen oder sie komplett interaktiv zu gestalten.

Ein weiterer interessanter Ansatz wäre es, Szenen in Virtual Reality zu präsentieren. Sich in einer stilisierten Welt zu bewegen, die an Animationen aus der Kindheit erinnert, wäre eine logische Erweiterung für die Szenen, welche für den Zweck dieser Arbeit entwickelt wurden. Auch Charakteranimationsprinzipien lassen sich hierdurch demonstrieren. Man stelle sich vor, dass

der in VR dargestellte Arm sich auf einmal langstreckt, um nach einem weit entfernten Objekt zu greifen. Somit würde man „Squash and Stretch“ an eigener Haut erleben. Wenn der Arm zurück schnellt, erlebt man einen Rückstoß („Follow-through“). Hier setzen alleine die Kreativität der Game-Designer und Entwickler die Grenzen.

Ein weiterer Ansatz für Videospielentwickler und Animatoren wäre eine Geschichte, in der man durch verschiedene Epochen der Zeit reist, die von den entsprechenden populären Animationsformen repräsentiert werden. Die Vergangenheit ist einfach gehalten, kann sich beispielsweise an der Gertie-Szene orientieren, wird aber im Verlauf der Geschichte immer komplexer und moderner. Der Animationsstil kann die Stimmung des Charakters andeuten und sich bei Enthüllungen komplett verändern. Es kann hier analysiert werden, wie sich verschiedene Animationsstile auf die Emotionen des Betrachters auswirken.

Eine verwandte Idee ändert je nach Standort der Geschichte den Animationsstil. Im Laufe der Geschichte werden unterschiedliche Länder besucht. In Japan ist der Stil vom Anime mit seinen großen Augen und kontrastreichen Farben inspiriert. In Amerika wird von Disney inspiriert ein größerer Fokus auf glatte, runde Formen, klare Farben und ausdrucksstarke Mimik gelegt. Und in Deutschland kann man sich von hiesigen Produktionen wie „Bibi Blocksberg“ inspirieren lassen, welche auf klare Linien und weiche Farben setzen. Dadurch wird nicht nur eine visuelle Vielfalt geschaffen, es wird sich auch mit mehreren Ebenen mit dem aktuellen Standort beschäftigt, was den Betrachter noch weiter in die Geschichte zieht.

Statt eines Videospiels oder einer Animation kann es sich auch anbieten, mehrere Nachbildungen in einer Art digitalem Museum zu präsentieren. Dies hat nicht nur einen Mehrwert für Studenten von Animation, die so einen interaktiven Einblick in verschiedene Animationsstile bekommen. Das Museum kann auch als Kunstwerk an sich gesehen werden, welches Interessierte dazu einlädt, es zu begutachten, und somit sowohl einen künstlerischen Wert als auch einen lehrreichen Effekt bietet.

Eine weitere Idee ist die Erforschung der Umgebung auf die Entscheidungen, die man in virtuellen Welten trifft. Es können dem Spieler hier per Dialogoption verschiedene Möglichkeiten geboten werden, die mehr oder weniger gewalttätig sind. Pro Person wird dies in mehreren Umgebungen wiederholt, welche von verschiedenen Animationsstilen inspiriert sind. Daraufhin kann analysiert werden, welche der Stile einen beruhigenden Effekt haben und bei welchen die Spieler animierter dazu sind, weniger gehemmt zu handeln.

Es besteht die spannende Möglichkeit, weitere Shader zu entwickeln, welche die visuelle Qualität der Nachbildungen erheblich steigern können. Viele Animationsstile interpretieren Wolken oder Wasseroberflächen komplett unterschiedlich. Es wird also ein weites Spektrum an technischen und künstlerischen Herausforderungen geboten, diese Stilelemente einzufangen.

Es besteht hier wieder die Möglichkeit, Shader interaktiv zu gestalten. Beispielsweise können Wellen geschlagen werden, wenn ein Objekt die Wasseroberfläche durchdringt. Dies würde den Beobachter weiter beeindrucken und in die Welt der Nachbildung ziehen.

Des Weiteren können Qualität und Authentizität für ähnliche Arbeiten auch durch Befragung einer repräsentativen Gruppe bewertet werden. Die Befragung sollte hier nicht nur Branchenexperten und Animationskünstler beachten, auch Laien mit Interesse an Animation sollten befragt werden. Hierfür können standardisierte Befragungstechniken wie die Likert-Skala zur Bewertung von Faktoren wie Authentizität, Stiltreue und ästhetischer Wahrnehmung verwendet werden. Diese können daraufhin mit Verfahren wie einem Mittelwertsvergleich mittels t-Test analysiert werden. Eine weitere Option ist es, semistrukturierte Interviews zu unternehmen, die wertvolle Hinweise darauf liefern können, welche Aspekte der Nachbildung besonders überzeugen und bei welchen es sich lohnt, diese weiter zu optimieren.

Literatur

- [18] *Spider-Man: Into the Spider-Verse*. Screenshot. Dez. 2018. URL: <https://www.youtube.com/watch?v=g4Hbz2jLxvQ> (besucht am 14.12.2024).
- [37] *The Worm Turns*. Screenshot. 1937. URL: <https://www.youtube.com/watch?app=desktop&v=M21UsdJnkBY&t> (besucht am 26.11.2024).
- [97a] *Prinzessin Mononoke*. Screenshot. Juli 1997.
- [97b] *Prinzessin Mononoke*. Screenshot. Juli 1997.
- [Ben16] G. Bendazzi. *Animation: A World History: Volume I: Foundations - The Golden Age*. Animation: A World History. CRC Press, Taylor & Francis Group, 2016. ISBN: 978-1138035317.
- [Ben17a] G. Bendazzi. *Animation: A World History: Volume II: The Birth of a Style - The Three Markets*. Animation: A World History. CRC Press, Taylor & Francis Group, 2017. ISBN: 978-1138035324.
- [Ben17b] G. Bendazzi. *Animation: A World History: Volume III: Contemporary Times*. Animation: A World History. CRC Press, Taylor & Francis Group, 2017. ISBN: 978-1138035331.
- [BHM95] Jack Burnham, John Hardy und Kyle Meadors. *Comparison of the Roberts, Sobel, Robinson, Canny, and Hough Image Detection Algorithms*. 1995. URL: <https://api.semanticscholar.org/CorpusID:141065776>.
- [Can05] J. Canemaker. *Winsor McCay: His Life and Art*. Harry N. Abrams, 2005. ISBN: 9780810992344.
- [Cav11] S. Cavalier. *The World History of Animation*. A RotoVision book. University of California Press, 2011. ISBN: 9780520261129.
- [Cav15] D. Cavallaro. *The Anime Art of Hayao Miyazaki*. McFarland, Incorporated, Publishers, 2015. ISBN: 9780786451296.

- [Coh08] Émile Cohl. *Fantasmagorie*. Screenshot, Werk befindet sich in der Public Domain. 1908. URL: https://upload.wikimedia.org/wikipedia/commons/4/44/Fantasmagorie_%28Cohl%29.GIF (besucht am 26.11.2024).
- [Com] RED Digital Cinema Camera Company. *Understanding Lens Vignetting*. URL: <https://www.red.com/red-101/lens-vignetting>.
- [Doc] Unity Documentation. Version Unity 6 (6000.0). URL: <https://docs.unity3d.com/Manual/render-pipelines-overview.html> (besucht am 23.11.2024).
- [Dun04] Andrew Dunn. Nov. 2004. URL: <https://upload.wikimedia.org/wikipedia/commons/9/99/Zoetrope.jpg> (besucht am 14.12.2024).
- [Enc24] The Editors of Encyclopædia Britannica. *Art Nouveau*. Okt. 2024. URL: <https://www.britannica.com/art/Art-Nouveau> (besucht am 21.11.2024).
- [Foua] The Blender Foundation. URL: <https://www.blender.org/about/> (besucht am 22.11.2024).
- [Foub] The Blender Foundation. URL: <https://www.blender.org/about/foundation/> (besucht am 22.11.2024).
- [Fouc] The Blender Foundation. URL: <https://www.blender.org/get-involved/user-stories/> (besucht am 22.11.2024).
- [Foud] The Blender Foundation. URL: <https://docs.blender.org/manual/en/latest/animation/armatures/skinning/introduction.html> (besucht am 01.12.2024).
- [KKD10] J.E. Kyprianidis, H. Kang und J. Döllner. „Anisotropic Kuwahara Filtering on the GPU“. In: *GPU Pro - Advanced Rendering Techniques*. Hrsg. von W. Engel. 2010, S. 247–264.
- [Kyp10] Jan Eric Kyprianidis, Amir Semmo, Henry Kang und Jürgen Döllner. „Anisotropic Kuwahara Filtering with Polynomial Weighting Functions“. In: *TPCG*. 2010. URL: <https://api.semanticscholar.org/CorpusID:16684644>.
- [McC14] Winsor McCay. *Gertie the Dinosaur*. Screenshot, Werk befindet sich in der Public Domain. 1914. URL: <https://www.youtube.com/watch?v=32pzHWUTcPc> (besucht am 26.11.2024).
- [McC19] Winsor McCay. „Lesson One“. In: *Applied Cartooning, Division 11*. Minneapolis: Federal School of Applied Cartooning, 1919.

- [Pen02] Robert Penner. *Robert Penner's Programming Macromedia Flash MX*. 1. Aufl. USA: McGraw-Hill, Inc., 2002. ISBN: 0072223561.
- [Pro] Walt Disney Productions. *Mickey Mouse / run cycle*. URL: <https://i1.wp.com/thinkinganimation.com/wp-content/uploads/2009/08/HowtoDraw-20Mickey10.jpg> (besucht am 29.11.2024).
- [TJ81] F. Thomas und O. Johnston. *The Illusion of Life: Disney Animation*. Abbeville Press, 1981. ISBN: 9780896592339.
- [Wol12] Justin Wolf. *Art Nouveau*. Hrsg. von Peter Clericuzio. Jan. 2012. URL: <https://www.theartstory.org/movement/art-nouveau/> (besucht am 21.11.2024).

Abbildungsverzeichnis

1.1	Nachbildung eines Zoetrops aus dem viktorianischen Zeitalter [Dun04]	1
1.2	„Spider-Man: Into the Spider-Verse“ arbeitet Stilmerkmale von 2D-Animation und Comic-Büchern erfolgreich in seine 3D-Animation ein [18]	2
2.1	Émile Cohls Fantasmagorie [Coh08]	8
2.2	Winsor McCays Gertie the Dinosaur, 1914 [McC14]	9
2.3	Beispiel der Verwendung des McCay Split Systems [McC19]	12
2.4	Screenshot aus einem Mickey Mouse Cartoon von 1937 [37]	16
2.5	San und Ashitaka, die Hauptcharaktere von Prinzessin Mononoke [97a]	17
2.6	Ashitaka vor einer grünen Gebirgslandschaft, bei dem Versuch, seinen Bogen zu spannen [97b]	18
3.1	Humanoide Armatur, erstellt in Blender, zu sehen innerhalb eines durchsichtigen Meshes [Foud]	23
3.2	Eine simple Shader-Pipeline	24
4.1	Schlüsselposen für Gerties Tanzanimation in Blender	28
4.2	Erfasste Umrisse verschiedener primitiven Formen mit dem geschriebenen Post-Processing-Shader	29
4.3	Der Post-Processing-Effekt simuliert die visuellen Artefakte eines auf einer alten Kamera aufgenommenen Bildes	30
4.4	Frames eines „Run Cycles“ für Mickey Mouse [Pro]	33
4.5	Animierte und interaktive Elemente in der 3D-Szene, Schrank (i), Auto (ii) und Flasche (iii)	34
4.6	Beispielszene links vor Anwendung des Kuwahara-Filters (i) und rechts danach (ii).	37
4.7	Der Kernel passt sich an die Beschaffenheit des Bildes an. [Kyp10]	41

5.1	Finale Nachbildung von Winsor McCays „Gertie the Dinosaur“	45
5.2	Finale Nachbildung von Disneys Stil der 1930er und 1940er Jahre	46
5.3	Finale Nachbildung von Studio Ghiblis Stil	47

Tabellenverzeichnis

5.1	Genutzte Shader sortiert nach Szene und Kamera innerhalb der Szene	44
5.2	Gemessene Renderzeiten der Post-Processing-Shader der Gertie-Szene	50
5.3	Gemessene Renderzeiten der Post-Processing-Shader der Disney-Szene	51
5.4	Gemessene Renderzeiten der Post-Processing-Shader der Ghibli-Szene	52
5.5	Gemessene Renderzeiten einer visuell komplexeren Szene unter Nutzung der Post-Processing-Shader der Gertie-Szene	53
5.6	Gemessene Renderzeiten einer Testszene in einer Auflösung von 2560 x 1440 Pixeln unter Nutzung der Post-Processing-Shader der Gertie-Szene	54

