HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Maximilian Putz

3D-Pipeline zur Imitation einer 2D-Ästhetik – Ein Leitfaden für Cartoon Look in Videospielen

Mittweida, 2022

Fakultät Computer- und Biowissenschaften

BACHELORARBEIT

3D-Pipeline zur Imitation einer 2D-Ästhetik – Ein Leitfaden für Cartoon Look in Videospielen

Autor:
Herr Maximilian Putz

Studiengang:
Medieninformatik und interaktives Entertainment

Seminargruppe:
MI18w2-B

Erstprüfer:
Prof. Dipl.-Ing. Alexander Marbach

Zweitprüfer:
M. Sc. Manuel Heinzig

Einreichung:
Mittweida, 26.09.2022

Bibliografische Beschreibung:

Putz, Maximilian:

Thema der Arbeit:
3D-Pipeline zur Imitation einer 2D-Ästhetik – Ein Leitfaden für Cartoon Look in Videospielen

Topic of thesis:
3D pipeline for the imitation of a 2D aesthetic – a guideline for cartoon look in video games

142 Seiten, Hochschule Mittweida, University of Applied Sciences Fakultät Computer- und Biowissenschaften, Bachelorarbeit, 2022

Referat:

In dieser Arbeit wird eine Vorgehensweise für die Erstellung von Grafiken zur Nachempfindung eines 2D Cartoon Looks mithilfe von 3D-Daten vorgestellt und evaluiert. Dafür werden vorerst essenzielle Definitionen in Bezug auf Stil geklärt, wichtige Stilelemente identifiziert, erläutert und in einer 3D-Umgebung praktisch umgesetzt. Es wird dabei eine tatsächliche Nachbildung von Spielelementen durchgeführt, um diese schlussendlich bewerten zu können.

Inhaltsverzeichnis

Glossar

Addon meint jegliche Form von Software-Erweiterung. Im Hinblick auf diese Arbeit wird der Begriff besonders im Zusammenhang mit der Software Blender verwendet.

Ambient Occlusion (deutsch Umgebungsverdeckung) ist ein spezieller Lichteffekt, welcher genauer in 2.2.3 Ambient Occlusion beschrieben wird.

API (Application Programming Interface) ist eine Ansammlung von Programmfunktionen, welche die Interaktion einer Software mit externen Systemen ermöglicht.

Art Style (deutsch Kunststil) stellt einen branchentypischen Begriff für die Einordnung einer bestimmten Zusammensetzung stilistischer Merkmale dar. Eine ausführliche Definition ist unter 2.1.4 Schlussfolgerung der Definition für einen 2D Art Style gegeben.

Asset meint einen, in sich geschlossenen, Bestandteil zur Erstellung eines Spiels. Ein Musikstück, ein Bild oder ein Script stellen ein Asset dar.

Baking meint hier allgemein gesehen das Abspeichern von berechneten Daten. So würde man beispielsweise beim Abspeichern einer computergenerierten Simulation vom Simulation Baking sprechen. Dabei ist die Serialisierung der Simulationsinformationen und nicht das Rendern dieser in Bilder gemeint.

Balancing betrifft im Rahmen der Spielentwicklung das Anpassen von Werten im Hinblick auf ein bestmöglich faires Spielerlebnis.

Bloom ist ein Effekt in der Computergrafik zur Darstellung heller Lichtquellen und wird in Kapitel 2.2.2 Bloom näher erläutert.

Cel Shading ist eine Methode zur stilisierten Darstellung von Schatten und wird in Kapitel 2.2.2 Cel Shading näher erläutert.

Collection ist eine divers einsetzbare Gruppierung von Objekten in Blender.

Color Ramp Node Eine Color Ramp ist ein Farbverlauf. Durch die Color Ramp Node, lässt sich in Blender ein Eingabewert auf einen speziell festgelegten Farbverlauf abbilden.

Cross Hatching ist eine bestimmte Art der stilisierten Schattierung in Kunstwerken und wird in Kapitel 2.2.1 Cross Hatching genauer erläutert.

Dash meint im Hinblick auf Spiele meist eine ruckartige und schnelle Bewegung (üblicherweise des Spieler-Charakters) in eine Richtung.

Deformation-Bone oder nur Bone meint eine digitale Art eines Knochens, welcher als Grundbaustein eines Rigs zur Verformung eines 3D-Modells verwendet werden kann.

Duplicates sind ein Mittel zur Verdeutlichung von schnellen Bewegungen und kommen häufig in alten Cartoons zum Einsatz. Die Technik lässt sich mit dem Effekt der Bewegungsunschärfe bei einer Kamera vergleichen. Eine genauere Erläuterung befindet sich in Kapitel 2.3.2 Duplicates und Smearing.

Eevee Rendering Engine ist ein Echtzeit-Renderer in Blender.

Frame bezeichnet im Rahmen von Animationen ein Einzelbild.

Frame Rate (deutsch Bildrate) beschreibt die Frequenz, in der Einzelbilder in einem Video, Film oder einer Animation abgespielt werden. Frame Rate wird in Kapitel 2.3.1 Frame Rate genauer erläutert.

Game Engine ist eine Entwicklungsumgebung, welche speziell für die Entwicklung interaktiver Inhalte, darunter besonders von Spielen, ausgelegt ist.

Gameplay ist ein branchentypischer Begriff und meint die genaue Art und Weise, wie ein Spieler mit einem Spiel interagieren kann bzw. interagiert.

Grease Pencil ist eine Funktion der Software Blender. Eine detailliertere Beschreibung dieser Funktion befindet sich in Kapitel 3.2.1 Vorstellung verwendeter Blender Funktionen.

Highlight (deutsch Glanzlicht) beschreibt eine Maximalstelle des zurückgeworfenen Lichts besonders bei glänzenden Oberflächen.

Icosphere (deutsch Regelmäßiges Ikosaeder) ist eine dreidimensionale, geometrische Form, bestehend aus 20 kongruenten, gleichseitigen Dreiecken.

Inlines ist eine bestimmte Form von Line Art. Inlines werden in Kapitel 2.2.1 Inlines genauer erklärt.

Keyframes (hier besonders im Sinne der 3D-Animation) speichern Werte eines animierten Objekts wie etwa Position, Rotation oder Skalierung. Durch verschiedene Interpolationen der Werte können dann Übergänge zwischen Keyframes erstellt werden.

Level meint einen, in sich geschlossenen, Abschnitt eines Spiels.

Line Art (deutsch Linienkunst) ist der Überbegriff für verschiedene Linien-Stile. Eine genauere Erklärung findet sich in Kapitel 2.2.1 Line Art.

Look meint das übergreifende Aussehen und das Gefühl eines Bildes durch einen bestimmten Stil.

Mapping Node ist eine bestimmte Node in Blender. Durch diese kann ein Eingabevektor anhand seiner Position, Rotation oder Skalierung modifiziert werden.

Mesh oder auch 3D-Mesh meint die unterliegende Struktur eines 3D-Modells, also die Zusammensetzung von Polygonen und deren Vertices mit entsprechenden dreidimensionalen Positionswerten.

Node taucht hier meist im Zusammenhang mit Blender auf. Gemeint ist ein Algorithmus, welcher verschiedene Parameter zur Werteingabe und -ausgabe bereitstellt und darüber mit gleichartigen Algorithmen verbunden werden kann.

Noise Texture ist eine bestimmte prozedurale Textur.

Normal Map ist eine Textur, in der die Ausrichtung einer Oberfläche abgebildet wird. Dadurch lässt sich etwa die Lichtinteraktion auch über die zugrunde liegende Form der Oberfläche hinaus verändern.

NPR (Non-photorealistic rendering) ist ein Überbegriff für die Erstellung von Bildern, Animationen und ähnlichen, bei denen bewusst keine realitätsgetreue Darstellung, sondern die Umsetzung bestimmter stilistischer Elemente verfolgt wird.

Outlines ist eine bestimmte Form von Line Art. Outlines werden in Kapitel 2.2.1 Outlines genauer erklärt.

Output Properties Tab ist eine bestimmte Benutzeroberfläche innerhalb der Software Blender.

Package (hier vorrangig Unity Package) meint eine spezielle Ansammlung von Programmfunktionen zur Erfüllung einer bestimmten Aufgabe.

Pipeline ist eine abgestimmte Ansammlung oder Reihenfolge von Prozessen zum Erzielen eines bestimmten Ergebnisses oder Herstellen eines Produkts.

Polygon (deutsch Vieleck) meint hier speziell im Sinne der Computergrafik eine Fläche als Grundbaustein zur Erstellung von 3D-Modellen.

Post Processing (deutsch Nachbearbeitung) ist ein Überbegriff für das nachträgliche Modifizieren von Bildinformationen. Es befindet sich eine genauere Erläuterung im Kapitel 2.4.2 Post Processing-Effekte.

Renderer ist eine Software zum Umwandeln von Daten, hier besonders 3D-Daten, in Bildinformationen.

Render Properties Tab ist eine bestimmte Benutzeroberfläche innerhalb der Software Blender.

Rigging beschreibt einen Prozess, bei dem ein "Skelett" (Rig) aus digitalen Knochen (Bones) erstellt wird. Dieses soll im weiteren Verlauf dazu dienen, ein 3D-Modell anhand von logisch platzierten Gelenken verformen zu können.

Rigify ist eine spezielle Software-Erweiterung für Blender mit Fokus auf die automatisierte Erstellung von Rigs.

Rubber Hose (deutsch Gummischlauch) beschreibt den charakteristischen Stil alter Cartoons, bei dem sich die Gelenke der Charaktere schlauchartig bewegen.

Sculpting (hier im Sinne von 3D-Sculpting) beschreibt die Verformung eines 3D-Modells anhand von Methoden, welche einer realistischen Verformung von Lehm nachempfunden sind.

Smearing ist, ähnlich zu Duplicates, ein Mittel zur Verdeutlichung von schnellen Bewegungen und kommt häufig in alten Cartoons zum Einsatz. Eine genauere Erläuterung befindet sich in Kapitel 2.3.2 Duplicates und Smearing.

Sprite meint eine Grafik, welche für die spezielle Einbindung in einer Entwicklungsumgebung z.B. für Spiele ausgelegt ist.

Subsurface Scattering ist eine Methode aus der Computergrafik, zur realistischeren Beleuchtung von 3D-Objekten. Eine genauere Erläuterung dieser Methode befindet sich in Kapitel 2.2.3 Subsurface Scattering.

Timing ist ein branchentypischer Begriff bei der Spielentwicklung und meint das richtige Festlegen eines oder mehrerer Zeitpunkte z.B. bei der Ausführung einer Animation.

Topologie meint im Hinblick auf die Betrachtung eines 3D-Modells die speziellen Charakteristiken der Ausrichtung des Kantennetzes.

Vertex (Plural: Vertices) meint im Hinblick auf ein 3D-Modell einen Eckpunkt eines Polygons. Ein Vertex speichert unter anderem Positionskoordinaten und ist der Grundbaustein für 3D-Modelle.

VFX (Visual Effects) meint üblicherweise animierte oder simulierte Effekte, welche sich nicht in die Kategorien Charaktere oder Objekte einordnen lassen.

Voronoi Texture ist eine bestimmte prozedurale Textur.

Weighting oder oft Skinning, meint den Prozess, bei dem Vertices einer Anzahl von Bones eines Rigs, mit einem Einflussfaktor (dem Gewicht) zwischen 0 und 1 zugeordnet werden.

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Geschichte von Cartoon-Animationen reicht mittlerweile weit über 100 Jahre zurück. Émile Cohl's "Fantasmagorie" gilt als der erste animierte Cartoon und erschien bereits 1908. [1] Zur Entstehungszeit der ersten Cartoons hatten diese nur eine Lauflänge von wenigen Minuten und dienten meist als Vorprogramm von abendfüllenden Spielfilmen. [2] Mit der Einführung des Tonfilms bekam das Medium einen erneuten Aufschwung, da die neuartige Unterstützung durch Geräusche, Dialog und Musik, Cartoon-Filme viel geeigneter für ein Massenpublikum machte. [2] 1928 fand in dem knapp achtminütigen Kurzfilm "Steamboat Willie" auch die später weltbekannte Disney-Cartoon-Figur "Micky Maus" ihren ersten Auftritt auf der Leinwand.[3]
Walt Disney beeinflusste in den darauf folgenden Jahren maßgeblich die Geschichte der Animations-Industrie. Mit "Schneewittchen und die sieben Zwerge" kam 1937 der erste abendfüllende animierte Film in die amerikanischen Kinos (1938 erschien der Film auch in Deutschland). Der Film war nicht nur ein riesiger Erfolg, sondern brachte ebenfalls technische Erfindungen zur Weiterentwicklung des Mediums hinter der Leinwand mit sich. Was die dabei zum Einsatz gebrachte "Multiplane-Camera" mit dem sogenannten Parallax-Effekt zu tun hat, soll jedoch in einem späteren Kapitel zum Thema werden.
Schon 1995 etablierte sich mit dem Film "Toy Story", in Zusammenarbeit mit "Pixar Animation Studios", das Medium der 3D-Animationsfilme. 3D-Animationsfilme versprachen ökonomisch gesehen eine viel bessere Einnahmequelle, da diese in den Produktionskosten den bisherigen 2D-Animationsfilmen nicht nachstanden und üblicherweise größere Umsätze generierten. Das hatte besonders bei Disney eine Umstellung der Produktions-Pipeline auf 3D-Animationsfilme zur Folge und die Anzahl veröffentlichter 2D-Animationsfilme wurde über die Zeit geringer. [4]

[1] Bendazzi, Giannalberto (2017): Animation. A World History, S. 31
[2] Kletschke, Irene (2011): Klangbilder. Walt Disneys Fantasia (1940), S. 18
[3] Disney+ (o. J.): Steamboat Willie
[4] Ted Tschang, Feichin; Goldstein, Andrea (2004): Production and Political Economy in the Animation Industry: Why Insourcing and Outsourcing Occur, S. 4-5

Im Jahr 2017 hatte der nostalgische Cartoon Look der 1930er [5] jedoch seine große Wiederbelebung. Das in diesem Jahr veröffentlichte Videospiel "Cuphead", griff den Stil der alten Cartoons auf und brachte ihn in die Videospielindustrie. Der Titel von "StudioMHDR" war ein riesiger Erfolg und gewann, neben zwei weiteren, vor allem auch den Preis für "Best Art Direction" bei den Game Awards 2017.[6]

1.1 Motivation

Das Entwicklerstudio "Motion Twin" veröffentlichte 2018 das Videospiel "Dead Cells". Dead Cells erstrahlt visuell in einem Pixel Art Style. Dieser Stil zeichnet sich vor allem durch die deutliche Darstellung von Farbquadraten, eben den sogenannten "Pixeln" aus. Pixel Art bildet damit eine nostalgische Referenz an die anfänglichen Zeiten der Computergrafik.

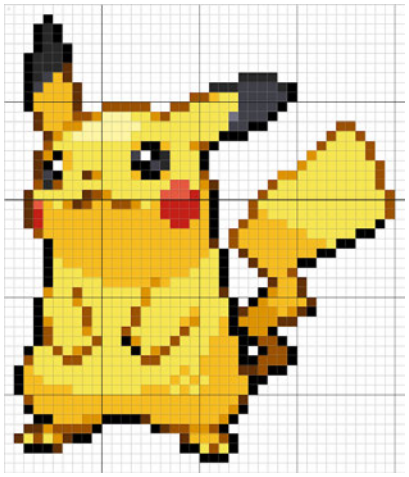


Abbildung 1.1: Grafik im Pixel Art Style

Für diese Arbeit viel relevanter ist jedoch, wie genau dieser Stil speziell im Fall von Motion Twin's Dead Cells umgesetzt wurde.

[5] McGowan, David (2019): Cuphead: Animation, the public domain, and home video remediation, S. 2

[6] The Game Awards (2017). The Winners 2017

> *"To make up for the lack of bandwidth and still deliver on quality, we had to find a pipeline that could give us great looking pixel art, without having to hand draw each and every retake"* [7]

Wie Thomas Vasseur in einem Beitrag für die Website "Game Developer" erklärt, war er für ein Jahr der einzige Mitarbeiter, der für die Erstellung aller artistischen Inhalte im Spiel zuständig war. Das umfasste neben der generellen Art Direction, Charaktere, Monster, sowie entsprechende Animationen und Special Effects. [7] Eine effektive Gestaltung des Workflows war deswegen für ihn essenziell. Die finalen Sprites von Dead Cells sind weder von Hand gezeichnet, noch von Hand animiert. Thomas Vasseur erstellte nach einem grundlegenden Konzept 3D-Modelle für die jeweils abzubildenden Charaktere. Folgend wurden diese auch in 3D animiert, exportiert und durch eine hauseigene Software im Pixel Art Style gerendert. [7]

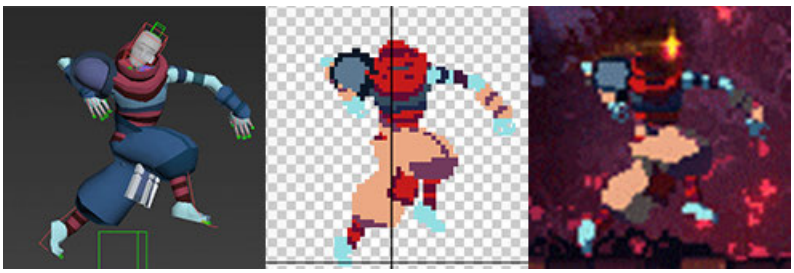


Abbildung 1.2: Abbildung der Etappen der Pipeline von Dead Cells
(von links nach rechts: 3D-Modell, Pixel Art Rendering mit neuer Farbzuweisung, Finales Bild im Spiel)

Thomas Vasseur zählte folgende Vorteile auf, welche dieser Workflow ihm gebracht hatte. Es gab keine Notwendigkeit mehr, jeden einzelnen Frame von Hand zu zeichnen, da nach der Fertigstellung der 3D-Animation die gesamte Bildfolge von der Software gerendert werden konnte. [7] Außerdem ergab sich die Möglichkeit, Animationen für verschiedene Modelle erneut zu benutzen. Darüber hinaus erklärt Vasseur, dass der entscheidendste Vorteil in der Möglichkeit, schnell und simpel Anpassungen an den Animationen machen zu können, liegt. Das kommt vor allem dann zum Tragen, wenn sich etwa Timings im Hinblick auf das Balancing des Spiels später ändern. Ein weiterer sehr interessanter Vorteil der Existenz von 3D-Daten war, dass für Charaktere eine Normal Map generiert werden konnte.

[7] Vasseur, Thomas (2018): Art Design Deep Dive: Using a 3D pipeline for 2D animation in Dead Cells

Abbildung 1.3: Aus 3D-Daten generierte Normal Map

Vereinfacht gesagt, kann durch solch eine Normal Map die Ausrichtung einer dreidimensionalen Oberfläche, zweidimensional abgebildet werden. Infolgedessen ließ sich das Volumen der Charaktere darstellen, wodurch zum Beispiel mithilfe eines Shaders, den Charakteren stilisierte Schatten hinzugefügt werden konnten. [8] Besonders bei Indie Game-Entwicklern ist ein effektives Ressourcenmanagement sehr wichtig, da meist nur beschränkt Mittel zur Verfügung stehen. [8] Dieser Fakt macht die eben genannte alternative Pipeline zur Erstellung von Pixel Art noch einmal interessanter.

In dieser Arbeit wird genau dieser Ansatz weiterführend betrachtet. Nach Inspiration durch die Dead Cells-Pipeline soll erforscht werden, inwiefern sich auch Art Styles, in einer höheren Auflösung mithilfe von 3D-Rendering-Techniken zur späteren Verwendung in Videospielen umsetzen lassen. Diese Arbeit betrachtet dafür den, in der Einleitung beschriebenen, Stil der 1930er-Cartoons. Eine exzellente Referenz dafür stellt das bereits erwähnte Spiel Cuphead von StudioMDHR dar.

1.2 Aufgabenstellung

In dieser Arbeit steht somit eine Pipeline, beziehungsweise eine vorgeschlagene Vorgehensweise, für die Erstellung stilisierter 2D Art Styles aus 3D-Daten im Fokus. Dafür werden vorerst Überlegungen getroffen, durch welche genau identifizierbaren stilistischen Mittel, sich ein bestimmter Art Style kategorisieren lässt. Vorbereitend dazu werden komplexe Begriffe, wie etwa Art Style definiert und festgelegt, wie diese im Rahmen dieser Arbeit zu verstehen sind. Die dabei herausgestellten stilistischen Mittel beziehen sich vorrangig auf die Einordnung von Cartoon-Stilen, wie sie in der Einleitung

[8] Vasseur, Thomas (2018)

schon erwähnt wurden. Durch die dabei entstehende Pipeline soll es möglich sein, Assets für Videospiele zu generieren. Das umfasst Hinter- und Vordergrundelemente, aber auch entsprechende Notwendigkeiten für die Animation von beispielsweise Objekten und Charakteren. Dafür werden im ersten Abschnitt stilistische Mittel zur Einordnung des Zeichenstils, Animationsstils und zuletzt auch Mittel zur finalen Komposition innerhalb einer Game Engine dargestellt. Danach werden die genannten stilistischen Elemente für Zeichen-, Animationsstil und Komposition anhand der Möglichkeiten ihrer Umsetzung innerhalb einer 3D-Software, im Falle dieser Arbeit Blender, betrachtet. Folgend sollen die vorgestellten Methoden zum Rendern von 3D-Daten in einen Cartoon-Stil praktisch evaluiert werden. Dafür wird der Rubber Hose Cartoon-Stil gezielt imitiert. Als Referenz dafür wird der Videospieltitel Cuphead von StudioMDHR herangezogen. Die Ergebnisse dieser Evaluation werden zuletzt reflektiert und geschaffene Möglichkeiten, aber auch Einschränkungen, sowie Grenzen der vorgestellten Methoden aufgezeigt.

2 Charakteristische Stilelemente zur Analyse eines Cartoon-Stils

Um folgend einen entsprechenden Cartoon-Stil nachstellen zu können, sollen in diesem Kapitel vorerst die Grundlagen und die Vorgehensweise zur Analyse eines Art Styles erläutert werden. Dazu wird vorerst der betrachtete künstlerische Rahmen generell eingegrenzt und die Bedeutung verwendeter Begriffe definiert. Nachfolgend sollen generelle Stilmittel kategorisiert und beispielhaft demonstriert werden. Da im Rahmen dieser Arbeit die Imitation eines Art Styles für Videospiele thematisiert wird, müssen bei der Analyse des Art Styles ebenfalls stilistische Merkmale für Animation und Komposition betrachtet werden.

2.1 Grundlagen zur Stilanalyse

In den nächsten Unterkapiteln werden grundlegende Definitionen für die später verwendeten Begrifflichkeiten vorgestellt. Darunter fallen besonders die verwendeten Begriffe zur Beschreibung eines Art Styles und wie diese im Rahmen dieser Arbeit zu verstehen sind. Schlussfolgernd soll eine Definition für den Begriff 2D Art Style aufgestellt werden.

2.1.1 Abgrenzung des betrachteten Kunstbereichs

Beim Begriff Kunst handelt es sich um ein schwer zu definierendes Konzept, welches auch innerhalb der Geisteswissenschaften stark umstritten ist.[1] Grundlegend existiert jedoch eine typische Unterteilung in einzelne Kunstgattungen. In simpler Form wird dabei in Bildende Kunst, Darstellende Kunst, Literatur und Musik unterteilt. In der Moderne und besonders mit dem Wachstum digitaler Medien haben sich zudem viele neue Gattungen

[1] Beil, Benjamin et al. (2018): Game Studies, S. 379

entwickelt. Darunter zählt etwa die moderne Gattung "Media Art", welche Video- und Computergrafik sowie Animation umfasst.[2]

Für den Rahmen dieser Arbeit wird dabei festgelegt, dass die Bereiche Musik und Literatur entfallen. Wenn auch die auditive Umsetzung einen wichtigen Bestandteil bei der Imitation eines Cartoon-Stils darstellt, wird diese folgend nicht betrachtet, um einen übersichtlichen Rahmen zu wahren.
Tatsächlich sollen folgend nur die visuellen, stilistischen Elemente zur Erstellung eines zweidimensional betrachtbaren Videospiels thematisiert werden. Darunter fallen die akkurate Nachbildung einzelner Stilelemente, die entsprechende Animation und auch die zusammenführende Komposition aller Bestandteile des finalen Bildes.

2.1.2 Definition Stil

Prof. em. Dr. Angelika Linke verfasste eine umfangreiche Definition für den Begriff Stil. In dieser wird der Begriff Stil in verschiedene Aspekte unterteilt. Diese umfassen Stil als signifikante Form, als typische Form, als Intention oder Effekt, als Differenz und Identitätsphänomen, als Wahl und als Kontrasterfahrung. Da diese einzelnen Aspekte jedoch dazu dienen, den Stilbegriff disziplinübergreifend einzuordnen, sollen diese folgend eingegrenzt werden. [3] In diesem Kapitel wird keine allgemeingültige Definition für Stil formuliert. Eher soll mit Hinblick auf gängige Definitionen des Begriffs eine prägnante Formulierung geschaffen werden, welche beschreibt, was in dieser Arbeit gemeint ist, wenn von Stil, spezieller von Art Style, gesprochen wird. Daraus ergibt sich, dass die Definition von Stil sich vorrangig auf die Beschreibung von bewegten und unbewegten, visuellen, zweidimensionalen Abbildungen bezieht. Prof. em. Dr. Angelika Linke beschreibt Stil unter anderem als typische Form.

> *"Stil als signifikante Form ist an das Phänomen der Typik gebunden. Stilelemente sind immer schon types, sind wiederkehrende und wiedererkannte Ausprägungen eines Musters, das seinerseits als Projektion aus der Zusammenschau der konkret vorkommenden Formen konstituiert wird."* [3]

Hieraus lässt sich entnehmen, dass sich ein bestimmter Stil immer aus verschiedenen Stilelementen zusammensetzt. Die Stärke der Ausprägung oder auch das Fehlen dieser Stilelemente ermöglicht es, mehrere Kunstwerke einem bestimmten Stil zuzuordnen. [3]

[2] Grau, Oliver (2004): Virtual Art. From Illusion to Immersion, S. 3
[3] Linke, Angelika (2009): Rhetorik und Stilistik / Rhetoric and Stylistics, S. 1134-1135

2.1.3 Definition Stilelement

Mit Hinblick auf die im Rahmen dieser Arbeit getroffene Definition von Stil wissen wir, dass Stil das Zusammenspiel bestimmter Ausprägungsgrade von Stilelementen darstellt. Ein Stilelement beschreibt nun eben genau so eine quantifizierbare Ausprägung im künstlerischen Sinne.

2.1.4 Schlussfolgerung der Definition für einen 2D Art Style

Das Wort Art Style (deutsch Kunststil) stellt einen branchentypischen Begriff aus der Videospielindustrie dar. Wenn im weiteren Verlauf dieser Arbeit von einem Art Style gesprochen wird, meint dies die jeweilige Ausprägung einzelner Stilelemente, im Hinblick auf Bild, Animation und Komposition. Anhand der Intensität oder dem Fehlen dieser Ausprägungen lassen sich verschiedene visuelle Werke zu einem bestimmten Art Style zuordnen. Von einem 2D Art Style wird insbesondere gesprochen, weil sich die folgend betrachteten Kunstwerke zweidimensional abbilden lassen. Räumliche Kunstwerke, wie etwa Skulpturen, 3D-Modelle oder vergleichbare entfallen dementsprechend aus der Definition eines 2D Art Styles.

2.2 Stilelemente eines 2D Art Styles

Innerhalb dieses Kapitels sollen typische Stilelemente zweidimensionaler Medien zusammengetragen werden. Dabei ist zu erwähnen, dass die folgende Auflistung keinen Anspruch auf Vollständigkeit erhebt. Die zusammengetragenen Stilelemente sind des Weiteren dem Thema der Arbeit entsprechend besonders zur Einordnung von Variationen verschiedener Cartoon-Stile ausgelegt. An gegebener Stelle werden Hinweise darüber gegeben, welche Stilelemente und Prinzipien für die Interpretation weiterführender Art Styles interessant wären.

2.2.1 Line Art

Wenn im Rahmen dieser Arbeit von Line Art (deutsch Linienkunst) gesprochen wird, meint das den gesamten Umfang von Linien, welche die Form und Oberfläche eines

dargestellten zweidimensionalen Objekts definieren. Das stilistische Element, was hier gemeint ist, lässt sich schon vor mehreren Jahrhunderten, etwa bei Renaissance-Künstler Albrecht Dürer feststellen. [4]

Abbildung 2.1: Die vier apokalyptischen Reiter von Albrecht Dürer, 1511

Line Art-Darstellungen sind jedoch besonders auch im Hinblick auf Non-photorealistic Renderings in den letzten Jahren beliebter und interessanter geworden. [4] Um später genauere Kategorisierungen vornehmen zu können, wird Line Art als Stilelement in drei Bestandteile unterteilt. Unterschieden wird dabei zwischen Outlines, Inlines und Cross Hatching. Jedes dieser Merkmale wird folgend genauer beschrieben. Abbildungen, welche Line Art als Stilelement beinhalten, haben dabei immer mindestens einen dieser Bestandteile, können jedoch auch alle drei aufweisen.

Outlines

Outlines (deutsch Außenlinien) stellen die Silhouette eines entsprechenden Objekts in Form einer Linie dar. Durch diesen Bestandteil wird dabei üblicherweise nur die Kontur dieses Objekts abgebildet.

[4]Elber, Gershon (1999): Interactive Line Art Rendering of Freeform Surfaces, S. 1

Inlines

Im Gegensatz zu Outlines sind mit Inlines (deutsch Innenlinien) jene Linien gemeint, die innerhalb der Kontur eines Objekts weiterhin die Form dieses Objekts definieren. In der folgenden Abbildung sollen anhand eines Cartoon-Handschuhs der Bereich Outlines und Inlines aufgezeigt werden. In diesem Beispiel sind beide Bestandteile des Stilelements Line Art dargestellt.

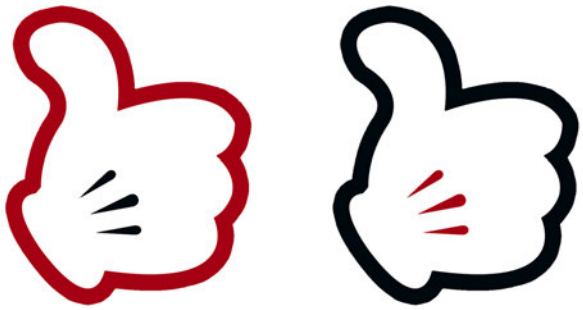


Abbildung 2.2: Darstellung der benannten Line Art-Typen in Rot gekennzeichnet: Outlines (links), Inlines (rechts)

Cross Hatching

Den dritten Bestandteil von Line Art stellt das Cross Hatching (deutsch Kreuzschraffur) dar. Bei diesem Stilelement werden Ansammlungen von parallel verlaufenden Linien gezeichnet und in mehreren Ebenen mit versetztem Winkel überlagert.[5] Diese können gerade sein, aber auch der Oberflächenform des darzustellenden Objekts folgen. Üblicherweise wird dieses Stilelement zur Darstellung von Schatten verwendet.

[5]Gran, Carlos And´ujar (2020): Non-photorealistic Rendering: Cross Hatching, S. 1

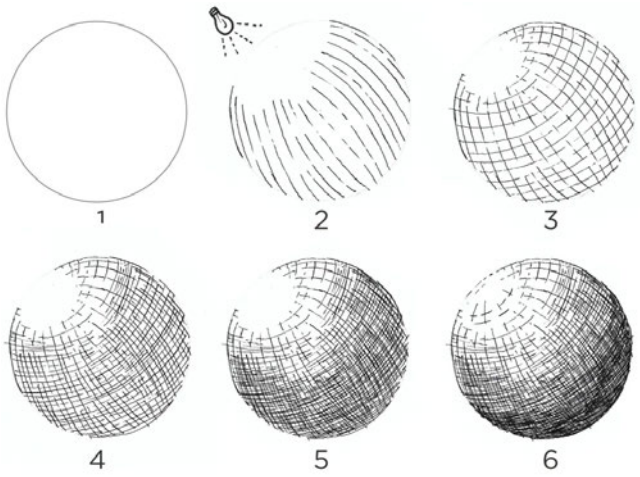


Abbildung 2.3: Stufenweiser Aufbau von Cross Hatching-Linien

2.2.2 Grundlegende Beleuchtungsmodelle für NPR

Da schlussendlich zweidimensionale Bilder aus dreidimensionalen Objekten gerendert werden sollen, empfiehlt es sich an dieser Stelle einen Zusammenhang zwischen den beiden Gebieten zu ziehen. Folgend soll deswegen vorerst auf simple Techniken zur Beleuchtung von dreidimensionalen Modellen eingegangen werden. Dabei liegt der Fokus nicht auf der technischen Umsetzung oder der Darstellung der Gesamtheit aller Möglichkeiten für dreidimensionale Beleuchtung. Vielmehr wird auch hier schon gezielt auf Rendering-Methoden eingegangen, welche ein Nachempfinden eines zweidimensionalen Zeichenstils im Sinn haben.
Grundsätzlich wird bei Beleuchtung zwischen direkter und indirekter Beleuchtung unterschieden.

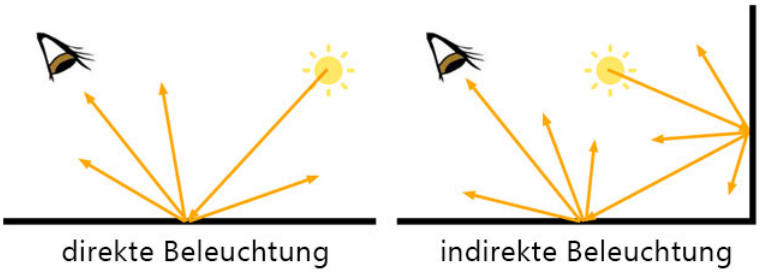


Abbildung 2.4: Darstellung zu direkter und indirekter Beleuchtung

Direct Illumination

Direct Illumination (deutsch direkte Beleuchtung) oder auch Local Illumination ist eine Art der Beleuchtung in der Computergrafik. In dieser Arbeit wird weiterführend von Direct Illumination bzw. Directional Lighting (deutsch gerichtete Beleuchtung) gesprochen, da für die lokale Beleuchtung von 3D-Modellen gerichtete Lichtquellen verwendet werden. Der Begriff Direct Illumination selbst fasst jegliche Algorithmen zum Rendern von Licht zusammen, bei denen auf einer entsprechenden Oberfläche eines 3D-Objekts nur jene Lichtinformationen dargestellt werden, welche direkt von einer Lichtquelle kommen.[6] Dieses Beleuchtungsmodell ist nicht unbedingt realistisch, für die Anwendung in NPR jedoch absolut genügend und wird im Folgenden vorrangig verwendet. Ein Beispiel für den Effekt von Directional Lighting soll der folgende Ausschnitt aus dem Film "Chihiros Reise ins Zauberland" von Studio Ghibli darstellen. Dabei fällt auf, dass die gerichtete Beleuchtung hier offensichtlich von rechts kommt. Auf den dargestellten Charakteren sieht man eine klare Unterteilung zwischen Licht- und Schattenseite. Des Weiteren kommt hier schon der Cel Shading-Effekt zum Einsatz, welcher detailliert im anschließenden Kapitel thematisiert wird.

Abbildung 2.5: Ausschnitt aus dem Animationsfilm "Chihiros Reise ins Zauberland"

In einem 3D-Renderer würde dieses Licht nun entsprechend der Oberflächeneigenschaften eines gegebenen Objekts diffus oder glänzend zurückgegeben werden. Dieser Ansatz ist zwar realitätsnäher, jedoch würde das Rendern des 3D-Objekts auf diese Weise sehr schnell preisgeben, dass es eben basierend auf einem 3D-Modell entstanden ist. Da versucht werden soll, die Ästhetik eines 2D-Bildes nachzustellen, gilt es Möglichkeiten dafür zu finden.

[6] Autodesk (o.J.): Indirect (global) vs. direct illumination

Cel Shading

Cel Shading, oder auch als Toon Shading bezeichnet, ist eine 3D-Rendering-Technik zur Nachstellung traditioneller 2D-Animationsstile.[7] Der Begriff Cel Shading richtet sich dabei mittlerweile nicht mehr nur an das rein technische Prinzip, sondern wird heutzutage auch als artistische Methode angesehen, um 3D-Modelle stilisiert darzustellen. Der Shading-Stil bzw. Cel Shading als Stilelement zeichnet sich durch hart abgetrennte Farbflächen, welche die jeweiligen Bereiche für Licht und Schatten abgrenzen, aus. In der folgenden Abbildung wurde ein 3D-Modell einmal mit einem typischen Shader gerendert und einmal mit zwei Varianten der eben benannten Cel Shading-Technik.

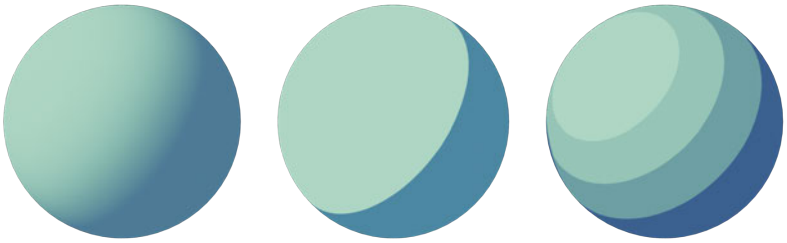


Abbildung 2.6: Gegenüberstellung: Lineare Beleuchtung (links) mit Cel Shading (mittig und rechts)

Unlit

Als Unlit (deutsch unbeleuchtet) werden alle Darstellungen betrachtet, welche nicht von Licht und Schatten betroffen sind und stattdessen mit einer 100% Beleuchtung dargestellt werden.[8] Die Darstellung ergibt sich hierbei vorrangig aus der Form, sowie der Abgrenzung einzelner Farbbereiche. Besonders in den letzten Jahren ist dieses Stilelement im Bereich von Vector Art häufig vertreten.

[7] Luque, Raul Reyes (2012): The Cel shading Technique, S. 1
[8] Ahearn, Luke (2018): 3D game environments: Create professional 3D game worlds, S. 11

Abbildung 2.7: Kunstwerk im Vector Art Style

Im Falle des Renderings von 3D-Objekten funktioniert das Prinzip analog, nur dass hier die Darstellung von Schatten nicht vom Künstler selbst weggelassen wird. Bei der Darstellung von 3D-Objekten müssen im Renderer entsprechende Einstellungen getroffen werden, sodass die Schatten nicht dargestellt und das Objekt stattdessen in einer 100% Beleuchtung beziehungsweise nur mit einer bestimmten Oberflächenfarbe gerendert wird. Von einem bestimmten Blickwinkel betrachtet, definiert sich das 3D-Objekt somit ebenfalls ausschließlich durch die Form und Abgrenzung der Farbbereiche. Dieses Stilelement stellt dadurch eine großartige und einfache Option dar, ein 3D-Objekt zweidimensional gezeichnet aussehen zu lassen.

Bloom

Als Bloom wird ein Effekt bezeichnet, welcher dafür sorgt, dass besonders helle Objekte einen Schein um sich herum emittieren. [9] Dieser Effekt hat eine realistischere Abbildung digitaler Bilder zum Ziel. In der Realität wird ebenfalls ein Schein um helle Objekte vom menschlichen Auge wahrgenommen. Das liegt zum einen an Streuungen von Licht in der Atmosphäre [10], zum anderen aber auch an der Art und Weise, wie Licht generell vom menschlichen Auge wahrgenommen wird. [9] Die Methode selbst ist notwendig,

[9] Spencer, Greg et al. (1995): Physically-based glare effects for digital images, S. 325-327

[10] Fan, Wenshan (2021): A fast and realistic bloom rendering method for large scale 3D scene, S. 178

weil digitale Abbildungen auf Bildschirmen nur mit einem limitierten Spektrum und Helligkeitsstufen dargestellt werden können. [9] Um sehr helle Objekte darstellen zu können, muss also auf andere Lösungen, wie etwa den Bloom Effekt, zurückgegriffen werden. In der folgenden Abbildung wird ein Bild mit dem Bloom-Effekt und ein Bild ohne diesen gegenübergestellt.

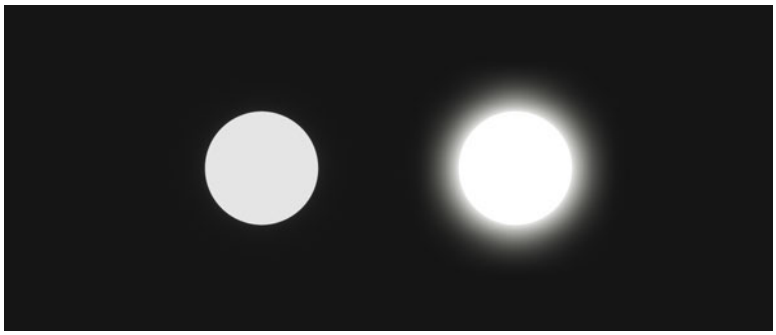


Abbildung 2.8: Vergleich ohne Bloom (links) und mit Bloom (rechts)

2.2.3 Erweiterte Beleuchtungsmethoden

In diesem Abschnitt sollen weiterführend eine Auswahl an Beleuchtungs- und Rendering-Methoden für eine realitätsnähere Darstellung kurz umrissen werden. Die Methoden selbst stehen nicht in direktem Zusammenhang zu NPR, sind jedoch dennoch wichtige Begriffe für eine detailgenaue Analyse diverser Art Styles. Das soll ebenfalls bedeuten, dass für den Rahmen dieser Arbeit nicht näher auf die Funktionsweise der folgend benannten 3D-Rendering-Algorithmen eingegangen wird. Die vorgestellten Techniken dienen vorrangig zur Erläuterung etablierter Systeme und sollen eine Grundlage für die Identifizierung der jeweiligen Beleuchtungsmodelle für die spätere Analyse des zu erzielenden Art Styles bieten.

Global Illumination

Global Illumination oder auch Indirect Illumination stellt wie auch die, in 2.2.2 Direct Illumination angesprochene, Local Illumination einen Überbegriff für Beleuchtungsalgorithmen aus der Computergrafik dar. Man spricht deshalb von “global“, weil für die

Darstellung der Oberfläche eines Objekts, die Lichtreflexionen anderer Objekte und Lichtquellen innerhalb einer gesamten Szene einbezogen werden.[11]

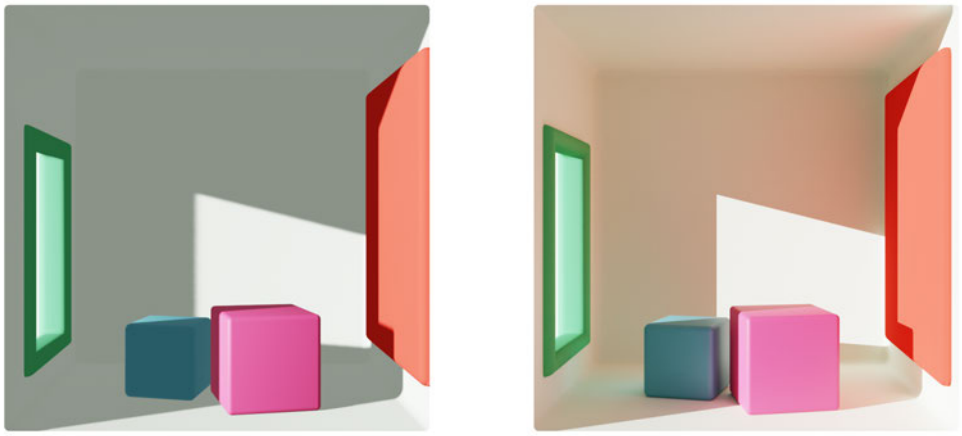


Abbildung 2.9: Vergleich Local Illumination (links) und Global Illumination (rechts)

Subsurface Scattering

Mit Subsurface Scattering (deutsch Volumenstreuung) ist im Rahmen dieser Arbeit eine Methode zur Simulation von Lichtwegen innerhalb transluzenter Materialien gemeint.[12] Durch diese Methode kann zum Beispiel eine realitätsnahe Darstellung der visuellen Eigenschaften menschlicher Haut erzielt werden.

[11] Dutre, Philip et al. (2006): Advanced Global Illumination, S. VII
[12] Jimenez, Jorge et al. (2015): Separable subsurface scattering, S. 188

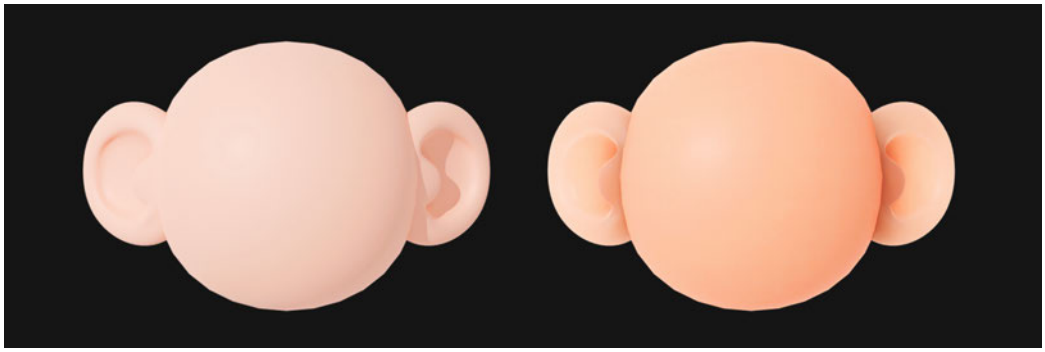


Abbildung 2.10: Vergleich ohne Subsurface Scattering (links) und mit Subsurface Scattering (rechts)

Ambient Occlusion

Ambient Occlusion soll die gezielte Auswahl an Rendering-Algorithmen zur realitätsnahen Darstellung von 3D-Objekten abschließen. Bei der sogenannten Ambient Occlusion (deutsch Umgebungsverdeckung) handelt es sich um einen Lichteffekt, bei dem reflektiertes Licht zwischen nah aufeinander liegenden geometrischen Oberflächen gedämpft wird.[13] Durch diesen Effekt kommt es also zu Schattenflächen an Stellen, wo Objekte sehr nah aufeinander liegen oder etwa an scharfen Kanten.

Abbildung 2.11: Vergleich ohne Ambient Occlusion (links) und mit Ambient Occlusion (rechts)

[13]Kontkanen, Janne; Laine, Samuli (2005): Ambient occlusion fields, S. 1

2.2.4 Farbe

Zur Analyse eines Art Styles müssen ebenfalls die Eigenschaften der verwendeten Farbdarstellung betrachtet werden. Im folgenden Abschnitt sollen die einzelnen Unterbereiche vorgestellt werden, welche im Rahmen dieser Arbeit dafür untersucht werden.

Farbspektrum

Das Farbspektrum, also die Menge an verwendeten Farben im Hinblick auf Farbton, Farbsättigung und Farbwert ist ein wichtiger Faktor zur detaillierten Imitation eines bestimmten Art Styles.

Abbildung 2.12: Visualisierung Farbton (Hue), Farbsättigung (Saturation), Farbwert (Value)

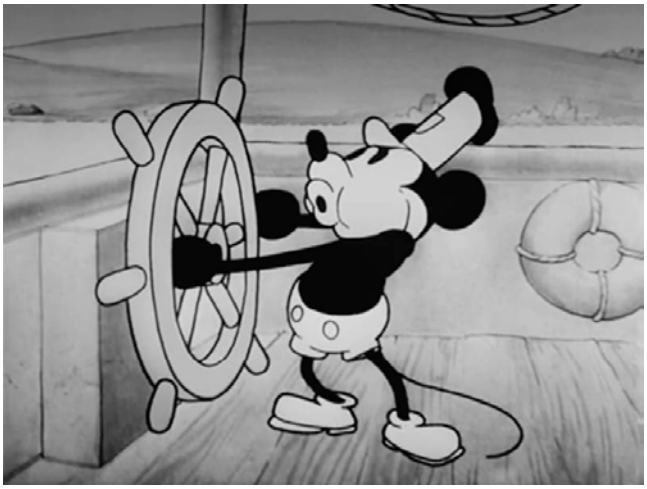


Abbildung 2.13: Ausschnitt aus dem Cartoon Steamboat Willie

Bei der beispielhaften Betrachtung des ersten Micky Mouse-Cartoons, fällt direkt auf, dass dieser noch in Schwarz-Weiß-Tönen ausgestrahlt wurde. Sollte beispielsweise genau der Art Style aus Steamboat Willie rekreiert werden, könnte von Anfang an davon ausgegangen werden, dass Farbton und Farbsättigung einen Wert von null haben müssen. Folglich liegt der Hauptfokus auf der richtigen Umsetzung der Farbwerte, also der Helligkeit entsprechender Elemente. Auch hier fallen direkt Eigenheiten des Art Styles auf. Es ist erkennbar, dass kein maximaler Weißwert abgebildet wird. Dies hängt damit zusammen, dass Steamboat Willie noch von Hand gezeichnet und später mit einer Kamera fotografiert wurde. Daraus ergibt sich, dass ebenfalls beim digitalen Nachempfinden des Stils, ein eingegrenzter Wertebereich der Farben und damit eine absichtliche Minderung des Bildkontrastes umgesetzt werden sollte.

Um die Aspekte des Farbtons und der Farbsättigung ebenfalls kurz zu beleuchten, wird noch ein anderes Beispiel betrachtet. Es empfiehlt sich dabei wieder die generelle Erstellung früherer Kunstwerke und die gegebenen Einschränkungen zu betrachten. Zu Zeiten nicht digitaler Kunst mussten Farbstoffe selbstverständlich gewonnen und hergestellt werden. Manche Farbpigmente waren dabei schwerer zu beschaffen als andere. Das hat zum Beispiel zur Folge, dass in alten Höhlenmalereien, wie der folgend abgebildeten von Altamira von etwa 13.500 v.Chr., neben Schwarztönen, fast ausschließlich Gelb- und Rottöne zu sehen sind.[14]

[14]Ulrich, Karl (2021): Wie kommt die Farbe ins Kunstwerk? S. 1

Abbildung 2.14: Malerei von Altamira, ca. 13.500 v.Chr.

Die Farbpigmente dafür konnten recht einfach aus natürlichen Mineralien gewonnen werden. Selbige Überlegung wäre aber auch beim Nachempfinden von Künstlern im vergangenen Jahrtausend anzustellen. Auch zum Beispiel die Farbpalette von Rembrandt von Rijn im 17. Jahrhundert basierte noch hauptsächlich auf Erdfarben. [15]

Farbquantität

Bei der Analyse eines Art Styles im Hinblick auf Farbe, gilt es ebenfalls die Anzahl an verwendeten Einzelfarben zu betrachten. Eine Einzelfarbe meint dabei eine Farbe, mit genau einem zugeordneten Wert-Tripel in Farbton, Farbwert und Farbsättigung. Ein hellerer Ton dieser Einzelfarbe würde also eine neue Einzelfarbe darstellen und man würde von einer höheren Farbquantität sprechen. Der Unterschied erstreckt sich dabei zwischen einem Art Style, welcher eine sehr umfangreiche Anzahl von Einzelfarben, etwa durch die Verwendung von Farbverläufen, abbildet und einem Art Style, welcher größere Flächen mit einem spezifischen Farbwert, also einer oder weniger Einzelfarben füllt.

Verlauf der Farben

Bei der Analyse soll in diesem Fall nicht nur die Interpolation zwischen zwei unterschiedlichen Farben als Farbverlauf betrachtet werden. Der Verlauf, welcher hier gemeint ist, kann in verschiedensten Variationen auftreten und muss fallspezifisch genauer untersucht

[15] Karl, Ulrich (2021): Wie kommt die Farbe ins Kunstwerk?, S. 3

werden. Bei Wasserfarben geschieht es beispielsweise sehr häufig, dass Farben ineinander laufen. Das geschieht jedoch üblicherweise nie so mathematisch akkurat, wie bei digitalen Farbverläufen. Gerade bei der Imitation physischer Farben müssen verschiedene Eigenschaften der Arbeitsmittel analysiert werden. Etwa ist die Deckkraft einer Farbe sehr entscheidend. Acrylfarbe ist zum Beispiel sehr deckend, während Wasserfarbe nur sehr leicht färbt, gegebenenfalls ineinander verläuft und auch in mehreren Schichten übereinander gemalt werden kann. Die digitale Imitation dieses Effektes stellt eine Herausforderung dar. Ein Ansatz dafür wird jedoch in Kapitel 3.2.3 Farbe beschrieben.

2.2.5 Leinwand-Textur

Bei der Imitation physischer Art Styles gilt es auch die Beschaffenheit des Zeichenuntergrunds zu betrachten. So kommt es besonders bei wenig deckenden Farben vor, dass die Textur des Papiers durchscheint. Ein Beispiel für die Nachstellung dieses Effekts innerhalb einer digitalen Umgebung wäre etwa das Videospiel “Paper Mario“ von Nintendo.

Abbildung 2.15: Paper Mario

2.2.6 Grad der Abstraktion

Zeichnungen, digital wie auch analog, reduzieren meistens das abgebildete Subjekt auf die, für die Darstellung wichtige Essenz. Das ergibt auch nur Sinn, denn es ist nicht

unbedingt notwendig jedes Blatt eines Baumes darzustellen, damit der Betrachter den Baum als solchen erkennt.

Abbildung 2.16: Zwei Bäume mit einem geringen Abstraktionsgrad (links) und einem hohen Abstraktionsgrad (rechts)

Dieser Abstraktionsgrad unterscheidet sich nun in verschiedenen Art Styles. Typischerweise lässt sich sagen, dass ein höherer Abstraktionsgrad mit sehr großen formgebenden Flächen einfacher umzusetzen ist. Das hat zur Folge, dass gerade in gezeichneten Animationsmedien oft ein höherer Abstraktionsgrad vorzufinden ist, als etwa bei Gemälden. Dieser Faktor wird zu einem Großteil von einem jeweiligen Artist selbst gesteuert. Aus diesem Grund wird der Abstraktionsgrad im nächsten Kapitel nicht bei der Umsetzung der hier genannten stilistischen Mittel mit aufgeführt. Der theoretische Aspekt ist für die Analyse eines Art Styles dennoch essenziell.

2.3 Merkmale eines 2D Animation Styles

Im folgenden Kapitel sollen charakteristische Merkmale von 2D-Animationen, besonders von Cartoons, herausgestellt werden. Dafür werden neben technischen Grundlagen, bestimmte historische Lösungen betrachtet, aber auch eine Auswahl genereller Animationsprinzipen thematisiert.

2.3.1 Frame Rate

Unter dem Begriff Frame Rate (deutsch Bildrate bzw. Bildfrequenz) versteht man die dargestellte Anzahl an Einzelbildern über einen bestimmten Zeitraum. Im weiteren Verlauf wird von Frames per Second (fps), also Einzelbildern pro Sekunde gesprochen. Animationen für Filme sind üblicherweise in 24fps gezeichnet. [16] Animationen für das Fernsehen wurden früher in Europa, Afrika und dem Mittleren Osten in 25fps erstellt. Grund dafür war das analoge Übertragungssystem PAL, welches mit einer Bildrate von 50fps läuft. In der heutigen Zeit muss sich anhand solcher Restriktionen nicht mehr orientiert werden. Zur Vereinfachung wird von einer Bildrate von 24fps für übliche Cartoons ausgegangen.

Eine entscheidende Technik, welche jedoch noch kurz erklärt werden soll, ist das sogenannte "Animating On Twos" (deutsch Animation auf Zweien). Dabei wird jedes gezeichnete Bild für die Länge von zwei Frames anstatt von einem auf dem Bildschirm abgebildet. [16] Aus Sicht von Animationsfirmen stellt dies eine großartige Möglichkeit zur Einsparung von Arbeitsaufwand dar. Für den Rahmen dieser Arbeit ist das Animieren "On Twos" als Stilelement für Animationen zu betrachten. Ein großartiges Beispiel dafür ist der 3D-Animationsfilm "Spider-Man™: Into the Spider-Verse". In diesem Animationsfilm von "Sony Pictures Imageworks" findet Animating On Twos seine Verwendung als stilistisches Mittel. [17]

> *"The impact of animating on twos, especially for fast-paced action, provided the desired illustrated visual style for the film, where each frame appeared as its own distinct image, like a panel in a comic book."* [17]

Wie aus dem Zitat hervorgeht, wurde die Animationstechnik gezielt dafür eingesetzt, das Gefühl und die Ästhetik eines Comic-Buches nachzuempfinden und in einen 3D-Animationsfilm zu konvertieren.

2.3.2 Duplicates und Smearing

Unter der Smearing-Technik verstehen man das gewollte Verzerren oder "Verschmieren" eines Charakters oder Objekts, sodass es der Bewegung nachhängt. Der dabei erzielte Effekt ist eine stilisierte und charakteristische Art der Bewegungsunschärfe und hilft dabei

[16] Roberts, Steve (2004): Character Animation in 3D: Use traditional drawing techniques to produce stunning CGI animation, S. 2-6

[17] Sony Pictures Imageworks (o.J.): Spider-man™: Into the spider-verse

schnelle Bewegungen für den Betrachter deutlicher darzustellen. Folgend abgebildet ist ein Ausschnitt aus der Animation des Charakters "Hilda Berg" aus Cuphead. Zur besseren Veranschaulichung sind die Einzelbilder fortlaufend nummeriert. In Frame 2 lässt sich deutlich der gemeinte Smearing-Effekt erkennen.

Abbildung 2.17: Smear-Effekt bei Hilda Berg aus Cuphead

Sogenannte Duplicates dienen ebenfalls dazu, den Effekt von schnellen Bewegungen zu visualisieren. Im Falle der Duplicates, wird ein bestimmter Teil des Charakters oder Objekts dupliziert. So kommt es dabei häufig vor, dass in einem Frame mehrere linke Arme gezeichnet werden. Dafür wird erneut ein Beispiel aus der Animation von Hilda Berg betrachtet. In Frame 6 lässt sich dabei vorerst erneut ein Smear-Frame erkennen und in den folgenden zehn Animation-Frames ist deutlich das Prinzip der Duplicates zu sehen.

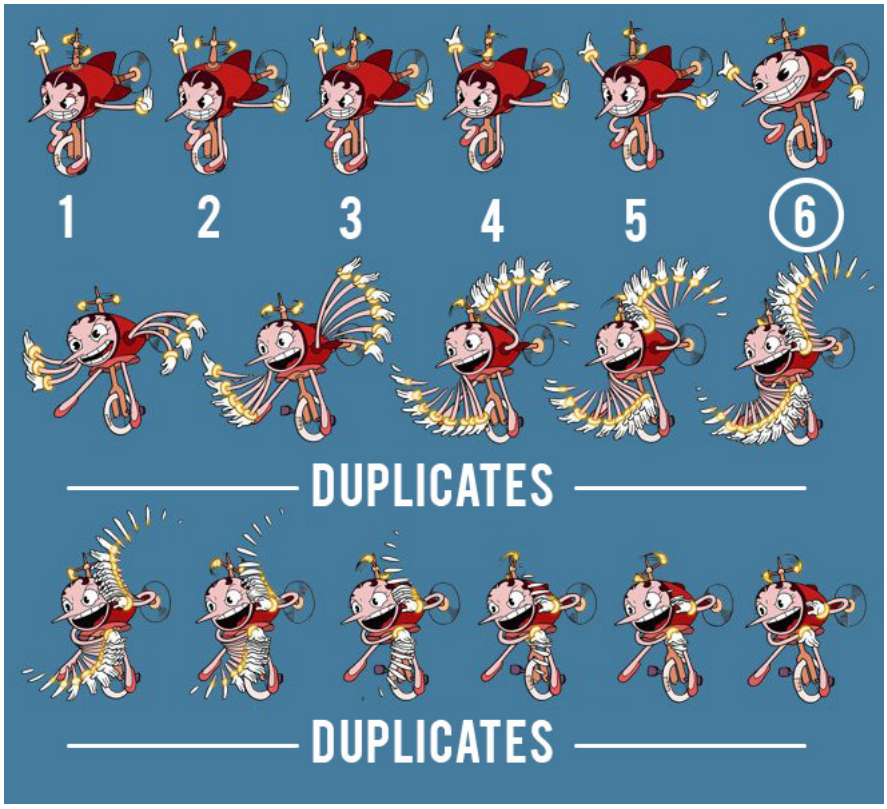


Abbildung 2.18: Duplicates bei Hilda Berg aus Cuphead

Wie in Kapitel 2.3.1 Frame Rate erklärt wurde, werden Cartoon-Animationen üblicherweise mit einer Framerate von 24fps abgespielt. Das bedeutet, dass sehr schnelle Bewegungen von einer meist sehr geringen Anzahl an Einzelbildern dargestellt werden. Duplicates können, wie auch Smears, ebenfalls dabei helfen, schnelle Bewegungen für den Zuschauer besser erkennbar und nachvollziehbar zu machen.

2.3.3 Rubber Hose Animation Style

Der Rubber Hose Animation Style gibt vielen der 1930er-Cartoons ihren charakteristischen Look. "Rubber Hose" lässt sich als Gummischlauch ins Deutsche übersetzen. Dieser Begriff referenziert auf die Art, wie sich die Gliedmaßen der animierten Charaktere in Rubber Hose Animationen bewegen. Bei den Gliedmaßen selbst lassen sich dabei meist keine Gelenke erkennen und die Dicke entlang der Gliedmaßen ist einheitlich. Die Bewegungen verlaufen in sehr kurvigen Formen, wobei sich üblicherweise kein Abknicken

des Verlaufs erkennen lässt. Häufig kommt es auch zu unnatürlichen Streckungen der Gliedmaßen. Überblickend entsteht dadurch ein sehr weicher, schwunghafter Animation Style, dessen Merkmale sich sehr schnell und deutlich erkennen lassen. Eine Visualisierung der beschriebenen kurvenartigen Verformung in gestreckten Gliedmaßen soll in der folgenden Darstellung gegeben werden.

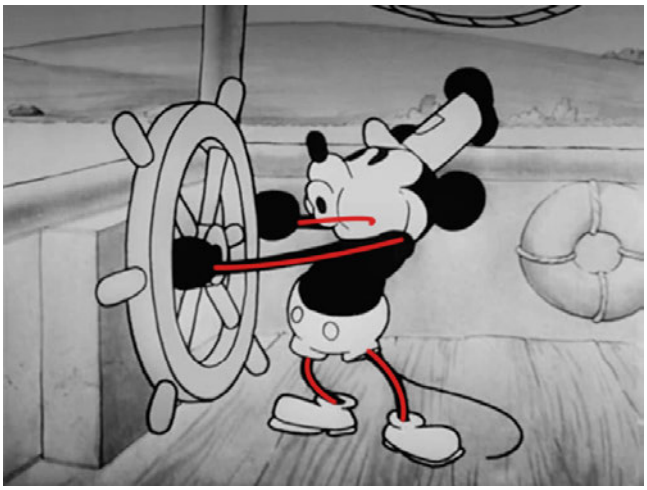


Abbildung 2.19: Kurvenartige Gliedmaßen, gekennzeichnet durch rote Linien

Squash and Stretch

Die "12 Principles of Animation" (deutsch 12 Prinzipien der Animation) sind eine enorm wichtige Grundkenntnis im Hinblick auf die Erstellung von Animationen in jeglichem Bereich. Diese Prinzipien wurden erstmals im Buch "The Illusion of Life: Disney Animation" aufgestellt, welches 1981 erschien. Obwohl die Prinzipien selbst von sehr großer Bedeutung für Animation generell sind, sollen nicht alle hier benannt und erläutert werden. Zum einen soll das einen übersichtlicheren Rahmen waren, zum anderen ist eine Vielzahl der darin verfassten Prinzipien für den jeweiligen Ersteller einer Animation und weniger zur Implementation in einer Pipeline von Belangen. Trotzdem soll hier kurz auf das Prinzip "Squash and Stretch" (deutsch Quetschen und Strecken) eingegangen werden. Dieses Prinzip beschreibt, dass Objekte bei Krafteinwirkungen entsprechend zusammengedrückt oder auseinandergezogen werden. Das Prinzip wird in einem recht überspitzten Maß in alten Cartoons verwendet. Des Weiteren soll speziell dieses Prinzip hier benannt werden, da es für die entsprechende Umsetzung von animierten Charakteren von Bedeutung ist.

2.4 Merkmale des Szenenaufbaus in einem 2D-Videospiel

Im folgenden Abschnitt werden kurz Methoden zur Darstellung und Komposition der vorher erwähnten Einzelobjekte und Animationen vorgestellt. Diese Schritte passieren erst nach dem stilisierten Rendern der einzelnen Assets, sind jedoch sehr zielführend für die Umsetzung eines glaubwürdigen Cartoon-Stils. Dabei soll der Fokus auch lediglich auf der Komposition und Nachbearbeitung der einzelnen Assets liegen. Zur Umsetzung eines fertigen Videospiels wäre auch die Darstellung der Benutzeroberfläche wichtig. Der Unterpunkt Benutzeroberfläche wird jedoch im Rahmen dieser Arbeit nicht behandelt.

2.4.1 Parallax-Effekt

Beim sogenannten Parallax-Effekt werden Bilder auf verschiedenen Ebenen dargestellt. Diese einzelnen Ebenen werden dann in unterschiedlichen Geschwindigkeiten relativ zur Kamera bewegt. Dabei wird üblicherweise die Ebene, welche am weitesten von der Kamera entfernt ist, am langsamsten bewegt und folglich die nächste Ebene am schnellsten. Durch diese Form der relativen Bewegung wird beim Betrachter die Illusion von Tiefe geschaffen. Der Effekt selbst datiert einige Jahrzehnte zurück. Schon 1937 wurde mit genannter Technik die Illusion von Tiefe in Disneys "Schneewittchen und die sieben Zwerge" erzielt. Da zu dieser Zeit die Bilder analog erstellt wurden, konstruierte man eine Vorrichtung, um diesen Effekt umsetzen zu können, die sogenannte "Multiplane Camera".[18]Auf dieser konnten, wie oben beschrieben, Zeichnungen auf verschiedenen Ebenen, in verschiedenen relativen Geschwindigkeiten bewegt werden.

[18]Holiday, Christopher; Pallant, Chris (2021): The depth deception: Landscape, technology and the manipulation of Disney's multiplane camera in Snow White and the Seven Dwarfs (1937), S. 66

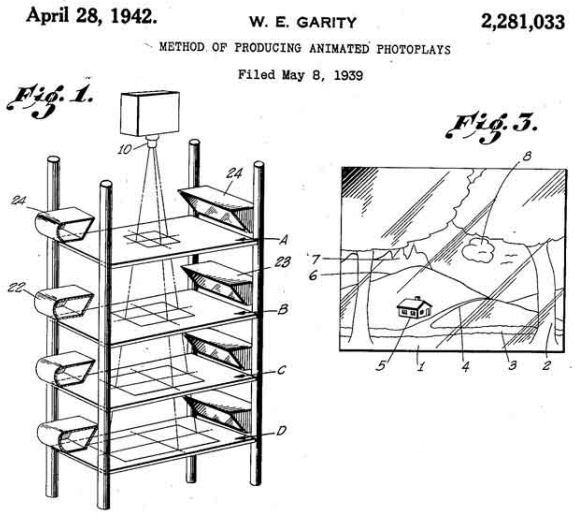


Abbildung 2.20: Multiplane Camera

2.4.2 Post Processing-Effekte

Post Processing (deutsch Nachbearbeitung) meint den Prozess des nachträglichen Anpassens eines dargestellten Bildes. Die, für die Imitation eines Cartoon-Stils, wichtigsten Methoden sollen folgend erläutert werden.

Chromatic Aberration

Chromatic Aberration (deutsch Chromatische Aberration) beschreibt einen Effekt, bei dem Lichtwellen entsprechend ihrer Wellenlänge an einer Linse unterschiedlich gebrochen werden. Das führt zu einer leichten Distorsion einzelner Farben, wodurch sich ein verschieden farbiger Randbereich um ein bestimmtes wahrgenommenes Objekt bildet.[19] Der Effekt findet besonders in der Computergrafik bei der Erstellung digitaler visueller Effekte, aber auch innerhalb von Videospielen häufig Anwendung. Die folgende Abbildung zeigt den gemeinten Effekt.

[19] Gauld, Dylan (2016): Growth: Visualisation of predictive mathematical models using 3D computer graphics and animation, S. 210

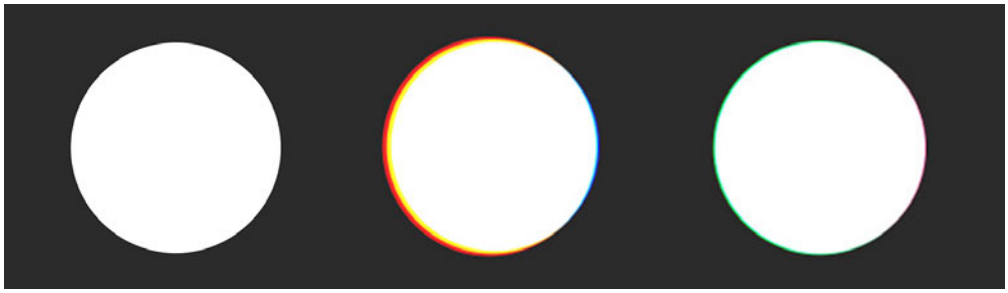


Abbildung 2.21: keine Chromatische Aberration (links), zwei Arten der Chromatischen Aberration (mittig und rechts)

Overlays

Alte Cartoons wurden noch mithilfe analoger Methoden gezeichnet und wiedergegeben. Spezifischer wurden diese früher auf Papier gezeichnet und dann mit einer Kamera abfotografiert. Dieser Prozess hatte, aufgrund der verwendeten Technik, Bildfehler zur Folge. So war etwa die Beleuchtung zwischen einzelnen Frames nicht akkurat und konnte im Hinblick auf die Helligkeit variieren. Kameras selbst verursachen ebenfalls Bildfehler, wie etwa ein Bildrauschen, eine Bildverzerrung aufgrund der Linsenkrümmung oder auch den sogenannten Vignette-Effekt. In Zeiten digitaler Medien werden viele dieser Bildartefakte bei der Erstellung eines Bildes nicht mehr verursacht. Für eine überzeugende Imitation des nostalgischen Looks der 1930er-Cartoons, müssen diese Bildartefakte folglich simuliert werden. Das lässt sich etwa durch die Überlagerung mit generierten Bildunreinheiten oder etwa durch das nachträgliche Hinzufügen eines Bildrauschens erzielen.

Abbildung 2.22: Bildartefakte und Vignette-Effekt

2.5 Zusammenfassung der Merkmale wichtiger Stilelemente

Um dieses Kapitel abzuschließen, wurden die herausgestellten Merkmale eines 2D Art Style und eines 2D Animation Styles in prägnanter Form auf den nächsten Seiten zusammengefasst. Das dient des Weiteren später zur besseren Einordnung der Ergebnisse bei der praktischen Umsetzung.

Tabelle 2.1: Kurzbeschreibung der Stilelemente eines 2D Art Styles

Stilelemente eines 2D Art Styles	
Stilelement	**Merkmale**
Line Art	
Outlines	Kontur um ein Objekt herum; üblicherweise in gleichbleibender Strichstärke
Inlines	Linien innerhalb der Form eines Objekts
Cross Hatching	ineinander überkreuzte Strichflächen; meist zur Schattierung
Beleuchtung	
Direct Illumination	Erkennbare Beleuchtungsrichtung aus meist einer speziellen Lichtquelle
Cel Shading	Harte Abgrenzung der Schattenflächen
Unlit	Keine erkennbaren Schatten- oder Lichtinformationen
Bloom	Deutlicher Schein um helle Objekte herum
Erweiterte Beleuchtung	
Global Illumination	Beleuchtung eines Objekts mit Beachtung anderer Objekte in der Szene
Subsurface Scattering	Simulation des Lichtwegs durch ein transluzentes Objekt; Objekt erscheint durchscheinender
Ambient Occlusion	Schatten an engen Auflageflächen zwischen Objekten und ähnlichen Kanten
Weitere	
Farbe	Zusammenspiel aus der verwendeten Farbpalette, der Farbquantität und dem Verlauf der Farben
Leinwand-Textur	Beeinflussung der Farbdarstellung durch ein unterliegendes Material
Grad der Abstraktion	Stärke der Simplifizierung eines bestimmten Objekts

Tabelle 2.2: Kurzbeschreibung der Stilelemente eines 2D Animation Styles

Stilelemente eines 2D Animation Styles	
Stilelement	**Merkmale**
Frame Rate	Anzahl der Einzelbilder, die jede Sekunde abgespielt werden
Duplicates und Smearing	Verschmieren oder Duplizieren von Teilen eines Objekts zur besseren Darstellung schneller Bewegungen
Rubber Hose	schlauch- und gummiartige Bewegung, vor allem von Gliedmaßen

Tabelle 2.3: Kurzbeschreibung der Stilelemente des Szenenaufbaus

Stilelemente des Szenenaufbaus in einem 2D-Videospiel	
Stilelement	**Merkmale**
Parallax-Effekte	Verschiebung verschiedener Tiefen-Ebenen in unterschiedlichen Geschwindigkeiten relativ zur Kamera
Post Processing	Anwendung verschiedener Nachbearbeitungen und Bildüberlagerungen; in diesem Fall hauptsächlich zur Nachstellung alter analoger Filmmethoden
Chromatic Aberrations	leichter Versatz verschiedener Farbwerte
Grain Overlay	Simulation eines Bildrauschens
Vignette-Effekt	Abdunkeln des Bildes in einem Verlauf, welcher von den Bildecken ausgeht

3 Umsetzung der genannten 2D-Stilelemente in einem 3D-Renderer

Nachdem die einzelnen Stilelemente im vorherigen Kapitel erläutert wurden, sollen nun Methoden zur technischen und praktischen Umsetzung vorgestellt werden. Dabei werden systematisch Ansätze zur Umsetzung der Stilelemente aus dem vorangegangenen Kapitel vorgestellt. Dafür soll erneut erwähnt werden, dass das Rendering der einzelnen Assets schon in Blender geschieht und keine weitere Software benötigt wird. Die gerenderten Bilder werden dann nur mithilfe der Unity Engine entsprechend zusammengefügt und für die Anwendung in einer interaktiven Echtzeit-Applikation ausgelegt.

3.1 Vorstellung der verwendeten Software

Für den Rahmen dieser Arbeit wurde sich fast ausschließlich auf die Verwendung von lediglich zwei Softwarepaketen beschränkt. Dies umfasst einerseits Blender als divers einsetzbare Software zur Erstellung visueller Medien und die Unity Engine, welche sehr gut für die Erstellung von Videospielen geeignet ist. Beide Programme sollen folgend kurz im Hinblick auf deren Anwendungsgebiet und Zweck vorgestellt werden.

3.1.1 3D-Software: Blender

Als 3D-Software wird für diese Arbeit Blender verwendet. Blender ist eine kostenfreie, Open-Source 3D-Suite. [1] Die Software deckt ein sehr breites Feld an Aufgaben innerhalb einer 3D-Pipeline ab. So können u.a. Modeling, Rigging, Animation, Simulation, Rendering und Compositing innerhalb einer Software erledigt werden. [1] Für die Einbindung in eine weiterführende Pipeline unterstützt Blenders API sogar Python Scripting, wodurch die Anwendung individualisiert und spezifische Tools entwickelt werden können. [1]

[1] Blender Foundation (o.J.): The Software

3.1.2 Game Engine: Unity

Die Unity Engine ist eine Software zur Erstellung von Echtzeit-Anwendungen. Die Engine selbst bietet eine große Variation an Hilfsmitteln für die Umsetzung verschiedenster Projekte in Visualisierung und Interaktivität.[2] Für den Rahmen dieser Arbeit vorrangig interessant sind jedoch die Game Development Tools der Unity Engine. Unity bietet eine Vielzahl an Packages, welche zur Umsetzung der gegebenen Aufgabenstellung eine sehr gute Grundlage bilden. Über einzelne Umsetzungsmöglichkeiten, besonders im Hinblick auf Post Processing-Effekte, wird im Kapitel 4 Praktische Nachbildung des Stils gesprochen.

3.2 Umsetzung der benannten Stilelemente eines 2D Art Styles

In Kapitel 2.2 Stilelemente eines 2D Art Styles wurde bereits eine Auswahl an Stilelementen, welche bei der Betrachtung eines 2D Art Styles zu identifizieren sind, aufgeführt. Dieses Kapitel soll darauf aufbauend, grundlegende Herangehensweisen vorstellen, wie die jeweils beschriebenen Stilelemente sich innerhalb der verwendeten Programme umsetzen lassen.

3.2.1 Line Art

In diesem Abschnitt werden verschiedene Methoden für die Erstellung von Outlines, Inlines und Cross Hatching in Blender vorgestellt. Der jeweils erzielte Effekt wird in Abbildungen dargestellt. Grundlage dafür stellt ein simples Cartoon-Hand-Modell dar. Dieses besteht aus zwei separaten Meshes, welche sich aus dem hauptsächlichen Hand-Mesh (in der Abbildung gelb) und der Erweiterung der Handschuh-Form (in der Abbildung blau) zusammensetzt.

[2] Unity Technologies (o.J.): Unity

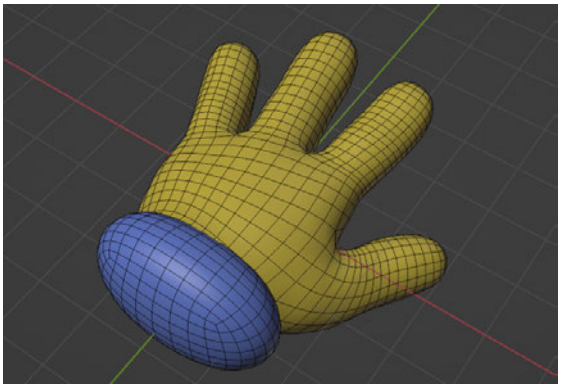


Abbildung 3.1: 3D-Modell Cartoon-Handschuh

Bei der weiterführenden Verwendung des Modells soll dieses mit einem Rig versehen werden. Dieses dient dazu, das 3D-Modell durch strategisch platzierte Deformation-Bones verformen zu können. Jedem dieser Deformation-Bones werden eine bestimmte Anzahl an Vertices zugewiesen, welche dieser dann beeinflusst. Man spricht dabei vom sogenannten Weighting. Durch die Erstellung sogenannter Controller können danach sogar ganze Gruppen dieser Deformation-Bones auf diverse Arten transformiert werden, um beim Arbeiten mit dem Rig schnell und einfach komplexere Posen umsetzen zu können. Ein Beispiel für solch ein Hand-Rig lässt sich in der nachfolgenden Abbildung sehen.

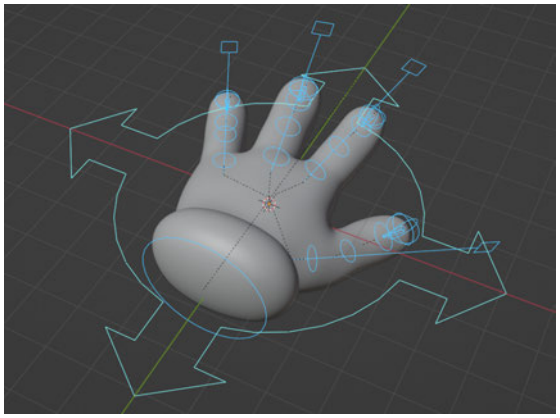


Abbildung 3.2: Cartoon-Handschuh-Rig

Für die Vorstellung der Techniken zum Rendern von Line Art ist diese Information wichtig, da eine Kompatibilität mit dem eben erwähnten Rig gewährleistet sein muss.

Vorstellung verwendeter Blender Funktionen

Für die Erstellung von Line Art werden folgend verschiedene Herangehensweisen beleuchtet. Diese verwenden unter anderem Blender-interne Funktionen, welche vorab kurz im Hinblick auf deren Funktionsweise erklärt werden sollen.

Die erste verwendete Blender-Funktion stellt dabei Grease Pencil dar. Grease Pencil-Objekte sind ein bestimmter Objekttyp in Blender. Dieser kann verwendet werden, um im 3D-Raum zu zeichnen. Die Linien werden durch einzelne Punkte dargestellt, was die Möglichkeit offen lässt, diese durch Sculpting-Werkzeuge auch nach dem Zeichnen zu editieren. Im "Edit Mode" können die Punkte auch gelöscht oder hinzugefügt werden. Des Weiteren lassen sich durch sogenannte "Modifier" auch prozedural Anpassungen an den gezeichneten Linien treffen. Ein recht triviales Beispiel, wäre das Hinzufügen von einem Noise-Modifier, um eine Linie an willkürlichen Punkten zu verschieben. Somit kann ein zittrigeres Erscheinungsbild erstellt werden.

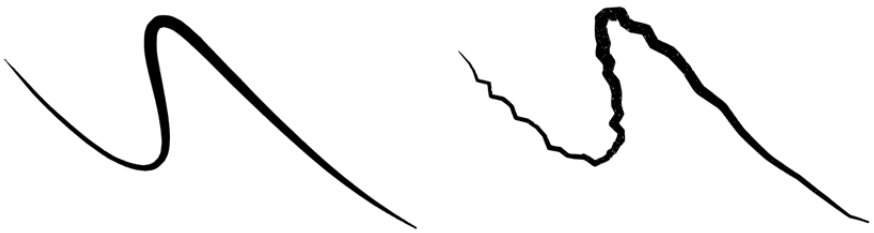


Abbildung 3.3: Grease Pencil-Objekte ohne Modifier (links) und mit Noise-Modifier (rechts)

Die zweite Funktion, welche vorab erläutert werden soll, ist Freestyle. Freestyle ist eine Non-photorealistic Rendering Engine in Blender. Diese verwendet Mesh-Daten und die Z-Depth, um daraus Linien anhand bestimmter selektierter Kanten zu generieren. [3] Durch die Einstellung von Parametern oder ggf. die Anwendung von Python Scripts kann eine Vielzahl verschiedener Linienstile erzielt werden. [3]

[3] Blender Foundation (o.J.): Freestyle: Einführung

Outlines

In diesem Abschnitt sollen die im Kapitel 2.2.1 Outlines erläuterten Konturlinien erstellt werden. Dafür werden drei Methoden für die Umsetzung in Blender vorgestellt. Abschließend hierfür werden die jeweiligen Ergebnisse der drei Methoden gegenübergestellt.

Mesh-Based

Outlines können einerseits durch die Erstellung eines duplizierten Meshs generiert werden. Dafür wird eine Technik aus der Computergrafik namens Backface Culling verwendet. Grundsätzlich besitzt ein in 3D abgebildetes Polygon immer eine Normale. Simpel gesagt ist das ein Richtungsvektor, welcher angibt in welche Richtung das Polygon zeigt. Beim sogenannten Backface Culling werden alle Polygone, welche nicht in Richtung der Kamera zeigen, nicht gerendert. Genau diese Backface Culling-Technik kann zum Erstellen der Mesh-basierten Outlines genutzt werden. Zuerst wird ein weiteres Mesh erstellt, welches größer ist, als das originale. Je größer das neue Mesh skaliert wird, desto dicker ist die später abgebildete Outline. Im zweiten Schritt werden die Normalen des neuen Meshs invertiert, sodass diese nach innen zeigen. Im letzten Schritt werden alle Einstellungen getroffen, damit das neue Mesh mit einem Backface Culling-Material und in entsprechender Farbe gerendert wird. Als Ergebnis ist jetzt nur die Differenz des größer skalierten Meshs, als Outline um das originale Mesh herum zu sehen.

Grease Pencil

Eine weitere Option zur Erstellung von Outlines anhand eines 3D-Objekts, wäre durch die Verwendung eines Grease Pencil-Objekts. Seit Blender Version 2.93, lassen sich Grease Pencil-Objekte mit einem "Line Art-Modifier" verwenden.[4] Damit dieser funktioniert, muss nur ein Grease Pencil-Objekt in der Szene erstellt worden sein. Der Inhalt davon ist nicht wichtig. Des Weiteren wird eine Kamera in der Szene benötigt. Danach kann über den Modifier-Tab der Line Art-Modifier auf dem Grease Pencil-Objekt hinzugefügt werden. Dieser braucht einige grundlegende Einstellungen. Dafür wird dem Modifier die Referenzen für das Objekt bzw. die Objekte übergeben, welche modifiziert werden sollen. Dabei kann auf einzelne Objekte, Collections oder sogar ganze Szenen referenziert werden. Zuletzt benötigt der Modifier noch einen zugewiesenen Grease Pencil-Layer und

[4]Blender Foundation (o.J.): Blender 2.93: Grease Pencil

ein Material. Letzteres kann danach beispielsweise auch für eine Änderung der Farbe modifiziert werden.

Freestyle

Die letzte vorgestellte Methode, um Outlines zu rendern, wäre durch Blenders Freestyle Render Engine. Diese lässt sich einfach durch einen Klick aktivieren. Dafür befindet sich eine Checkbox im Render Properties-Tab. Sobald Freestyle aktiviert wurde, lassen sich auch detaillierte Einstellungen für den Linieneffekt im View Layer Properties-Tab finden. Die einzelnen Funktionalitäten und Möglichkeiten zur Individualisierung sollen am Ende dieses Kapitels den anderen Methoden zur Erstellung von Outlines gegenübergestellt werden.

Tabelle 3.1: Ergebnisse der verschiedenen Outline Rendering-Methoden

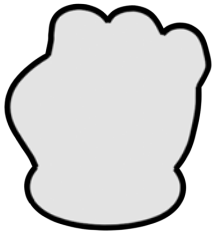
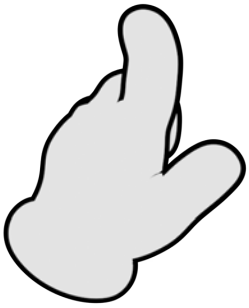
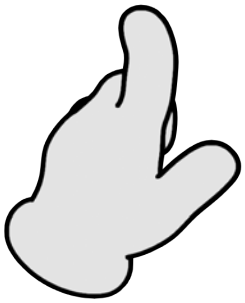
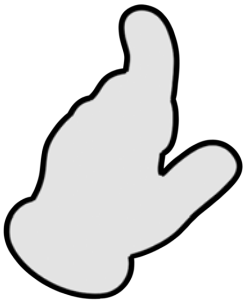
Mesh-Based	**Grease Pencil**	**Freestyle**
		
		
		

Tabelle 3.2: Gegenüberstellung der verschiedenen Outline Rendering-Methoden

Mesh-Based	**Grease Pencil**	**Freestyle**
Problemlose Funktion mit Rig		
kann manuell editiert werden		kann nicht manuell editiert werden
zeichnet Linien ggf. auch in Mesh		analog zu den anderen, kann aber auch nur die Kontur zeichnen
kann normal in Szene betrachtet werden	kann nur aus der Kamera betrachtet werden	kann erst im Rendering betrachtet werden
es muss auf Überlappungen des Outline-Meshs geachtet werden		
ist nicht einfach skalierbar		

Inlines

Die Methoden, welche für die Erstellung von Outlines verwendet wurden, können auch für die Erstellung von Inlines eingesetzt werden. Auch hier sollen wieder vorerst die Methoden kurz beschrieben und anschließend die Ergebnisse gegenübergestellt werden.

Mesh-Based

Die Generierung von Inlines, die auf Meshes basieren, stellt eine recht simple Lösung dar. Die Linien können mit herkömmlichen Modellierungs-Methoden erstellt und auf das Modell angepasst werden. An diesem Punkt kommt jedoch die Kompatibilität mit dem vorher erwähnten Rig zum Tragen. Die Vertices der erstellten Mesh-Inlines sind an dieser Stelle noch nicht an das Hand-Rig gebunden. Es muss also schlussendlich noch ein Deformation-Bone gefunden werden, an den die Mesh-Inlines gebunden werden können,

damit diese sich entsprechend mit dem Hand-Rig bewegen. Dies lässt sich jedoch durch simple Rigging-Techniken umsetzen.

Grease Pencil

Als Option gegenüber der Mesh-Based Inlines können diese auch als Grease Pencil-Objekt an die gewünschte Stelle gezeichnet werden. Es existiert die Möglichkeit, die Grease Pencil-Striche direkt auf die Oberfläche gegebener 3D-Objekte zu zeichnen. Auch hier muss erneut ein Weighting an das Hand-Rig berücksichtigt werden. Grease Pencil-Objekte setzen sich ebenfalls aus einzelnen Punkten zusammen. Diese können analog zu Rigging-Techniken von 3D-Meshes behandelt werden. Dabei ist lediglich zu beachten, dass es Blender-intern keine Option für ein automatisches Weighting dieser Grease Pencil-Punkte gibt (Stand Blender 3.2.2).

Freestyle

Die Freestyle Render Engine bietet leider nur eingeschränkte Möglichkeiten zur Individualisierung der Linien-Effekte. Vom Artist selbst kann dabei nur auf die vordefinierten Linientypen des Freestyle-Renderers zurückgegriffen werden. Alle verfügbaren Optionen sind in der folgenden Abbildung dargestellt. Diese sind aber vorrangig daran orientiert, die Linien anhand der bestehenden Geometrie des Meshes zu generieren. Es können also Edges des Meshes genau markiert werden, welche als Linie gerendert werden sollen. Es können jedoch Linien auch nur da gerendert werden, wo eine Mesh-Edge ist, um das zu ermöglichen.

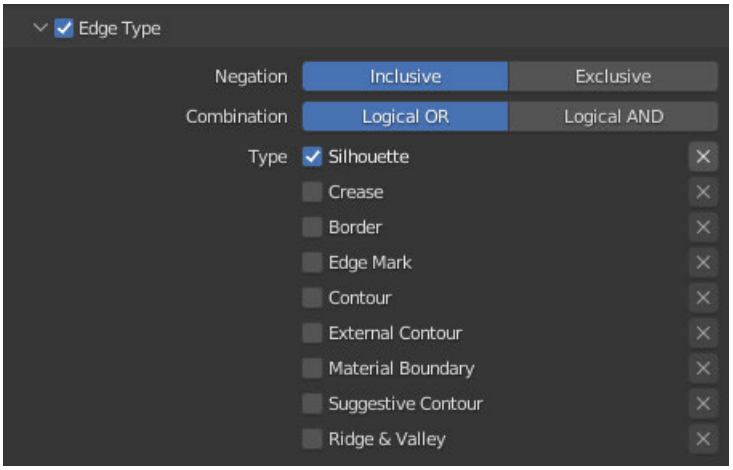


Abbildung 3.4: Verfügbare Edge-Types des Freestyle-Renderers

Tabelle 3.3: Ergebnisse der verschiedenen Inline Rendering-Methoden

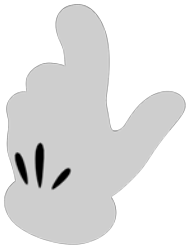
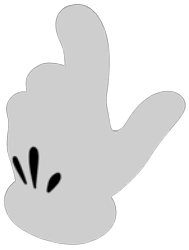
Mesh-Based	**Grease Pencil**	**Freestyle**
		

Tabelle 3.4: Gegenüberstellung der verschiedenen Inline Rendering-Methoden

Mesh-Based	Grease Pencil	Freestyle
kann komplett individuell gezeichnet werden		wird hauptsächlich entlang von Mesh Edges gezeichnet
verschiedene Liniendicke entlang des Verlaufs einer Linie ist einfach variabel zu gestalten		variable Liniendicke muss über Verlaufsgraphen gesetzt werden
muss von Hand gezeichnet bzw. erstellt werden		kann über Edge Types generiert werden

Cross Hatching

Cross Hatching, zumindest in der Art, wie es bisher besprochen und dargestellt wurde, lässt sich mit den vorhergehend thematisierten Methoden nicht umsetzen. Zur Umsetzung des Cross Hatching-Effekts kann jedoch ein Shader erstellt werden. Die vorgestellte Funktionsweise basiert auf einer Methode von Kristof Dedene, welche er auf seinem YouTube-Kanal veröffentlichte. Das grundlegende Prinzip zur Erstellung eines Cross Hatching-Shaders soll folgend erklärt werden. Eine vollständige Abbildung des verwendeten Shader-Graphen befindet sich im Anhang dieser Arbeit unter Cross Hatching Shader.

Des Weiteren soll hier auch für eine Erklärung des Shaders auf die Online-Quelle von Kristof Dedene selbst verwiesen werden.[5]

Durch eine Kombination aus prozeduralen Texturen, lässt sich ein Shader erstellen, welcher parallele Linien abbildet. Grundlage dafür bietet die sogenannte "Wave Texture". Es lässt sich sogar durch einen Parameter der Wave-Textur, die Distorsion der Linien festlegen.

[5]Dedene, Kristof (2020): Tutorial: Procedural hatching and manga shaders for EEVEE Blender

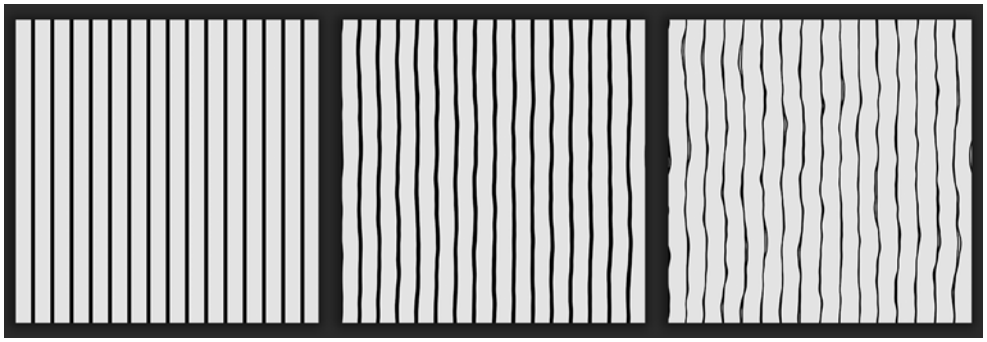


Abbildung 3.5: Prozedural durch Wave-Textur generierte Linien mit verschiedenen Distorsionen

Über die Kombination sogenannter "Voronoi Textures" lassen sich schon kleine Flächen abgegrenzt darstellen, bei welchen sogar der Abstand zueinander festgelegt werden kann. Blenders Shading Interface verfügt außerdem über eine Mapping Node. Über diese können Texturen in ihren grundlegenden Transformationen, also Translation, Rotation und Skalierung, verändert werden. Durch gezielte Verwendung dieser Mapping Node lassen sich die vorher verwendeten Voronoi-Texturen leicht im Hinblick auf ihre Translation verändern. Das hat eine Veränderung der Ausrichtung der Parallellinien zur Folge. Mithilfe einer Mix-Node lassen sich jetzt zwei verschiedene solcher Rotationsvariationen überlagern.

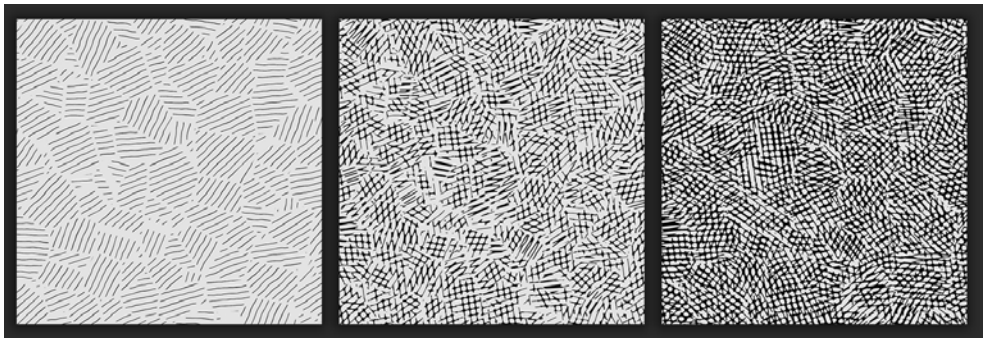


Abbildung 3.6: Aufbau der Cross Hatching-Linien in mehreren Ebenen

Da diese über eine unterschiedliche Rotation verfügen, kommt es zu einer gekreuzten Darstellung der einzelnen Linien, wodurch schon grundlegend der Effekt, welcher in 2.2.1 Cross Hatching beschrieben wurde, erzielt wird. Diese Veränderung der Rotation

und Überlagerung kann dabei nach Belieben häufig wiederholt werden. Je mehr Linien-Texturen überlagert werden, desto dichter wird später die Cross Hatching-Textur. An diesem Punkt liegt der Cross Hatching-Effekt gleichmäßig über das gesamte Modell verteilt. Diese Technik soll jedoch dafür verwendet werden, ein 3D-Modell stilisiert zu beleuchten bzw. zu schattieren. Dafür muss im finalen Schritt eine Interaktion der erstellten Textur mit Lichtquellen umgesetzt werden. Dafür kann beispielsweise der "Diffuse BSDF" Shader verwendet werden. Der Diffuse BSDF Shader ist ein Blender-interner Shader und dient grundlegend für das Rendern diffuser Oberflächen, bei denen die jeweilige Rauheit als Parameter gesetzt werden kann. Der Shader bildet also die Beleuchtung eines bestimmten Objekts ab.

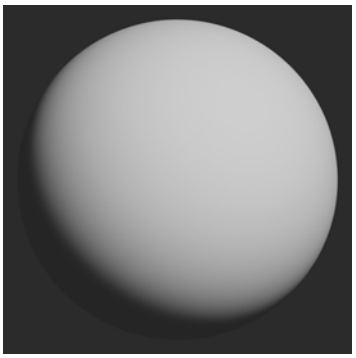


Abbildung 3.7: Ergebnis des Diffuse BSDF mit Beleuchtung durch ein Sun-Objekt

Die Eevee Render Engine von Blender bietet nun die Option, diesen Shader in Farbinformationen zu konvertieren. Das eröffnet im weiteren Verlauf die Möglichkeit, den Shader für die Maskierung der Cross Hatching-Textur an beleuchteten Stellen zu verwenden. Speziell werden jetzt die Lichtinformationen des Shaders mit der Cross Hatching-Textur über eine Mix-Node mit dem Interpolationsmodus "Linear Light" kombiniert. Dabei entsteht eine sehr gerade Kante. Diese kann ebenfalls mit Distorsion versehen werden, indem die Lichtinformationen mit einer Noise-Textur kombiniert wird.

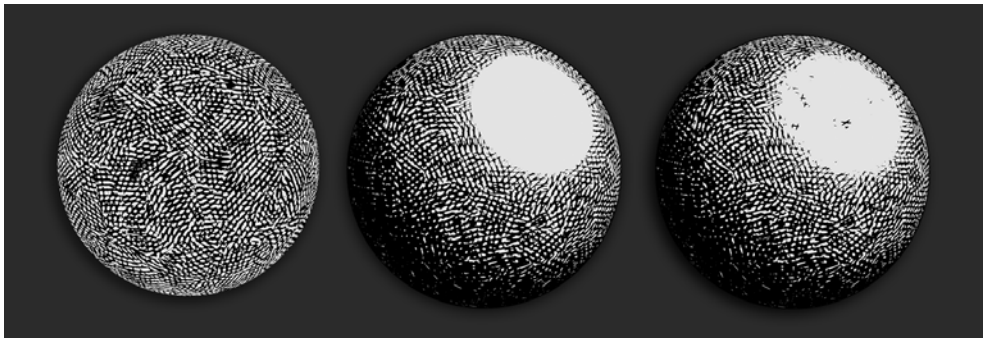


Abbildung 3.8: Vergleich der Lichtadaptionen: ohne Lichtmaske (links), mit Lichtmaske (mittig), Lichtmaske mit Noise (rechts)

3.2.2 Grundlegende Beleuchtungstechniken für NPR

Dieses Unterkapitel befasst sich mit einigen weiteren grundlegenden Shadern, sowie zukünftig verwendeten Blender-Objekten und -Einstellungen.

Unlit

Die visuelle Abbildung eines Unlit-Objekts wurde bereits in Kapitel 2.2.2 Unlit beschrieben. Dieser Effekt lässt sich innerhalb von Blender ebenfalls äußerst simpel erzielen. Jeder Blender-Shader bzw. jedes Material verfügt über eine Material Output Node. An diese können Informationen übergeben werden, welche dann auf die Oberfläche der Objekte, welchen das Material zugewiesen wurde, gerendert wird. Ein Unlit Shading kann erzielt werden, indem an den Surface Output dieser Node, keine Shader-Informationen, sondern lediglich ein RGB-Wert übergeben wird.

Directional Lighting

Wie in 2.2.2 Direct Illumination erklärt wurde, finden sich in animierten Zeichnungen oft Lichtquellen, die nur aus einer Richtung ein jeweiliges Objekt beleuchten. In Blender wird im Rahmen dieser Arbeit zum Festlegen dieser Lichtrichtung vorrangig das Sun-Objekt verwendet. Dieses kann Modelle über die gesamte Szene hinweg aus einer gewählten Richtung beleuchten. Alleinstehend reicht dieses Objekt jedoch nicht aus, um einen 2D

Art Style umzusetzen. Dieses Licht-Objekt wird deswegen vorrangig für die Kontrolle der Lichtinformation von Shadern verwendet, welche dann einen 2D Art Style imitieren.

Cel Shading

Der Cel Shading-Effekt kam dabei schon in Kapitel 2.2.2 Cel Shading zur Sprache. Ein sehr grundlegender Cel Shader lässt sich in Blender mit wenigen Shader-Nodes erstellen. Dafür wird zuerst die Lichtinformation eines Blender Shaders, wie etwa dem "Diffuse BSDF" oder dem "Principled BSDF" in Farbinformationen konvertiert. Wichtig dabei ist, dass dieser Schritt (Stand: Blender 3.2.2) nur mit der Eevee Render Engine funktioniert. Die konvertierten Lichtinformationen werden dann an eine Color Ramp übergeben. Bei dieser Color Ramp muss daraufhin der Interpolationsmodus geändert werden. Standardmäßig steht dieser auf Linear, was nur eine sehr graduelle Abbildung der Werte zur Folge hat. Die Interpolation kann jedoch ebenfalls auf Konstant umgestellt werden. Dadurch werden die Werte, welche kleiner als ein festgelegter Schwellwert sind, in einer bestimmten Farbe gerendert und alle darüber liegenden in einer weiteren bestimmten Farbe. Die Anzahl an Zwischenstufen und deren Schwellwert können dabei individuell gewählt werden.

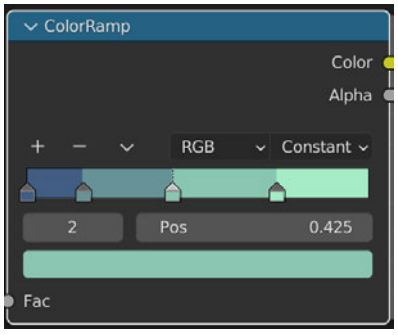


Abbildung 3.9: Beispielhafte Einstellung der Color Ramp Node

Bloom

Der Bloom-Effekt lässt sich in Blender im Render Properties-Tab durch einen einfachen Haken aktivieren. Wichtig dabei ist jedoch, dass dieser Effekt auch nur dann erkennbar ist, wenn eine den Einstellungen entsprechend genügend starke Lichtquelle gerendert

wird. Die Emission von Licht lässt sich in Blender zum Beispiel durch einen Emission-Shader, den "Emission BSDF", umsetzen. Die Implementation des Bloom-Effektes wäre jedoch möglicherweise direkt in der Unity Engine geeigneter. Üblicherweise wirkt dieser Effekt besser, wenn er erst zur Laufzeit des Mediums generiert wird und nicht schon in den Animations-Frames festgelegt ist. So kommt es beispielsweise zu einer authentischeren Darstellung, wenn ein sehr helles Objekt von einem anderen verdeckt wird. Das Bloom-Objekte würde bei der Echtzeit-Implementation noch an den Kanten des verdeckenden Objekts scheinen, während es im anderen Fall direkt abgeschnitten würde. Bei der Umsetzung mit Unity wird der Bloom-Effekt jedoch im Hinblick auf die Post Processing-Effekte betrachtet.

3.2.3 Farbe

Das folgende Kapitel fasst die technische Umsetzung der Teilaspekte aus dem Bereich Farbe zusammen.

Farbspektrum

Das verwendete Farbspektrum lässt sich zu sehr großen Teilen durch den 3D-Artist steuern. Vor allem bei den unter Kapitel 3.2.2 Grundlegende Beleuchtungstechniken für NPR besprochenen Shadern lässt sich die später gerenderte Farbe direkt als RGB-Wert durch den Artist festlegen. Im Falle von Directional Lighting bietet die schon erwähnte Color Ramp Node eine tolle Möglichkeit, das abgebildete Farbspektrum einzugrenzen. Durch diese lassen sich die Farbwerte sehr schnell in ein kleineres Spektrum abbilden. Zur simplen Visualisierung dieser Vorgehensweise wird an dieser Stelle eine Kugel betrachtet, welche mit einem einfachen Directional Light gerendert wird. Wie üblich können in der Eevee Render Engine, die Lichtinformationen nun in Farbinformationen konvertiert werden. Durch eine Color Ramp Node lassen sich jetzt diese Farbinformationen auf ein selbst gewähltes Farbspektrum abbilden. Zur Verdeutlichung des Effekts wird in der folgenden Abbildung nur ein Schwarz-Weiß-Bild dargestellt. Bei der somit ausschließlichen Betrachtung der Farbwerte, also der Helligkeit, lässt sich die Verringerung des abgebildeten Farbspektrums deutlich erkennen.

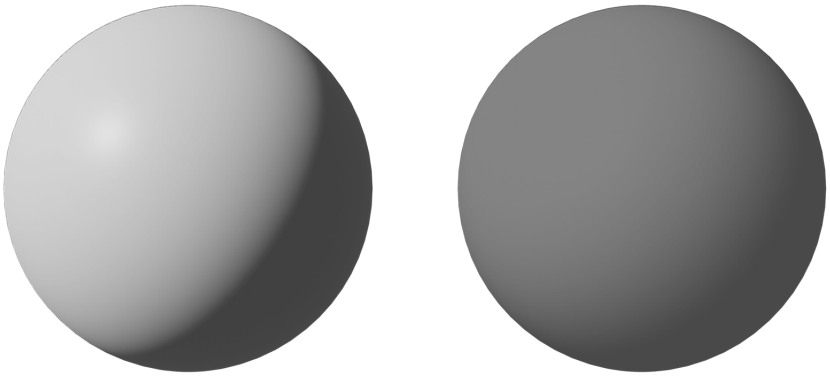


Abbildung 3.10: Begrenzung des Farbspektrums durch die Color Ramp Node

Ein ähnlicher Effekt lässt sich ebenfalls mit der Verringerung des Bildkontrastes erzielen. Dafür ist wichtig zu erwähnen, dass sich im Render Properties-Tab unter Color Management, Einstellungen für den erzeugten Bildkontrast finden lassen. Diese Option bietet jedoch etwas weniger präzise Einstellungsmöglichkeiten, da hierdurch nicht genau die Farbwerte an den jeweiligen Enden des Spektrums festgelegt werden können.

Farbquantität

Die Anzahl der verwendeten Farben ist wie auch das Farbspektrum ein Aspekt, welcher vorrangig vom 3D-Artist selbst verwaltet werden muss. Eine große Hilfe stellt dabei die Verwendung von Material-Instanzen dar. Ein Material kann dabei mehreren 3D-Objekten zugewiesen werden. Das garantiert zum einen, dass diese zugewiesenen Objekte genau dieselbe Farbe darstellen, zum anderen bringt es eine große Zeitersparnis bei der späteren Anpassung der jeweiligen Farbe mit sich. Diese muss dadurch nur einmal im Material angepasst werden und wird danach direkt für alle 3D-Objekte, welche dieses Material verwenden, übernommen.

Verlauf der Farben

Auch für die Steuerung des Verlaufs der Farben kann erneut die divers einsetzbare Color Ramp Node genutzt werden. Durch das Verschieben der Farbregler und das Ändern der Interpolationsmodi kann genau eingestellt werden, wie abrupt ein Übergang von Farben geschehen soll oder auch, ob dieser hart abgeschnitten wird. In der folgenden Abbildung sollen beispielhaft drei Optionen abgebildet werden, wie durch Einsatz der Color Ramp Node der Verlauf von Farben ineinander kontrolliert werden kann. Für eine umfangreiche Vorstellung der Farbkontrolle ist diese Abbildung nun auch mit vollwertigen Farbinformationen, also auch mit Farbton und Sättigung gerendert.

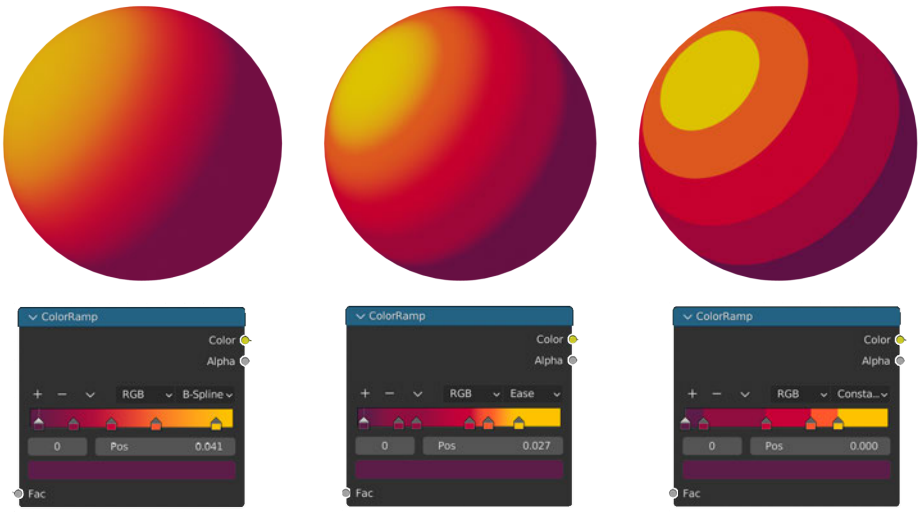


Abbildung 3.11: Drei verschiedene Optionen für Farbverläufe

Leinwand-Textur

Wie in Kapitel 2.2.5 Leinwand-Textur beschrieben, kann es bei manchen analogen Farbstoffen dazu kommen, dass im finalen Bild die Beschaffenheit des jeweiligen Untergrunds noch zu erkennen ist. Dieser Effekt lässt sich mithilfe eines Shaders visualisieren. Dafür wird lediglich eine Textur benötigt. In besonderen Anwendungsfällen könnte die gewünschte Oberflächenbeschaffenheit auch prozedural generiert werden. Um den Rahmen dieser Arbeit zu wahren, wird dieser Schritt jedoch nicht genauer thematisiert. Eine beliebige Leinwand Textur lässt sich dann innerhalb eines Shaders über die MixRGB Node mit anderen Farb-Inputs zusammenführen. Diese kann direkt auf die Geometrie

des Objekts projiziert, aber auch immer frontal zur Kameraansicht über das Objekt gelegt werden.

3.3 Umsetzung der benannten Stilelemente eines 2D Animation Styles

In dem folgenden Unterkapitel soll nun auch die technische Umsetzung der bereits besprochenen Stilelemente eines Animation Styles erklärt werden.

3.3.1 Frame Rate

Die Einstellung der Frame Rate selbst ist sehr simpel. Unter Blenders Output Properties-Tab lässt sich ein Reiter für das Festlegen dieser finden. Neben der Auswahl aus üblich verwendeten Frame Rate-Formaten, kann hier auch eine individuelle Frame Rate gesetzt werden. Dabei sollte beachtet werden, dass diese bestmöglich vor der Animation gegebener Objekte festgelegt werden sollte, da andernfalls die Animationsgeschwindigkeit von der Änderung betroffen wird. Da sich diese Arbeit an den 1930er-Cartoons bzw. an Cuphead orientiert, wird von einer Frame Rate von 24fps ausgegangen.

Animation On Twos

In Kapitel 2.3.1 Frame Rate wurde bereits thematisiert, dass in einigen Produktionen nur jeder zweite Frame animiert wird. Dieser Effekt lässt sich sogar noch nach der Erstellung einer etwaigen 3D-Animation umsetzen. Dafür wird exemplarisch davon ausgegangen, dass eine Animation in 24fps erstellt wurde. Zur besseren Darstellung wird eine beispielhafte Animation eines springenden Balls betrachtet. Simplifiziert kann die gemeinte Bewegungsabfolge in der nächsten Abbildung gesehen werden.

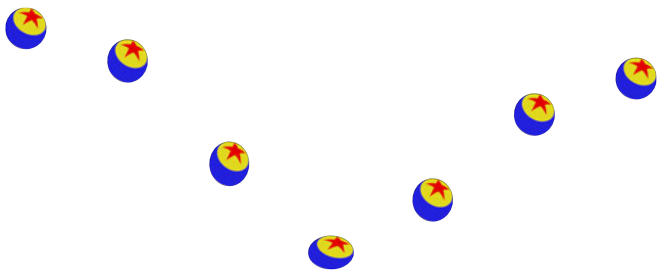


Abbildung 3.12: Bildweise Darstellung der Sprunganimation

Nun wird die Bewegung dieses Balls auf der vertikalen Achse betrachtet. Diese sieht für die Sprunganimation wie in der folgenden Abbildung dargestellt aus.

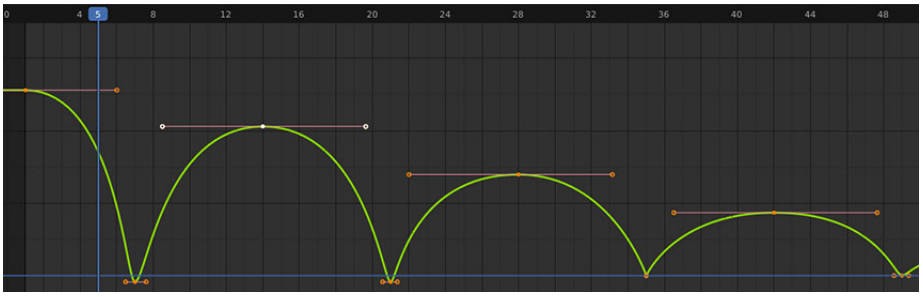


Abbildung 3.13: Abbildung des interpolierten Graphen

Es ist zu erkennen, dass nur wenige Keyframes gesetzt wurden und die übrigen Positionen des Balls durch eine Interpolationsmethode errechnet werden. Damit der gewünschte Animation On Twos-Effekt erzielt werden kann, muss diese interpolierte Animation vorerst durch ein Baking zwischengespeichert werden. Spezieller wird dadurch die jeweilige Position des Balls auf dem dazugehörigen Frame gespeichert.

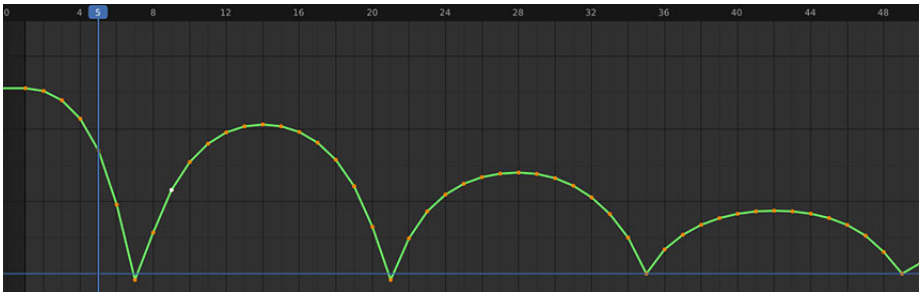


Abbildung 3.14: Baking der Animationsdaten

Jetzt hat jedoch jeder der 24 Frames einen Wert. Eigentlich sollen aber lediglich zwölf Frames existieren, welche jeweils für zwei Frames lang auf dem Bildschirm bleiben. Um die dafür überflüssigen Informationen zu löschen, kann ein simpler Trick verwendet werden. Dafür werden die Keyframes um den Wert 0.5 auf der x-Achse des Animationsgraphen herunterskaliert. Dadurch wird jeder zweite Wert durch den jeweils nächsten überschrieben. Danach werden die Keyframes mit dem Wert 2 wieder hochskaliert, um das ursprüngliche Animations-Timing wieder zu erhalten. Die Animation ist jetzt wieder dieselbe, nur dass nun jeder zweite Frame gelöscht wurde.

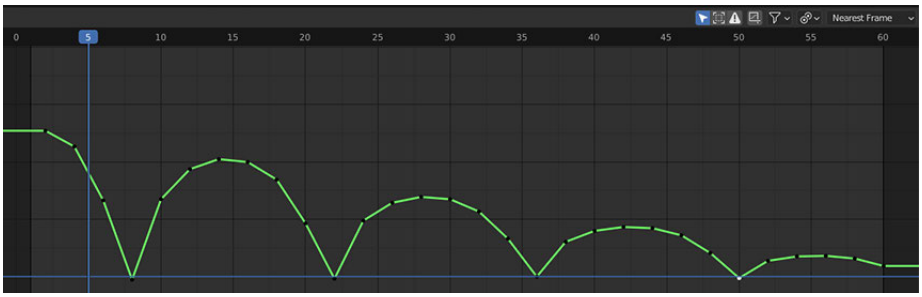


Abbildung 3.15: Überschreiben jedes zweiten Animationsframes

Zuletzt muss dafür gesorgt werden, dass jeder Animations-Frame für zwei Frames auf dem Bildschirm verweilt. Dafür genügt es, den Interpolationsmodus des Animationsgraphen auf Konstant zu stellen. Wie in der letzten Abbildung zu erkennen ist, verweilt ein bestimmter Animations-Frame jetzt zwei Frames lang auf demselben Wert, bevor dieser sich ändert. Dadurch wurde der Effekt einer handgezeichneten Animation On Twos simuliert.

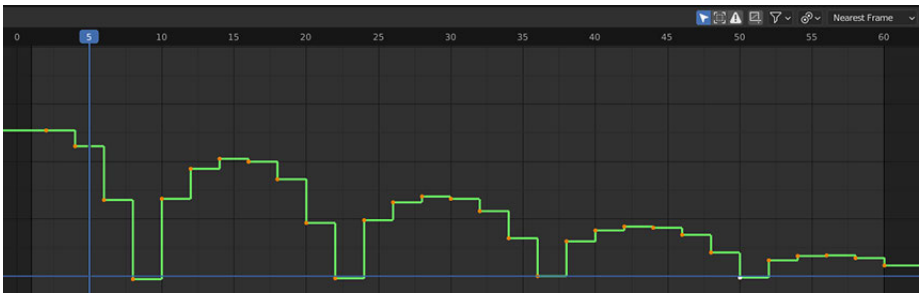


Abbildung 3.16: konstanter Interpolationsmodus

3.3.2 Rigging-Besonderheiten

Für eine detailgetreue Nachstellung eines Cartoon-Looks benötigt das verwendete Charakter-Rig Möglichkeiten, welche von üblichen Arbeitsweisen eines Charakter-Rigs abweichen. Die zu beachteten Eigenschaften wurden bereits in Kapitel 2.3 Merkmale eines 2D Animation Styles thematisiert. Folgend soll auf grundlegende Möglichkeiten zur Umsetzung dieser Eigenschaften eingegangen werden.

Duplicates und Smearing

Innerhalb der 3D-Umgebung kann sich für die Erstellung des Duplicates-Effekts, wortwörtlich der Duplizierung bestimmter Objekte, bedient werden. So können ganze Körperteile des Charakters, ggf. sogar mit dazugehörigem Rig dupliziert werden. Weiterführend muss dieser duplizierte Part nur am richtigen Frame entsprechend posiert werden.

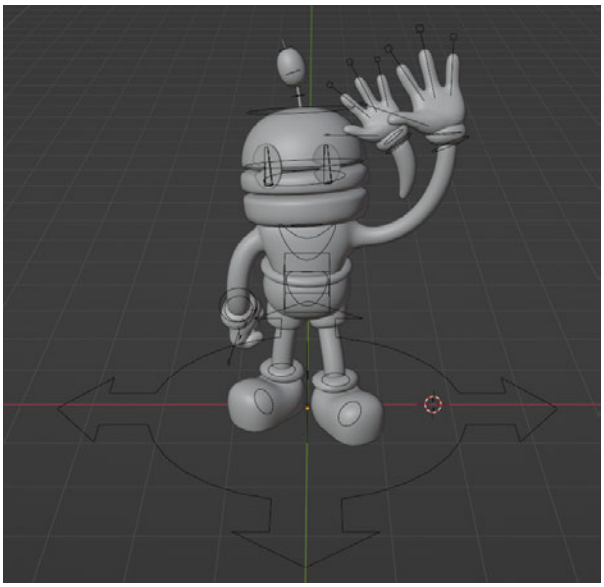


Abbildung 3.17: Beispielhafter Duplicate-Effekt mit dupliziertem Arm und Rig

In Blender lässt sich die Sichtbarkeit eines gegebenen Objekts mithilfe von Keyframes kontrollieren. Das duplizierte Körperteil kann also nur für die Frames sichtbar gemacht werden, für die das Duplicate bzw. die Duplicates auch zu sehen sein sollen. Andernfalls kann auch simpel die Skalierung auf 0 gesetzt und nur für den aktiven Frame auf 1 geändert werden. Diese Technik wäre für den Export in eine Game Engine geeigneter.

Für den Anwendungsfall dieser Arbeit funktionieren beide Methoden, da die Animation innerhalb von Blender gerendert und erst danach in die Game Engine exportiert wird. Eine weitere Möglichkeit wäre, die Transparenz des Duplicates zu animieren. Auf einem Material kann, insofern vorhanden, die Transparenz ebenfalls durch Keyframes gesteuert werden. So ließe sich sogar ein Verblassen des oder der Duplicates darstellen.

Rubber Hose Animation Style

Um einen Rubber Hose Animation Style nachstellen zu können, muss eine schlauchartige Bewegung vor allem bei den Gliedmaßen animierbar sein. Dafür können sehr hilfreiche Konfigurationen schon bei der Erstellung des Rigs getroffen werden. Die hauptsächliche Herausforderung ist dabei, wie schon benannt, die gummiartige, weiche Bewegung der Gliedmaßen eines Charakters. Blender bietet eine sehr praktische Funktion für Deformation-Bones. Durch sogenannte "Bendy Bones", kann jeder Deformation-Bone in beliebig viele Untersegmente unterteilt werden. Diese Segmente stellen dann eine simple Bezierkurve nach. Die folgende Abbildung zeigt das exakt gleiche Rig. Im Gegensatz zur linken Seite, verwendet die rechte Seite lediglich die Bendy Bones-Option, wobei in diesem Fall jeder einzelne Deformation-Bone in 16 Untersegmente unterteilt wurde. Durch die Bendy Bones-Funktion kann somit die schlauchartige Bewegung der Gliedmaßen gewährleistet werden.

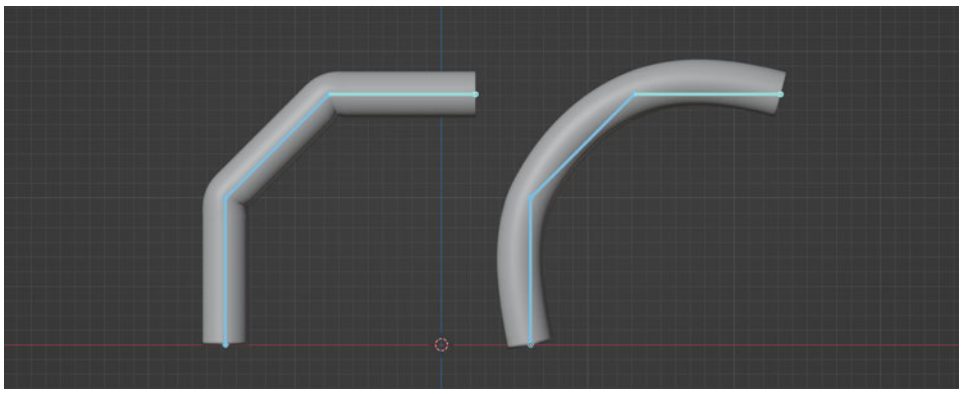


Abbildung 3.18: Vergleich Rig ohne Bendy Bones (links) mit einem Rig mit Bendy Bones (rechts)

3.4 Finale Komposition in einer Game Engine

Um dieses Kapitel abzuschließen, soll zuletzt auf die Umsetzung der Szenenmerkmale, welche in Kapitel 2.4 Merkmale des Szenenaufbaus in einem 2D-Videospiel bereits erläutert wurden, eingegangen werden. Wie aus der Überschrift bereits zu entnehmen ist, werden die folgend vorgestellten Techniken innerhalb der Unity Engine realisiert.

3.4.1 Parallax-Effekt

Für die Implementation des Parallax-Effekts, wie er in Kapitel 2.4.1 Parallax-Effekt schon erklärt wurde, müssen verschiedene Tiefenebenen erstellt werden. Folglich müssen sich Hintergrundelemente auf einem abgetrennten Bild zu Vordergrundelementen befinden. Dabei können so viele Ebenen angelegt werden, wie es für eine authentische Darstellung des Effekts notwendig ist. Über Scripts kann nun die Bewegung jeder einzelnen dieser Ebenen in Relation zur Kamerabewegung gesteuert werden. Dafür wird beispielhaft davon ausgegangen, dass eine Fokusebene existiert. Auf dieser werden der Spieler-Charakter und andere Gameplay-relevante Objekte abgebildet. Darüber hinaus existieren natürlich noch die vorher benannten Ebenen für Hintergrund und Vordergrund. Des Weiteren wird von einer beweglichen Kamera ausgegangen, welche entlang der x-Achse bewegbar ist.

Bei der Implementation des Parallax-Effekts müssen nun die relativen Geschwindigkeiten der Ebenen zu dieser Kamerabewegung festgelegt werden. Dies lässt sich zum Beispiel darüber steuern, dass diese Ebenen über ein Script mit derselben Bewegungsgeschwindigkeit der Kamera transformiert werden. Über einen Multiplikator lassen sich dann Abstufungen davon einstellen. Mit einem Multiplikator von 1 bewegt sich eine Ebene somit identisch zur Kamera und scheint dadurch unverändert im Bild zu bleiben. Mit einem Multiplikator zwischen 0 und 1 bewegt sich eine Ebene entsprechend langsamer in Relation zur Kamera. Bei einem Multiplikator von genau 0 bleibt eine Ebene unbeeinflusst von diesem Parallax-Effekt. Der Wert 0 wird bei der Fokusebene gesetzt. Bei einem negativen Wert bewegt sich eine Ebene in die entgegen gesetzte Richtung der Kamera und scheint sich somit schneller als die Fokusebene zu bewegen. Das kommt bei Objekten, welche vor der Fokusebene gelegen sind, zum Einsatz.

3.4.2 Post Processing-Effekte

Die Unity Engine bietet eine vorgefertigte Auswahl an Post Processing-Effekten kostenfrei, direkt über das gleichnamige "Post Processing" Package an. Über dieses Package lassen sich durch simple Einstellungen, die in Kapitel 2.4.2 Post Processing-Effekte benannten Effekte implementieren. Darunter existiert ein Effekt zum Rendern von Chromatic Aberration, Vignette, Grain und auch zur Simulation einer Linsenverzerrung. Darüber hinaus lassen sich neben weiteren Effekten auch vielseitige Anpassungen an der Farbdarstellung machen.

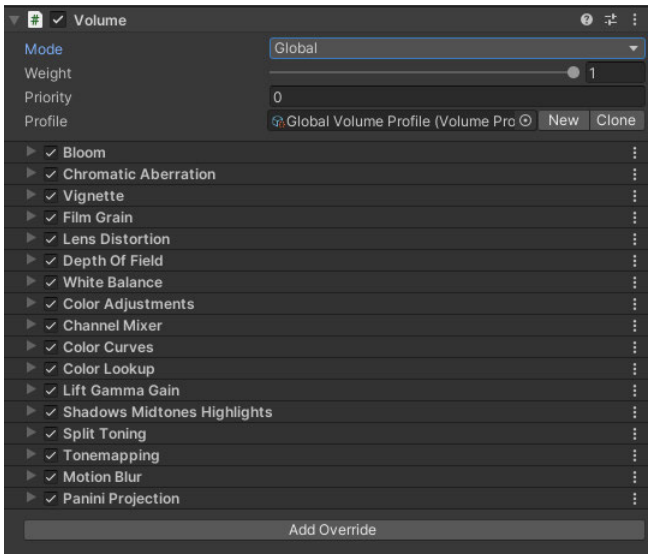


Abbildung 3.19: Alle Post Processing Overrides des Post Processing Packages (Version 3.2.2)

Overlays

Der Grain Post Processing-Effekt lässt sich ebenfalls dafür verwenden, eine individuelle Textur über das Kamerabild zu legen. Mit einer speziellen Film Grain-Textur können somit auch die gröberen Unreinheiten, welche 1930er-Cartoons aufweisen, simuliert werden.

4 Praktische Nachbildung des Stils

In diesem Kapitel werden die theoretisch vorgestellten Methoden praktisch evaluiert. Wie in der Aufgabenstellung schon beschrieben, ist es das Ziel, den Art Style von Cuphead zu imitieren. Dafür wird vorerst der Art Style selbst, unter zu Hilfenahme der in 2.2 Stilelemente eines 2D Art Styles aufgeführten Stilelemente, analysiert. Am Ende davon soll eine Aufschlüsslung entstehen, in der das Vorkommen und Aussehen der einzelnen aufgelisteten Stilelemente speziell im Fall von Cuphead dargestellt wird. Das dient zum einen der besseren Identifikation, was genau umgesetzt werden muss, unterstützt jedoch auch die schlussendliche Evaluation der Ergebnisse. Selbstverständlich kann im Rahmen dieser Arbeit nicht das gesamte Spiel Cuphead nachempfunden werden. Folglich wurde eine Auswahl an diversen Komponenten eines Videospiels getroffen, welche in verschiedenen Bereichen die Vorgehensweise beleuchten soll. Dafür wird vorerst ein Charakter im Stil von Cuphead modelliert und dann entsprechend der Stilelemente, welche in der Analyse herausgestellt wurden, stilistisch angepasst. Dieser Charakter soll ebenfalls mit einem Rig versehen werden, um diesen in drei grundlegenden Bewegungen zu animieren. Diese Bewegungen sollen eine Stand-Animation, Lauf-Animation und Dash-Animation darstellen. Neben dem übergreifenden visuellen Eindruck soll mit dieser Auswahl an Animationen vor allem auch die Nachstellung der Animations-Stilelemente aus Cuphead überprüft werden. Des Weiteren soll eine Plattform, auf welche der Spieler springen kann und ein animierter VFX-Effekt entstehen. Neben den Gameplay-Elementen werden ebenfalls Elemente zur Darstellung des Hintergrunds und des nicht interaktiven Vordergrunds erstellt. Diese sind vor allem deshalb interessant, weil sie in einem anderen Art Style dargestellt sind. Zuletzt sollen dann alle erstellten Elemente in der Unity Engine zusammengeführt und mit den aus Cuphead herausgestellten Post Processing-Effekten ergänzt werden. In einem weiteren Unterkapitel werden dann die finalen Ergebnisse aus dieser praktischen Nachbildung präsentiert und folgend mit Hinblick auf den zuvor analysierten Stil evaluiert.

4.1 Analyse der spezifischen Stilelemente eines Rubber Hose Animation Styles

Um für die folgende Analyse eine bessere Nachvollziehbarkeit gewährleisten zu können, basieren alle vorgestellten Ergebnisse auf einer umfangreichen Referenzübersicht. Diese lässt sich, in entsprechende Abschnitte unterteilt, im Anhang unter der Punkt Referenzübersicht einsehen. Cuphead selbst bringt den nostalgischen Stil der sogenannten Rubber Hose Animation [1] in die Neuzeit. Folglich ist es nur logisch, neben Cuphead als direkte Referenz auch das übergeordnete stilistische Genre zu betrachten. Bei vielen Animationen im Rubber Hose Style, aber auch generell moderneren Cartoons, fällt ein charakteristisches Merkmal besonders auf. Die Charaktere und animierten Figuren bzw. Gameplay-relevanten Objekte erstrahlen oft in einem anderen Art Style als der Hintergrund. Da Hintergründe meist über ganze Szenen hinweg unverändert bleiben, während die animierten Charaktere sich jeden Frame ändern, sind die Hintergründe oft viel aufwendiger gezeichnet, als es die animierten Charaktere sind. Das äußert sich vor allem durch eine größere Farbvariation und die Darstellung komplexerer Schattierungen sowie anderer kleinerer Details.

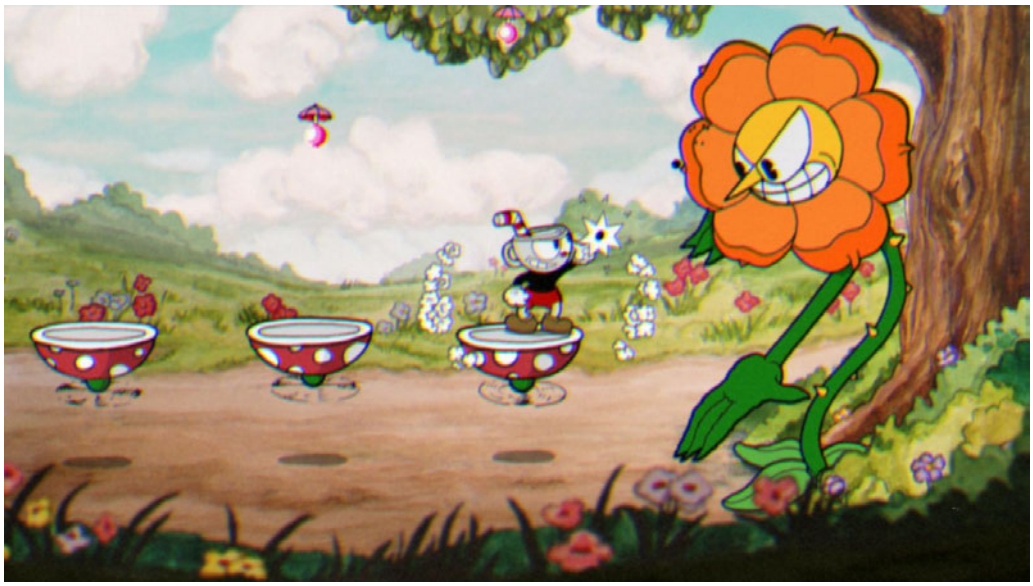


Abbildung 4.1: Verschiedene Art Styles für Hinter- und Vordergrundelemente in Cuphead

[1] McGowan, David (2019): Cuphead: Animation, the public domain, and home video remediation. Journal of Popular Culture, S. 4

Dieses Prinzip wird im Art Style von Cuphead ebenfalls sehr deutlich abgebildet, wie in der vorangegangenen Abbildung zu sehen ist und muss deswegen auch unbedingt bei der Imitation des Art Styles beachtet werden. Aus diesem Grund wird in den folgenden Unterkapiteln zwischen der Rekreation des Art Styles der statischen Objekte und dem der animierten bzw. Gameplay-relevanten Objekte unterschieden. Die gummiartige Bewegung der Gliedmaßen wurde schon zuvor näher erläutert und ist auch in Cuphead zu finden. Zudem lässt sich auch die Verwendung von Squash and Stretch besonders in den Animationen identifizieren.

4.1.1 Stilelemente der Gameplay-relevanten Objekte

Wie vorher erklärt, soll in diesem Unterkapitel der Art Style der Gameplay-relevanten Objekte aus Cuphead isoliert betrachtet werden. Dabei wird auf die in Kapitel 2.2 Stilelemente eines 2D Art Styles und Kapitel 2.3 Merkmale eines 2D Animation Styles aufgeführten Stilelemente zurückgegriffen.

Line Art

Der Begriff Line Art steht hier erneut übergeordnet für die Bereiche Outlines, Inlines und Cross Hatching.

Abbildung 4.2: Der Spieler-Charakter und ein Gegner aus Cuphead

Bei der Betrachtung der Ingame-Aufnahme aus dem Spiel, fällt an den animierten Charakteren auf, dass eine dickere Outline um die Silhouette herum existiert. Kleine Details, die zur Formgebung beitragen, sind durch Inlines dargestellt. Diese finden sich zum

Beispiel als ikonische Striche der typischen Cartoon-Handschuhe, aber auch zur besseren Darstellung der Schuhform und sehr häufig zur Abgrenzung verschiedener Farbflächen. Es lassen sich folglich Outlines und Inlines identifizieren. Dabei ist es wichtig zu vermerken, dass die Inlines deutlich dünner gegenüber den Outlines gezeichnet wurden. Zuletzt ist anzumerken, dass keine Form des Cross Hatchings zu erkennen ist.

Beleuchtung

Bei der Beleuchtung der Gameplay-relevanten Objekte fällt direkt auf, dass diese zu einem Großteil mit dem Unlit Shading-Modell dargestellt werden. Besonders deutlich ist das bei den spielbaren Charakteren. Auf dem Charakter selbst ist kein Schatten abgebildet. Dieser wird lediglich durch abgegrenzte Farbflächen mit einem festen Farbwert dargestellt. Ein Schatten wird jedoch unterhalb der Charaktere in Form einer etwas dunkleren Ellipse angedeutet.

Abbildung 4.3: Elliptischer Schatten unter dem Charakter in Cuphead

Abseits der Charaktere lassen sich bei anderen Gameplay-relevanten Objekten vereinzelt leichte Schatten erkennen, wobei jedoch noch ein anderer Effekt zur Andeutung von Licht verwendet wird. Auf Objekten, welche glänzend dargestellt werden sollen, befinden sich Highlights. Auf einer bestimmten Farbfläche befindet sich dabei meist genau ein solches Highlight. Diese sind, wie andere Farbflächen, hart abgegrenzte Formen und folgen angedeutet der Oberfläche des unterliegenden Objekts.

Abbildung 4.4: Highlights (markiert durch weiße Pfeile)

Besonders im Hinblick auf die Gameplay-relevanten Objekte lässt sich auch die Verwendung des Bloom-Effekts feststellen. In der nachfolgenden Abbildung sind zwei spezielle Beispiele für die Verwendung dieses Bloom-Effekts in Cuphead dargestellt.

Abbildung 4.5: Bloom-Effekte in Cuphead

Farbe

Die Charaktere selbst haben eine sehr überschaubare Farbquantität. Die Farbpalette eines einzelnen Charakters setzt sich dabei üblicherweise aus einer einstelligen Anzahl aus Farben zusammen. Es tauchen keine Variationen des Farbtons auf. Ein Charakter wird also nur in einem beispielsweise speziellen Rotton gezeichnet, welcher gegebenenfalls an mehreren Stellen auftaucht. Die Farben werden darüber hinaus in sehr klar abgegrenzten Flächen dargestellt und laufen somit nicht ineinander. Manche visuellen Effekte weichen jedoch davon ab. Wie in der Abbildung zu sehen, wird bei dem Feuereffekt ein Farbverlauf

verwendet. Im Gegensatz zur vorher beschriebenen Farbverwendung lassen sich hier also mehrere ähnliche Farbtöne finden, welche auch ineinander übergehen.

Abbildung 4.6: Ausschnitt aus Cuphead

Leinwand-Textur

In den Grafiken der Gameplay-relevanten Objekte lässt sich selbst keine Textur erkennen. Die Farbe wird nur durch die, in Kapitel 4.1.1 Farbe erklärten Teilaspekte definiert. Eine geringe Variation tritt aufgrund des Grain-Effekts auf. Dieser wird jedoch genauer in Kapitel 4.3 Analyse der Post Processing-Effekte betrachtet.

Grad der Abstraktion

Im Hinblick auf die Gameplay-relevanten Objekte lässt sich an verschiedensten Stellen ein hoher Abstraktionsgrad feststellen. Das beruht simpel erneut darauf, dass die Umsetzung in dieser Form eine hohe Zeitersparnis mit sich bringt. Bei den Charakteren fällt dies durch die Darstellung der Gliedmaßen als einheitliche Schlauchformen auf. Andeutungen für Gelenke oder etwa Stofffalten der Kleidung entfallen dabei. Hände werden nur mit vier anstatt von fünf Fingern gezeichnet, auch wenn sie menschlichen Händen nachempfunden sind. Es finden sich an Schuhen und auch der Kleidung keine kleinen Details, wie etwa Schnürsenkel. Generell ist jeder Teil des Charakters nur insofern gezeichnet, wie es nötig ist, um die Silhouette, Form und Farbe des jeweils dargestellten Objekts zu definieren.

Frame Rate

Studio MDHR orientierte sich bei der Erstellung von Cuphead wie schon erwähnt an den Cartoons der 1930er Jahre. Die Animationen sind also nicht nur von Hand gezeichnet, sondern auch in 24fps animiert. Das Spiel selbst läuft jedoch trotzdem in 60fps, um ein responsives Spielerlebnis für den Spieler zu gewährleisten.[2]

Duplicates und Smearing

In Cuphead lassen sich eine Vielzahl von Anwendungsfällen von Smears und Duplicates finden. Das hilft nicht zuletzt, den nostalgischen Look der 1930er-Cartoons einzufangen. Neben den üblichen Smear- und Duplicates-Effekten, finden sich außerdem häufig kleine Linien, welche schnelle Bewegungen unterstützen.

Abbildung 4.7: Geschwindigkeits-Linien in Cuphead

4.1.2 Stilelemente der statischen Objekte

In diesem Abschnitt sollen die Stilelemente der statischen, also nicht Gameplay-relevanten Objekte, betrachtet werden. Dabei soll an gegebenen Stellen bei Stilelementen auf den speziellen Unterschied zu Gameplay-relevanten Objekten hingedeutet werden.

[2]Unity Technologies (o.J.): Cuphead von StudioMDHR

Line Art

Der Hintergrund, besonders bei Cuphead, aber auch in vielen anderen Rubber Hose Animation Cartoons, erstrahlt üblicherweise in einem gemalt aussehenden Art Style. Dabei lässt sich sehr offensichtlich erkennen, dass im Gegensatz zu Gameplay-relevanten Objekten, keine Form von Line Art festzustellen ist.

Beleuchtung

Ebenfalls einen großen Unterschied zu Gameplay-relevanten Objekten stellt die Beleuchtung der statischen Hinter- und Vordergrundelemente dar. Diese sind viel realistischer beleuchtet dargestellt. Es finden sich Helligkeitsabstufungen innerhalb eines Farbfeldes. Darüber hinaus lassen sich auch stellenweise indirekte Beleuchtung und Ambient Occlusion erkennen. Die Beleuchtung der Objekte generell ist jedoch diffuser. Im Vergleich zu den Gameplay-relevanten Objekten sind fast nie vergleichbar scharfe Highlights zu finden.

Farbe

Der Hintergrund und andere statische bzw. nicht Gameplay-relevante Objekte weisen eine sehr viel höhere Farbquantität als die für das Gameplay relevanten Objekte auf. Das liegt zum einen an der im vorherigen Kapitel beschriebenen, viel detaillierteren Beleuchtung. Zum anderen lässt sich eine Art des Verlaufs der Farben ineinander erkennen, welcher an die Charakteristiken von Wasserfarben erinnert. Besonders bei den Farbverläufen der Hintergrundelemente lässt sich darüber hinaus ein leichtes, aber grobes Bildrauschen feststellen. Dies könnte folgend ebenfalls mit einem Noise-Overlay nachgestellt werden. Eine Leinwand-Textur lässt sich aber darüber hinaus auch bei den Hintergrundelementen nicht identifizieren.

Grad an Abstraktion

Die statischen Elemente sind weitaus weniger abstrakt dargestellt. Bei diesen finden sich oftmals kleinere Oberflächenstrukturen und eine präzisere Darstellung der Beleuchtung. Des Weiteren werden auch kleinere Details zu der hauptsächlichen Form ergänzt. Das soll anhand der folgenden Abbildung verdeutlicht werden.

Abbildung 4.8: Abbildung eines Levels aus Cuphead, folgend thematisierte Objekte sind vergrößert dargestellt

In der dargestellten Ingame-Aufnahme aus Cuphead befinden sich Holzpfeiler als animierte und als statische Objekte. Die jeweils gemeinten Stellen sind daneben vergrößert dargestellt. Der Holzpfeiler, welcher sich am Schiff befindet und animiert ist, ist dabei deutlich abstrakter dargestellt. Dieser besteht nur aus einer rechteckigen Form, welche mit zylindrischen Enden abgeschlossen ist und zwei Ringen zur besseren Darstellung der Rundung. Die statischen Holzpfeiler im Wasser sind im Vergleich dazu viel realitätsnäher. Diese weisen eine viel detailgetreuere Darstellung von Licht und der Oberflächenstruktur des Holzes auf. Zudem finden sich sogar kleine Einkerbungen am oberen Ende des Pfeilers. Außerdem wird sogar das Seil, durch welches das Holz festgebunden ist, detailliert dargestellt. Um diesen Abschnitt abzuschließen ist wichtig zu erwähnen, dass auch die statischen Elemente nicht realistisch dargestellt sind und sogar einen stark gemalten Look abbilden. Die statischen Elemente verfügen jedoch über einen viel höheren Detailgrad, besitzen mehr Bildinformationen, welche über die Definition der Form hinausgehen und folgen sehr viel mehr Regeln eines realen Abbildes.

4.2 Analyse der VFX

Speziell im Fall von Cuphead sind VFX meist sehr abstrakt umgesetzt. Das Element, was in dieser Kategorie am häufigsten zu finden ist, sind kleine weiße Wolken. Diese tauchen unter anderem beim Laufen, Landen oder Dash des Spieler-Charakters, aber auch zum

Beispiel beim Besiegen kleinerer Gegner in verschiedenen Variationen auf. Darüber hinaus werden auch andere Formen, wie etwa Sterne oder Blitze bei den VFX eingebunden. Zur besseren Verdeutlichung der gemeinten Elemente ist eine kleine Sammlung von betreffenden Ausschnitten aus Cuphead in der nächsten Abbildung zu sehen.

Abbildung 4.9: Drei VFX aus Cuphead

4.3 Analyse der Post Processing-Effekte

Cuphead benutzt nur eine überschaubare Menge an Post Processing-Effekten, welche hauptsächlich darauf ausgelegt sind, den nostalgischen Look der 1930er-Cartoons nachzustellen. Der Bloom-Effekt wurde bereits in Kapitel 4.1.1 Beleuchtung näher thematisiert, soll hier jedoch erneut benannt werden, da dieser später als Post Processing-Effekt umgesetzt wird. Der auffälligste Post Processing-Effekt ist das Overlay, welches sich über den gesamten Bildschirm erstreckt. Dieses fügt Bildunreinheiten in Form von weißen Punkten, aber auch größeren Kratzern hinzu. Daran anschließend lässt sich auch der Vignette-Effekt erkennen. Durch diesen werden die Ecken des Bildes etwas abgedunkelter dargestellt als die Bildmitte. Dabei ist zu erwähnen, dass verschiedene Level in Cuphead diesen Vignette-Effekt unterschiedlich stark verwenden. Der Effekt ist in manchen Levels sehr deutlich, in manchen jedoch auch kaum zu erkennen. Beide Effekte, also Overlay und Vignette, verfolgen das Ziel, im schlussendlichen Bild technische Einschränkungen der 1930er-Cartoons zu simulieren und das übergreifende Gefühl dieses Mediums aufzugreifen. Cuphead selbst stellt jedoch keine vollständige Imitation eines 1930er-Cartoons dar, sondern bringt eher ikonische Merkmale davon in die heutige Zeit. Der Charme der alten Ästhetik wurde dabei aufgegriffen, jedoch fällt dieser beim Spielen nahezu nicht störend auf. Es lässt sich viel mehr als stilistisches Mittel, anstatt als einen technischen Defizit

erkennen. Die Effekte des Overlays und des Vignette-Effekts sind beide zur besseren Verdeutlichung in der folgenden Abbildung dargestellt.

Abbildung 4.10: Cuphead Overlay (zur besseren Sichtbarkeit invertiert) und Vignette-Effekt (nachgestellt)

4.4 Zusammenfassung der Analyseergebnisse

In diesem Abschnitt sollen die Ergebnisse aus der vorangegangenen Analyse übersichtlich und knapp in einer Tabelle dargestellt werden. Dies dient später dazu, die entstandene Umsetzung besser mit dem grundlegenden Ziel abgleichen zu können.

Tabelle 4.1: Zusammenfassung der Analyseergebnisse

Stilelement	**Ausprägung**
Generell	Differenzierung von Hinter- und Vordergrundelementen
Merkmale der animierten und Gameplay-relevanten Objekte	
Line Art	
Outline	vorhanden
Inlines	vorhanden, etwas dünner als Outlines
Cross Hatching	nicht vorhanden
Beleuchtung	Unlit Shading; ggf. mit elliptischem Schatten darunter; selten Schatten auf Gameplay-Elementen
Bloom	vorhanden
Farbe	geringe Farbquantität
Leinwand-Textur	keine
Abstraktionsgrad	hoch
Frame Rate	24fps
Duplicates und Smearing	vorhanden; bei Charakteren nur leicht, bei Gegner häufig stark ausgeprägt
Merkmale der statischen Objekte	
Line Art	nicht vorhanden
Beleuchtung	realistischer als animierte Objekte; Abstufungen von Helligkeit; teils indirekte Beleuchtung und Ambient Occlusion; diffuser als animierte Objekte
Farbe	höhere Farbquantität; Verlauf der Farben ineinander; Bild- bzw. Farbrauschen
Leinwand-Textur	keine
Abstraktionsgrad	weniger abstrakt als animierte Objekte; dennoch künstlerisch
Merkmale der VFX-Elemente	
analog zu Gameplay-relevanten Objekten meist in sehr simplen Formen zur Unterstützung des Gameplay-Geschehens	
Merkmale der Post Processing-Effekte	
starkes Bildrauschen zu erkennen variierend starker Vignette-Effekt	

4.5 Praktische Umsetzung der Stilelemente aus der Analyse

Basierend auf den Ergebnissen der vorangegangenen Art Style-Analyse von Cuphead sollen jetzt Vorgehensweisen vorgestellt werden, um diesen Art Style mit einer 3D-Pipeline umsetzen zu können. Dafür wird die in der Analyse getroffene Unterteilung in statische und animierte bzw. Gameplay-relevante Objekte weiterhin verwendet, um die Differenzierung der verschiedenen Art Styles klarer darzustellen.

4.5.1 Praktische Umsetzung eines animierten Charakters

In den folgenden Unterkapiteln werden die einzelnen Stationen zur finalen Erstellung der Animations-Frames eines Charakters im Stil von Cuphead mithilfe der bisher besprochenen 3D-Techniken beleuchtet.

Modellierung des Charakters

Für die Modellierung des Charakters selbst konnten herkömmliche Vorgehensweisen verwendet werden. Für Charakter-Modelle, welche innerhalb einer Echtzeit-Anwendung zum Einsatz kommen, ist es üblicherweise notwendig, eine überschaubare Anzahl an Polygonen zu wahren. Da der Charakter, sowie die dazugehörigen Animationen noch in Blender gerendert wurden, musste jedoch in diesem Fall kein besonderer Wert auf eine performante Gestaltung des Meshs gelegt werden. Bei der Erstellung des Modells wurde sogar gegenteilig darauf geachtet, eine erhöhte Anzahl an Edge Loops um die Gliedmaßen herum zu platzieren, da diese später gummiartig deformiert werden sollen. Des Weiteren wurde das Modell durch Zusammenstecken einzelner Unter-Meshes erstellt. Dies hat eine bessere Verteilung der Outlines, beim Verwenden der Mesh-Based Outline-Methode, zur Folge.

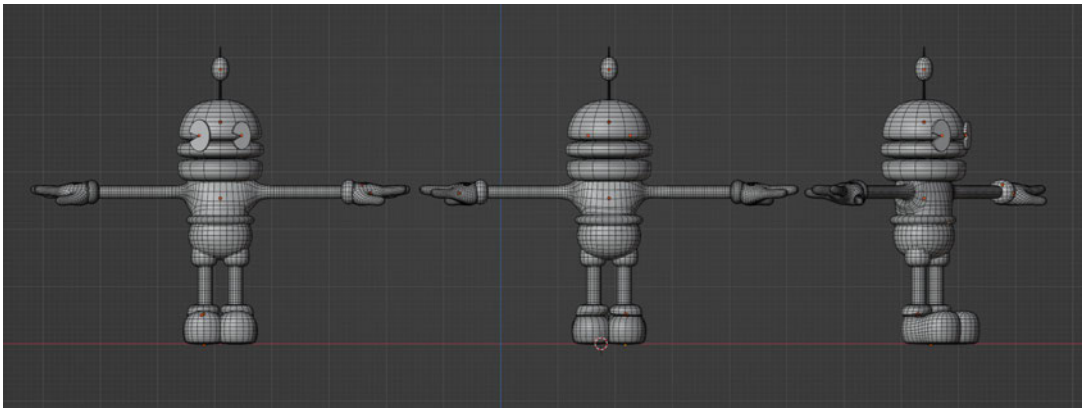


Abbildung 4.11: Topologie des Charakter-Modells

Implementation der Stilelemente des Charakters

Wie aus der Stilanalyse hervorgegangen ist, muss der Charakter nur mit einem Unlit-Shading gerendert werden. Die dafür zu verwendende Vorgehensweise beim Erstellen eines entsprechenden Shaders in Blender wurde bereits in einem früheren Kapitel erwähnt und kommt hier erneut zum Einsatz. Für das richtige Shading des Charakters genügte es, einen Shader zu erstellen, der nur einen RGB-Wert in den Surface Output entgegennimmt. Daran anschließend, mussten lediglich entsprechende Farben für die jeweiligen Teile des Modells festgelegt werden.

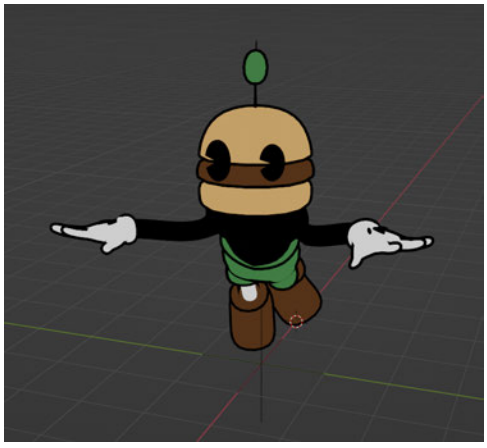


Abbildung 4.12: Finale Farbpalette von Burger Boy

Aus der Analyse ist jedoch ebenfalls hervorgegangen, dass der Charakter über spezielle Outlines und Inlines verfügen muss. Für die Umsetzung dieses Stilelements wurde die in Kapitel 3.2.1 Outlines thematisierte Methode zur Erstellung von Mesh-Based Outlines und Inlines verwendet. Etwa die Details auf den Handschuhen oder eine Inline innerhalb der Schuhe wurde somit durch die Erstellung kleiner Meshes an den gegebenen Stellen gewährleistet. Darüber hinaus wurde auch die Freestyle Render Engine zur Erstellung einer dickeren Kontur um den Charakter herum verwendet.

Rigging des Charakters

Zur Erstellung des Charakter-Rigs konnte zu einem Teil das Rigify-Addon von Blender genutzt werden. Durch dieses Addon werden alle Anpassungen, welche ein Rig auf die Verwendung mit Squash and Stretch vorbereiten, automatisch getroffen. Die Gliedmaßen, also beide Arme und Beine, wurden jedoch mit individuell erstellten Rigs versehen und dann mit dem Rigify Rig zusammengefügt. Auch diese lassen sich später über Squash und Stretch deformieren. Dabei wurde die schon angesprochene Bendy Bones-Technik eingesetzt. Der genaue Verlauf der Kurve von Bendy Bones lässt sich über eine Start- und End-Handle einstellen. Diese wurden jeweils an Knochen gebunden. Somit lässt sich innerhalb des Rigs der Gliedmaßen, auch durch die Rotation der Hand- und Fußgelenks-Controller, der Verlauf der Arme steuern. Daher kommt es auch zustande, dass sich entlang der Gliedmaßen keine weiteren Controller befinden, was bei einem typischen Rig der Fall wäre.

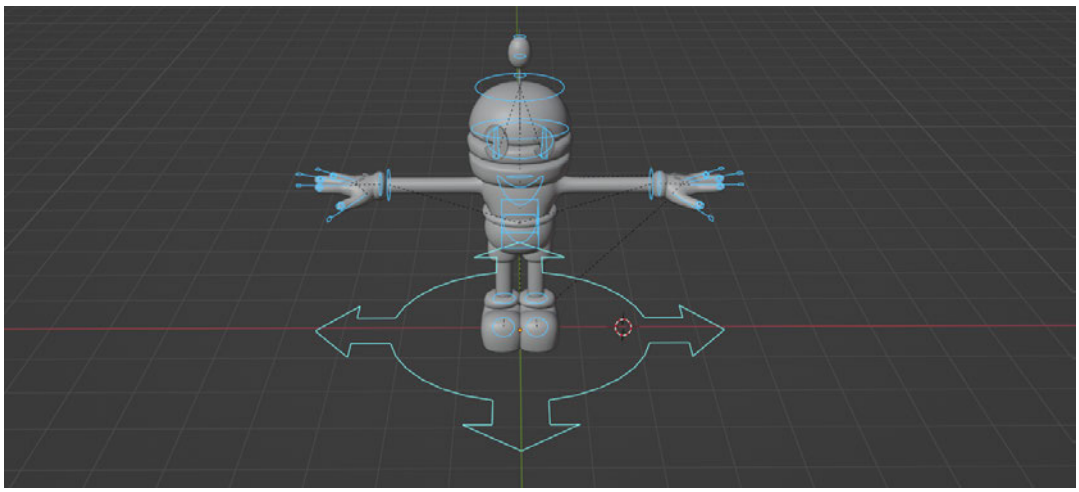


Abbildung 4.13: Rig des Charakter-Modells

Animation des Charakters

Bei der Animation des Charakters gilt es zu beachten, dass die einzelnen Animationen nahtlos wiederholbar sein müssen, da diese innerhalb eines Videospiels eingesetzt werden sollen. Das gilt insbesondere für die Lauf- und Steh-Animation. Zur genaueren Verdeutlichung wird von einer Animation mit insgesamt 24 Frames ausgegangen. Bei dieser ist es nun entscheidend, dass der 24. Frame nahtlos in den ersten übergeht. Auf diese Weise kann die Animation auch mehrfach hintereinander abgespielt werden, ohne dass ein Schnitt zu erkennen ist. Das ist deshalb von Bedeutung, da letztendlich durch den Spieler gesteuert wird, welche Animation wie lang bzw. wie oft abgespielt wird. Zur Gewährleistung davon stellt eine 3D-Software eine große Hilfe dar. Speziell im Fall von Blender kann durch einen Modifier sehr leicht eine zyklische Animation erstellt werden. Dadurch kann sich der Artist bequemer auf die eigentliche Animation fokussieren, auch ohne einzelne Posen zur nahtlosen Wiederholung im Voraus geplant zu haben. Des Weiteren macht es die Arbeit mit Keyframes natürlich möglich, im Nachhinein feine Änderungen an der Animation vorzunehmen. Besonders bei schnell wirkenden Animationen, wie etwa dem Dash, wurden auch die zuvor geschaffenen Funktionalitäten für die Umsetzung des Squash and Stretch-Prinzips verwendet. Darüber hinaus konnten herkömmliche Methoden zur Animation verwendet werden. Eine Darstellung der speziell dabei entstandenen Animations-Frames befindet sich im Kapitel 4.6 Vorstellung der Ergebnisse.

4.5.2 Praktische Umsetzung eines Gameplay-relevanten Objekts

Als Beispiel für ein Gameplay-Element soll eine Plattform umgesetzt werden, auf der der Spieler landen kann. Es wird dabei versucht eine der Plattformen, welche in der folgenden Abbildung dargestellt sind, nachzubilden.

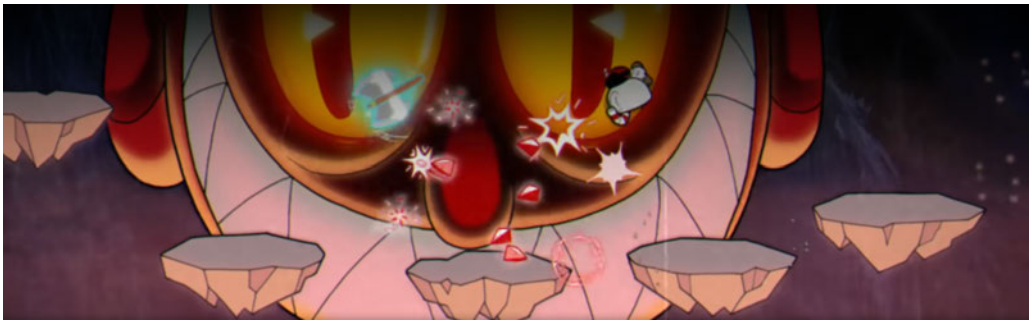


Abbildung 4.14: Abbildung der gemeinten Plattformen in Cuphead

Bei diesen Plattformen gilt es eine kleine Besonderheit zu beachten. Wie auf der Abbildung zu erkennen ist, sind die Outlines, besonders oben, dicker als die Inlines. Das grundlegende Modell dafür ist sehr schnell zu erstellen. Dafür wurde die Form der oberen Fläche modelliert, nach unten extrudiert und entsprechend der Steinstrukturen angepasst. Über einen Unlit Shader, können danach auch schon die jeweiligen Farben für die einzelnen Flächen vergeben werden.

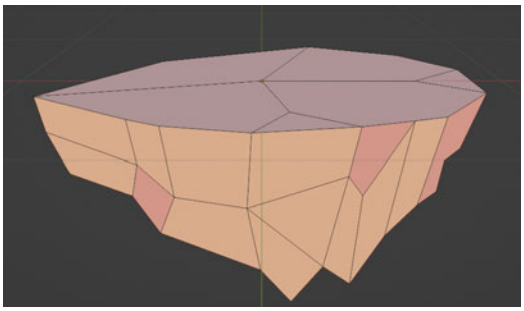


Abbildung 4.15: Modell der Steinplattform mit zugewiesenen Materialien

Für eine authentische Nachstellung des Line Arts werden zwei der bereits vorgestellten Methoden kombiniert. Die Outlines werden anhand eines Outline-Meshs, also Mesh-Based erstellt und für die Inlines wird der Freestyle Renderer von Blender verwendet. Im Freestyle Renderer lässt sich eine Einstellung treffen, dass alle "Material Bounderies", also Abgrenzungen der Farben, mit einer Inline versehen werden. Alle weiteren Linien können über "Edge Marks", also das Kennzeichnen einer Kante, welche dann als Linie gerendert wird, festgelegt werden. Mit dem Ziel, dass die Inlines ohne Komplikationen gerendert werden können, wird das Outline-Mesh separat vom Inline-Mesh gerendert und schlussendlich mittels simpler Kompositionstechniken übereinandergelegt.

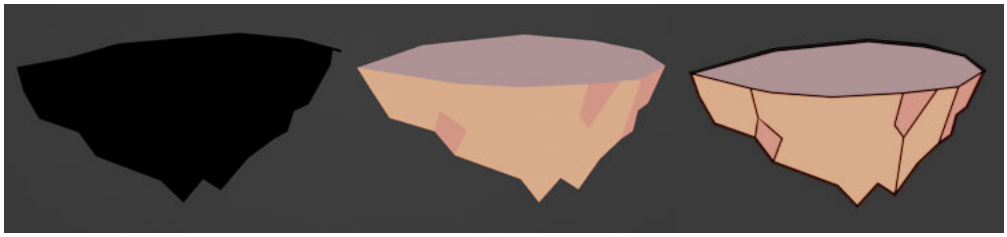


Abbildung 4.16: Renderpass 1 (links), Renderpass 2 (mittig), Ergebnis (rechts)

4.5.3 Praktische Umsetzung der statischen Objekte

In diesem Kapitel soll besonders die Erstellung der Hintergrundelemente thematisiert werden. Dabei werden neben einem Füllhintergrund weitere Ebenen in verschiedenen Tiefen in Relation zur Kamera erstellt. Vorerst kann dafür mit herkömmlichen Modellierungsmethoden eine Szene erstellt werden. In speziell diesem Fall handelt es sich um die Erstellung einer Höhlenszene.

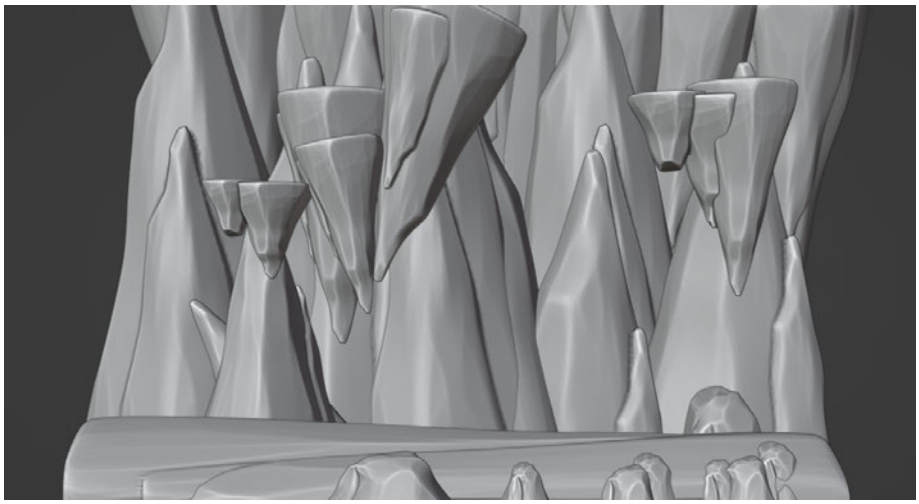


Abbildung 4.17: Hintergrundszene in 3D

Auch hier brachte die Verwendung einer 3D-Software eine sehr große Zeitersparnis, da nur eine geringe Anzahl an Grundmodellen erstellt werden musste und diese danach dupliziert werden konnten, um die gesamte Szene zu erstellen. Folgend wurde ein Shader erstellt, um das Aussehen von Wasserfarben zu simulieren. Die zugrunde liegende verwendete

Technik basiert auf einer von Diego Gangl vorgestellten Methode.[3] Eine Übersicht des dafür verwendeten Shader-Graphen befindet sich im Anhang unter Shader zur Simulation von Wasserfarbe. Nach dem Erstellen des Shaders, wurde dieser mit leicht unterschiedlichen Farbvariationen auf die Steine der Szene angewendet. Die Szene wurde anschließend entsprechend beleuchtet und es wurden Einstellung zum Rendern von Ambient Occlusion getroffen. Weitere Methoden aus der Kategorie der erweiterten Beleuchtung wurden in diesem Fall nicht verwendet. Die Szene wurde dann in einzelnen Ebenen gerendert. Das Ergebnis des Renderings wurde danach in der Software Photoshop nachbearbeitet. Dabei ist zu erwähnen, dass innerhalb von Photoshop ausschließlich Nachbearbeitungs-Effekte auf die gerenderten Bilder angewendet wurden. Dies umfasste hauptsächlich das Hinzufügen eines künstlichen Bildrauschens. Eine Auswahl der gerenderten Ebenen werden in der folgenden Abbildung isoliert gezeigt.

Abbildung 4.18: Gerenderte und nachbearbeitete Ebene des Hintergrunds isoliert

4.5.4 Praktische Umsetzung eines VFX-Elements

Wie aus Kapitel 4.2 Analyse der VFX zu entnehmen ist, erscheinen in Cuphead sehr häufig kleine Wolken, welche Bewegungen und Aktionen untermalen. Folgend soll exemplarisch einer dieser Wolkeneffekte nachgebildet werden. Dafür wird vorerst eine Kugel, spezieller eine Icosphere, als grundlegende Form erstellt und ein weißer Unlit Shader zugewiesen. Die Grundform wird nun unterteilt, um eine bessere Rundung und Arbeit mit dem Objekt zu gewährleisten. Als Nächstes wird der Displace-Modifier auf die Kugel angewendet. Durch diesen Modifier können die Vertices eines 3D-Objekts anhand einer Textur versetzt werden. In diesem Fall sollen die Vertices entlang ihrer eigenen Normale durch eine Voronoi-Textur verschoben werden. Durch diese Textur lassen sich prozedural zusätzliche Rundungen auf der Oberfläche der zugrunde liegenden Kugel erstellen.

[3] Diego Gangl (2020): Creating a watercolor material in Eevee

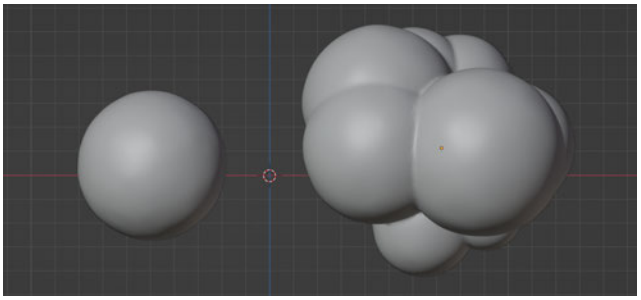


Abbildung 4.19: Normale Kugel (links), Kugel mit Voronoi-Displace (rechts)

Diese prozedurale Textur bringt einen weiteren Vorteil mit sich. Indem die Koordinaten der Kugel im World Space, dem Koordinatensystem der gesamten Szene, betrachtet werden, lässt sich die Wolke sehr leicht animieren. Um den Wolkenausstoß nun zu animieren, muss das grundlegende Wolken-Objekt einige Male dupliziert werden. Folgend werden diese, von der Mitte ausgehend, mit einer jeweiligen Geschwindigkeit nach außen bewegt, sodass diese eine kreisförmige Emission nachempfinden. Durch die Verwendung der prozeduralen Textur in World Space, animiert sich die Form der einzelnen Wolken dabei automatisch mit, da sich der verwendete Textur-Ausschnitt mit der Bewegung der Kugel ändert. Diese Wolken-Animation wird auf dem Boden der Szene platziert, um später den Effekt beim Aufkommen des Charakters nachzustellen. Die folgende Abbildung zeigt den Verlauf der gerenderten Animation in ausgewählten Frames.

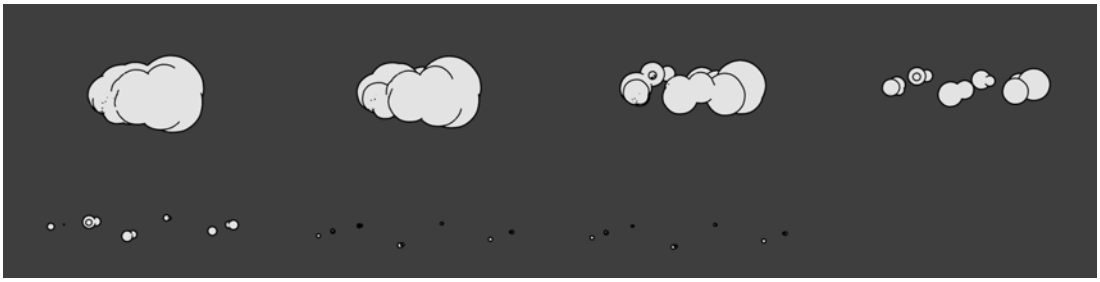


Abbildung 4.20: Animations-Frames des Wolken VFX

Ein weiterer Vorteil dieser Vorgehensweise ist, dass auch der Detailgrad der Kugel über die Voronoi-Textur gesteuert werden kann. Die nächste Abbildung zeigt die gleiche Kugel, welche über zwei verschiedene Voronoi-Texturen beeinflusst wurde.

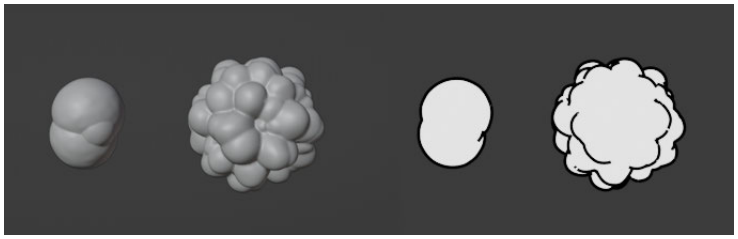


Abbildung 4.21: Kugel durch zwei unterschiedliche Voronoi-Texturen beeinflusst

4.5.5 Komposition der erstellten Assets in einer Game Engine

Zuerst wurden die erstellten Assets in die Unity Game Engine importiert. In der Unity Engine müssen entsprechende Einstellungen getroffen werden, damit die importierten Bilder als sogenannte "Sprites" interpretiert werden. Die Verwendung von Sprites, anstatt von einfachen Bilder, vereinfacht die Arbeit im Folgenden. Alle Gameplay-relevanten Elemente wurden dann auf einer bestimmten Ebene platziert, während die Hinter- und Vordergrundelemente entsprechend ihrer vorgesehenen Tiefe nach hinten oder vorn versetzt wurden. Das dient hier hauptsächlich der Übersicht und ist für das Ergebnis nicht weiter von Bedeutung.

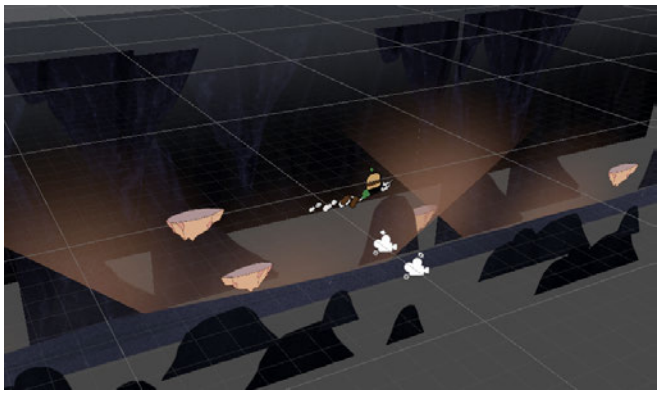


Abbildung 4.22: Aufbau der Unity Szene in Ebenen

Die dabei entstandene Szene wird von einer orthogonalen Kamera gerendert. Das bedeutet, dass der Eindruck von Perspektive nicht durch die Kamera dargestellt wird. Stattdessen wurde, wie vorher erläutert, der Parallax-Effekt zum Nachempfinden von Tiefe implementiert. Dafür wurde ein simples Script geschrieben, welches im Anhang dieser Arbeit unter Parallax-Script zu finden ist. Wie aus der Analyse hervorgegangen ist,

müssen zwei verschiedene Art Styles für die Hintergrund- und die Vordergrund-Elemente umgesetzt werden. Die Verschiedenheit der beiden Stile ist zwar schon beim Rendern der Bilder in Blender berücksichtigt worden, besonders für die Anwendung von Post Processing-Effekten sollten hier die Stile jedoch erneut differenziert werden. Um Post Processing-Effekt auf Hinter- und Vordergrund unabhängig voneinander anwenden zu können, wurde sich für ein Zwei-Kamera-Rendering entschieden. Dabei rendert eine der beiden Kameras nur die Hintergrundelemente und eine nur die Vordergrund- bzw. die Gameplay-relevanten Elemente. Die Ergebnisse beider Kameras werden dann übereinandergelegt, um das finale Bild zu erhalten. Das bringt, wie schon angesprochen, schlussendlich den Vorteil mit sich, dass auf Hintergrundelemente Post Processing-Effekte angewendet werden können, von denen die Gameplay-relevanten Objekte unbetroffen bleiben. Die Charakter-Animationen, sowie die entstandenen VFX-Animationen wurden importiert und als Einzelbild-Animationen angelegt. Bei diesen galt es zu berücksichtigen, dass auch in Unity die richtige Frame Rate von 24fps eingestellt ist. Da es sich im Rahmen dieser Arbeit hauptsächlich um die Evaluation der visuellen Darstellungen handeln soll, wurde auf die weiterführende Implementation von Spielfunktionen verzichtet.

4.6 Vorstellung der Ergebnisse

In diesem Kapitel sollen vorerst nur die erstellten Grafiken präsentiert und kurz beschrieben werden. Eine genaue Evaluation sowie ein Abgleich mit den Ergebnissen der vorangegangenen Analyse findet im nächsten Kapitel 4.7 Evaluation der Ergebnisse statt.

Ergebnisse des animierten Charakters

Zur besser erkennbaren Darstellung ist folgend nur jeweils ein Frame jeder Animation abgebildet. Die vollständige Anzahl aller erstellten Animations-Frames befindet sich im Anhang dieser Arbeit unter dem Abschnitt Charakter Animationen.

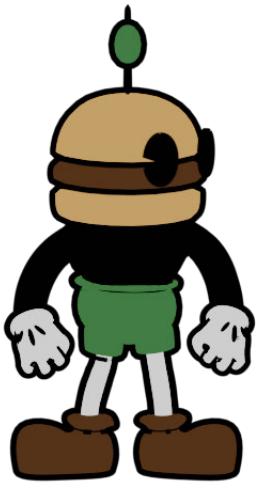


Abbildung 4.23: Ein Frame aus der Idle-Animation des Charakters

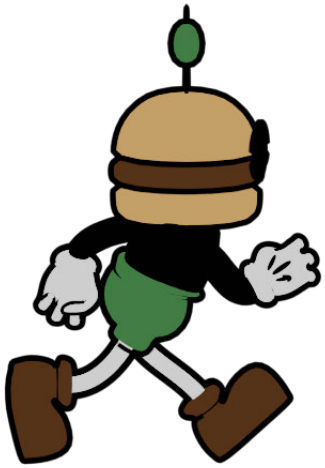


Abbildung 4.24: Ein Frame aus der Run-Animation des Charakters

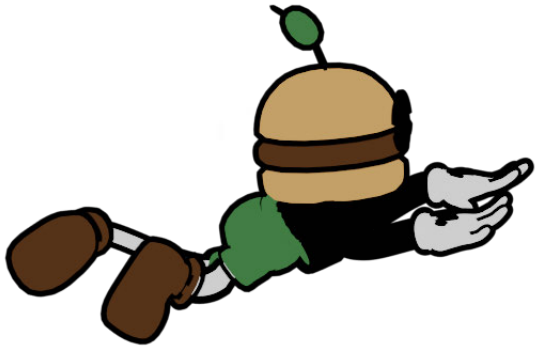


Abbildung 4.25: Ein Frame aus der Dash-Animation des Charakters

Ergebnisse der VFX

Die folgende Abbildung zeigt vier ausgewählte Frames aus der erstellten VFX-Animation. Die vollständigen Animations-Frames dafür lassen sich im Anhang unter VFX-Animation betrachten.

Abbildung 4.26: Vier ausgewählte Frames aus der VFX-Animation

Ergebnisse der Gameplay-relevanten Elemente

Das folgende Ergebnis zeigt eine Plattform, mit der der Spieler interagieren kann. Folglich ist diese auch im Art Style der animierten bzw. in diesem Fall der Gameplay-relevanten Objekte gestaltet.

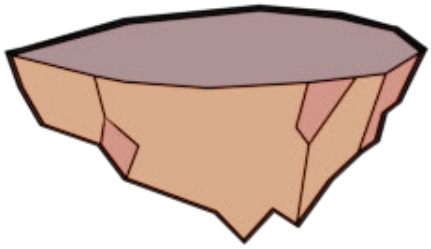


Abbildung 4.27: Gameplay-relevante Plattform

Ergebnisse der statischen Objekte

Zuerst soll hier eine Ebene der Hintergrundelemente isoliert und ohne die Anwendung von Post Processing dargestellt werden. Die folgende Grafik ist also eine Bilddatei, wie sie in Blender gerendert wurde nach der Bearbeitung mit Photoshop.

Abbildung 4.28: Isolierte statische Ebene vor dem Import in die Unity Engine

Zur Darstellung der finalen visuellen Erscheinung der statischen Elemente zeigt die folgende Abbildung nun die kombinierten Einzelebenen nach der Anwendung von Post Processing-Effekten in der Unity Engine.

Abbildung 4.29: Finale Zusammensetzung der statischen Elemente mit Post Processing-Effekten

Ergebnisse der Komposition in der Game Engine

Zum Abschluss der Ergebnisdarstellung zeigt die letzte Abbildung einen Ausschnitt aus der Unity Engine, in dem alle erstellten Assets zusammengesetzt zu sehen sind.

Abbildung 4.30: Finale Komposition aller erstellten Assets in der Unity Engine

4.7 Evaluation der Ergebnisse

Die Evaluation der finalen Ergebnisse bezieht sich neben der generellen Analyse aus Kapitel 4.1 Analyse der spezifischen Stilelemente eines Rubber Hose Animation Styles für eine besser Übersicht vorrangig auf die Zusammenfassung der Analyseergebnisse in tabellarischer Form (4.1 Zusammenfassung der Analyseergebnisse). Um weiterhin eine bessere Nachvollziehbarkeit zu gewährleisten, sind die folgenden Unterkapitel analog der Analysepunkte aus der Tabelle untergliedert.

Evaluation des animierten Charakters

Die folgenden Unterkapitel gehen, wie vorher beschrieben, den einzelnen Analysepunkten entsprechend bewertend auf die Ergebnisse der Charakter-Animationen ein. Das jeweils thematisierte Asset wird zum Anfang jedes Unterkapitels erneut in kleiner Form abgebildet.

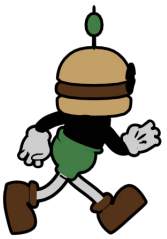


Abbildung 4.31: Ein Frame aus der Run-Animation des Charakters

Line Art

Line Art stellt dabei einen der herausforderndsten Teile dar. Grundlegend konnte das Stilelement Line Art bei der Erstellung des Charakters umgesetzt werden. Die Vorgabe, dass die Kontur des Charakters in einem dickeren Linienstil nachgezeichnet sein muss, wurde ebenfalls erfüllt. Es müssen jedoch kleinere Abstriche bei der Konsistenz der Linien gemacht werden. Als ein Nachteil der Mesh-basierten Outlines, wurde in einem früheren Kapitel bereits genannt, dass auf die Überlappung der Meshes geachtet werden muss. Das konnte innerhalb der Animation nicht in jedem Fall beachtet werden, weshalb es in seltenen Fällen zu ungleichen Strichstärken kommt, wo dies nicht der Fall sein sollte. Des Weiteren wurden außerdem zum Teil Linien an ungewollten Stellen generiert. Im Hinblick auf die Kritikpunkte sind jedoch zwei Sachen zu erwähnen. Zum einen ist die Umsetzung nicht sehr ausgereift, da die verwendete Methode nur über einen Zeitraum von wenigen Wochen erstellt und getestet wurde. Es ist durchaus vorstellbar, weitaus bessere Lösungen zum Rendern von Line Art zu finden. Des Weiteren kommt ein wichtiger Punkt zum Tragen, weshalb diese Methode besonders für die Anwendung innerhalb von Spielen vorgestellt wurde. Innerhalb der Spielumgebung nimmt der Charakter selbst natürlich nur einen Bruchteil des Bildschirms ein und wird folglich relativ klein dargestellt. Dies wirkt sich auf das Ergebnis positiv aus, da kleinere Bildfehler dadurch kaum bis gar nicht ins Gewicht fallen.

Beleuchtung

Die Beleuchtung stellte für den gewählten Stil, in diesem Fall dem Stil der Gameplay-Elemente, nur eine sehr geringe Herausforderung dar. Da besonders die animierten Charaktere nahezu komplett unbeleuchtet dargestellt wurden, konnte durch einen einfachen

Unlit Shader eine triviale Lösung gefunden werden, um die jeweiligen Farbflächen auch in einer 3D-Umgebung akkurat umzusetzen.

Farbe

Ebenso wie die Beleuchtung waren die damit verbundenen Farben keine riesige Hürde. Große Teile des Charakters konnten sogar in Schwarz abgebildet werden. Für die Anpassung des Bildkontrastes wurden durch Blender genug Möglichkeiten geboten.

Abstraktionsgrad

Der Abstraktionsgrad stellte jedoch erneut eine größere Schwierigkeit in der Umsetzung dar. Besonders an den Händen ist zu erkennen, dass diese einen recht hohen Detailgrad besitzen, was in der Referenz nicht in einem so hohen Ausmaß zu finden ist. Hier fällt erneut das, im Hinblick auf die Line Art-Umsetzung angesprochene, Problem auf. Die Rendering-Methoden setzen dabei Linien an Stellen, wo es nicht erwünscht ist bzw. an Stellen, wo es bei der Animation in 2D nicht getan wird. Das liegt schlussendlich daran, dass besonders im Fall der Hände ein sehr detailreiches Modell in einem sehr kleinen Bildausschnitt gerendert wird. Ein möglicher Lösungsansatz dafür wäre es, die Hände vom restlichen Körper unabhängig zu rendern, damit schon genauere Einstellungen im Hinblick auf den Abstraktionsgrad getroffen werden können. Im Folgenden würden dann die vorher gerenderten Hände nur als 2D-Bilder an das Charakter-Modell gebunden werden. Das wäre problemlos umzusetzen, da die Hände in der Animation konstant in Richtung der Kamera ausgerichtet sind, würde allerdings den Erstellungsprozess komplizierter machen.

Frame Rate

Die Erstellung der Animation in einer Frame Rate von 24fps stellte keinerlei Probleme dar, da diese Einstellung simpel im Renderer von Blender gesetzt werden kann. Die Simulation einer Animation On Twos musste in diesen Fall nicht durchgeführt werden.

Duplicates und Smearing

Wie aus der Analyse zu entnehmen, muss bei der Animation der Charaktere das Stilelement der Duplicates oder des Smearings nicht eingebunden werden. Einzelne Partikel-Effekte, wie etwa kleine Linien, welche die Schnelligkeit des Charakters verdeutlichen, könnten jedoch auch problemlos in der Unity Engine hinzugefügt werden. Bei der Animation des Charakters wurde stattdessen vorrangig das Prinzip Squash and Stretch angewendet, um die Bewegungen zu verdeutlichen. Durch das erstellte Rigify-Rig bzw. die Verwendung von Bendy Bones verlief dies problemlos.

Evaluation der VFX

Die VFX-Elemente lassen sich sehr nah im Vergleich zu den animierten Elementen betrachten. Jedoch lassen sich bei der Evaluation der VFX-Elemente manche Fehlerquellen nur weniger stark erkennen.

Abbildung 4.32: Vier ausgewählte Frames aus der VFX-Animation

Line Art

Da bei den VFX-Elementen nur eine Strichstärke gerendert werden musste und durch die Verwendung der Freestyle Rendering Engine, lassen sich hier keine inkonsistenten Linien finden. Es lassen sich dennoch kleinere Rendering Artefakte feststellen.

Abstraktionsgrad

Der Abstraktionsgrad trifft den der Referenz auch nicht vollkommen. Beim Rendern wurden erneut zusätzliche Linien erstellt, welche beim genauen Nachempfinden der Referenz nicht zu finden wären. Auch hier lässt sich allerdings sagen, dass sich dies mit Sicherheit bei einer weiterführenden Implementation der Pipeline beheben ließe. Die Erstellung der

VFX-Elemente mithilfe der 3D-Pipeline sieht nach Evaluation der Ergebnisse im Vergleich zu den vorher genannten animierten Objekte also erfolgreicher aus.

Abschließend muss für dieses Kapitel leider festgestellt werden, dass hier mit einigen Bildfehlern keine sehr gute Imitation des zu erzielenden Stils erstellt werden konnte.

Evaluation der Gameplay-relevanten Elemente

In diesem Unterkapitel sollten aufgrund der Analyse erneut die Stilelemente Beleuchtung und Farbe evaluiert werden. Da diese hier analog zu den vorher beschriebenen Grafiken sind, werden diese weggelassen.

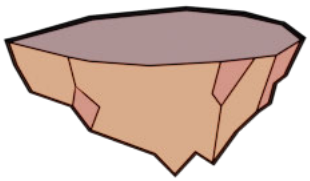


Abbildung 4.33: Gameplay-relevante Plattform

Line Art

Im Fall der Gameplay-relevanten Objekte, lassen sich die Probleme, welche bei der Animation des Charakters aufgetreten sind, gut beheben. Da die hier umgesetzten Objekte nicht animiert sind, können Fehler bei den Outlines und Inlines gut behoben werden. Dadurch können hier alle gewollten Aspekte des Stilelements erreicht werden und es entsteht im Hinblick auf Line Art der Gameplay-relevanten Objekte ein sehr zufriedenstellendes Ergebnis.

Abstraktionsgrad

Wie schon im Unterpunkt Line Art angesprochen, profitiert ebenfalls der dargestellte Abstraktionsgrad von der Möglichkeit der detaillierten Nachbearbeitung des Assets. So-

mit konnte auch beim Stilelement Abstraktionsgrad hier ein sehr gutes Ergebnis erzeugt werden.

Evaluation der statischen Elemente

Bei der Umsetzung der statischen Elemente kommt es wieder zu einigen Unterschieden in Bezug auf Beleuchtung und Farbe. Deshalb werden diese Unterpunkte bei der Evaluation dieses Abschnitts erneut beachtet.

Abbildung 4.34: Finale Zusammensetzung der statischen Elemente mit Post Processing-Effekten

Beleuchtung

Wie auch aus der Analyse hervorgeht, ist die Beleuchtung der statischen Elemente viel komplexer gestaltet. Gerade dafür war der Einsatz einer 3D-Pipeline sehr hilfreich. Die gesamte Szene konnte nach der Erstellung einerseits durch eine globale Grundbeleuchtung als auch durch einzelne Lichtquellen detailliert ausgestaltet werden.

Farbe

Die vorher benannte Beleuchtung stellt die Grundlage für die Farbe der statischen Elemente dar. Auch das Stilelement Farbe musste viel komplexer dargestellt werden, als bei den vorher thematisierten Gameplay-Elementen. Besonders im Bereich der Farbgebung konnte der Einsatz eines Shaders zur Simulation eines Wasserfarben-Looks eine große Einsparung des Arbeitsaufwands erzielen. Dabei ist zu erwähnen, dass nach der Fertigstellung des Shaders und der Grundbeleuchtung, nun enorm schnell eine zweite

Variation der Hintergrundszene durch das einfache neue Anordnen der Modelle erstellt werden kann. Schlussendlich ist noch zu erwähnen, dass Anpassungen am Farbrauschen in Photoshop gemacht wurden. Das fällt zum einen negativ ins Gewicht, da ein zusätzlicher Arbeitsschritt benötigt wurde, jedoch stellt genau diese Option auch einen sehr positiven Aspekt der vorgestellten Arbeitsweise dar. Da nach dem initialen Rendern der Szene Bildinformationen vorliegen, können diese auch vor dem Import in die Unity Engine durch verschiedenste Software nachbearbeitet und angepasst werden.

Abstraktionsgrad

Der Abstraktionsgrad musste bei den statischen Objekten geringer ausfallen, als bei den animierten. Auch das war sehr gut durch die Verwendung von 3D-Modellen, sowie einer 3D-Beleuchtung umzusetzen. Durch diese grundlegenden 3D-Methoden konnte etwas mehr Detail in die Szene gebracht werden, während durch den gezielten Einsatz von Shadern dennoch die gemalte Ästhetik gewahrt wurde.

Evaluation der Komposition in der Game Engine

Bei der Komposition der Assets lassen sich logischerweise keine Abstriche machen, da durch die vorgestellte Pipeline schlussendlich nur mit 2D-Grafiken in der Unity Engine gearbeitet wird.

Abbildung 4.35: Finale Komposition aller erstellten Assets in der Unity Engine

Das bedeutet, dass die hier verwendete Arbeitsweise in der Game Engine identisch zu Cuphead ist. Alle dabei entstandenen Abweichungen von der Referenz lassen sich deshalb aufgrund von mangelnder Zeit und Erfahrung begründen.

4.8 Möglichkeiten und Einschränkungen der verwendeten Pipeline

Wie in den vorhergegangenen Kapiteln schon thematisiert wurde, kam eine 3D-Pipeline bei der Erstellung der Bilder zum Einsatz. Es wäre ebenfalls denkbar 3D-Daten direkt in die Unity Engine zu importieren und Vorkehrung zur Imitation des gewünschten Stils zur Darstellung in Echtzeit zu treffen. Abschließend sollen deswegen kurz in Form einer Tabelle die dabei entstandenen Möglichkeiten, aber auch Einschränkungen durch das vorherige Rendern der Grafiken in Bilder gegenübergestellt werden.

Tabelle 4.2: Möglichkeiten und Einschränkungen einer 3D-Pipeline beim vorherigen Rendern eines 2D Art Styles

Möglichkeiten	**Einschränkungen**
Volle Ausschöpfung der Rig- und Animations-Werkzeuge von Blender (Kompatibilität mit Unity Engine muss nicht beachtet werden)	Keine Möglichkeit zur Adaption des Rigs in Echtzeit (z.B. Zeigen der Hand auf bestimmtes Objekt)
Keine Einschränkungen aufgrund von Performanz im Hinblick auf Polygonanzahl oder aufwendige Shader	keine oder nur eingeschränkte Physik-Simulation zur Echtzeit
Gerenderte Assets lassen sich in anderer Software weiterführend nachbearbeiten	Austausch bestimmter Körperteile oder Anbringung von Gegenständen ist schwieriger

5 Fazit

In dieser Arbeit sollten heranführend die Grundlagen zur Analyse eines Art Styles und zur Herausstellung einzelner Stilelemente geschaffen werden. Am Beispiel des Cartoon-Looks wurden diese Stilelemente identifiziert und Methoden zur Nachbildung vorgestellt. Diese aufgezeigte Vorgehensweise soll dabei einen möglichen Grundstein für die Analyse und Umsetzung weiterer Art Styles bieten.
Das hauptsächliche Thema stellte die Demonstration und Evaluation einer 3D-Pipeline dar, mit der die Ästhetik eines 2D-Cartoons nachgestellt werden kann. Die Ergebnisse dieser Forschung fielen dabei sehr positiv aus. Bei der Mehrheit von Untersuchungen konnte der gewünschte Art Style sehr genau nachgestellt werden. Dabei brachten besonders die Erstellung statischer Elemente für Hintergrund, aber auch für Gameplay-Elemente und darunter wiederum der VFX-Elemente sehr gute Ergebnisse hervor. Bei der Animation des Charakters konnten noch einige Fehler festgestellt werden, sodass diese Resultate hier nicht komplett für eine authentische Nachstellung des Stils ausreichen.
Generell betrachtet, bietet sich jedoch mit der Vielzahl an Möglichkeiten des 3D-Raums ein tolles Mittel zur Erstellung visueller Inhalte. Die Untersuchen innerhalb dieser Arbeit haben dabei besonders hervorgebracht, dass 3D für die Generierung statischer, aber auch simulierte Elemente sehr gut einsetzbar ist. Dabei wächst die Herausforderung mit dem Maß an komplexen Details, welche besonders zur Nachstellung humanoider Charaktere dargestellt werden müssen.
Überblickend sind die Ergebnisse der Evaluation dieser Methode jedoch wie schon erwähnt äußerst zufriedenstellend. Ein möglicher Lösungsansatz für die ungenügenden Ergebnisse bei der Charakter-Animation wäre die Verwendung der vorgestellten Methoden in einem hybriden System. So sollte die vorgestellte Pipeline weniger als komplette Ersatzlösung für die Erstellung eines 2D-Videospiels dienen, sondern kann sehr attraktive Optionen zum Ersparen von Arbeitsaufwand bei einer 2D-Produktion bieten.

Literaturverzeichnis

Literaturquellen

[1] Ahearn, Luke (2018): 3D game environments: Create professional 3D game worlds. London (Routledge)

[2] Beil, Benjamin et al. (2018): Game Studies. Mainz (Springer VS)

[3] Bendazzi, Giannalberto (2017): Animation. A World History. New York (Routledge).

[4] Bühler, Peter et al. (2018): Animation. Grundlagen - 2D-Animation - 3D-Animation. Berlin, Heidelberg (Springer-Verlag).

[5] Dutre, Philip et al. (2006): Advanced Global Illumination. Natick, MA (A K Peters)

[6] Elber, Gershon (1999): Interactive Line Art Rendering of Freeform Surfaces (https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00322, verfügbar am 15.08.2022)

[7] Fan, Wenshan (2021): A fast and realistic bloom rendering method for large scale 3D scene (https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9440634, verfügbar am 30.08.2022)

[8] Gauld, Dylan (2016): Growth: Visualisation of predictive mathematical models using 3D computer graphics and animation (https://discovery.dundee.ac.uk/ws/files/38591084/Final_Published_Version.pdf, verfügbar am 03.09.2022)

[9] Gooch, Bruce; Gooch, Amy (2001): Non-Photorealistic Rendering. Boca Raton, Florida (CRC Press).

[10] Gran, Carlos And´ujar (2020): Non-Photorealistic Rendering: Cross Hatching (https://upcommons.upc.edu/bitstream/handle/2117/339734/152575.pdf, verfügbar am 16.08.2022)

[11] Grau, Oliver (2004): Virtual Art. From Illusion to Immersion. Cambridge (MIT Press).

[12] Holiday, Christopher; Pallant, Chris (2021): The depth deception: Landscape, technology and the manipulation of Disney's multiplane camera in Snow White and the Seven Dwarfs (1937)

[13] Jimenez, Jorge et al. (2015): Separable subsurface scattering. journal of the European Association for Computer Graphics [S. 188-197] (https://onlinelibrary.wiley.com/doi/10.1111/cgf.12529, verfügbar am 03.09.2022)

[14] Kletschke, Irene (2011): Klangbilder. Walt Disneys Fantasia (1940). Stuttgart (Franz Steiner Verlag).

[15] Kontkanen, Janne; Laine, Samuli (2005): Ambient occlusion fields (https://artis.inrialpes.fr/Membres/Olivier.Hoel/ssao/aofields.pdf, verfügbar am 03.09.2022)

[16] Linke, Angelika (2009): Rhetorik und Stilistik / Rhetoric and Stylistics (https://www.zora.uzh.ch/id/eprint/26864/1/9783110213713.1.1.1131.pdf, verfügbar am 22.08.2022)

[17] Luque, Raul Reyes (2012): The Cel shading Technique (https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.691.5831&rep=rep1&type=pdf, verfügbar am 30.08.2022)

[18] McGowan, David (2019): Cuphead: Animation, the public domain, and home video remediation. Journal of Popular Culture (https://doi.org/10.1111/jpcu.12751, verfügbar am 05.08.202)

[19] Roberts, Steve (2004): Character Animation in 3D: Use traditional drawing techniques to produce stunning CGI animation. Oxford, England (Focal Press)

[20] Rossl, Christian; Kobbelt, Leif (2002): Line-art rendering of 3D-models (https://www.eecs.umich.edu/courses/eecs498-2/papers/rossl.pdf, verfügbar am 15.08.2022)

[21] Silber, Daniel (2015): Pixel Art for Game Developers. Boca Raton, Florida (CRC Press).

[22] Spencer, Greg et al.(1995): Physically-based glare effects for digital images. New York (ACM Press)

[23] Ted Tschang, Feichin; Goldstein, Andrea (2004): Production and Political Economy in the Animation Industry: Why Insourcing and Outsourcing Occur (https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=3852&context=lkcsb_research, verfügbar am 05.08.2022)

[24] Ulrich, Karl (2021): Wie kommt die Farbe ins Kunstwerk?: Teil 1: Vom Beginn der europäischen Malerei in der Steinzeit bis zur Renaissance. Chemie in unserer Zeit, 56 (https://onlinelibrary.wiley.com/doi/pdf/10.1002/ciuz.202000087, verfügbar am 30.08.2022)

Internetquellen

[25] Autodesk (o.J.): Indirect (global) vs. direct illumination (https://knowledge.autodesk.com/support/maya-lt/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/MayaLT/files/BoL-Indirect-global-vs-direct-illumination-htm.html, verfügbar am 03.09.2022)

[26] Blender Foundation (o.J.): Blender 2.93: Grease Pencil (https://wiki.blender.org/wiki/Reference/Release_Notes/2.93/Grease_Pencil, verfügbar am 31.08.2022)

[27] Blender Foundation (o.J.): Freestyle: Einführung (https://docs.blender.org/manual/de/dev/render/freestyle/introduction.html, verfügbar am 31.08.2022)

[28] Blender Foundation (o.J.): The Software (https://www.blender.org/about/, verfügbar am 22.08.2022)

[29] Dedene, Kristof (2020): Tutorial: Procedural hatching and manga shaders for EEVEE Blender (https://www.youtube.com/watch?v=2ZR5XIjBmho&t=480s&ab_channel=KristofDedene, verfügbar am 14.08.2022)

[30] Diego Gangl (2020): Creating a watercolor material in Eevee (https://www.youtube.com/watch?v=ZKObDQEPR3A&t=328s&ab_channel=DiegoGangl, verfügbar am 13.09.2022)

[31] Disney+ (o. J.): Steamboat Willie (https://www.disneyplus.com/de-de/movies/steamboat-willie/1Lh1k4ammOG5, verfügbar am 05.08.2022)

[32] Sony Pictures Imageworks (o.J.): Spider-man™: Into the spider-verse (https://www.imageworks.com/our-craft/feature-animation/movies/spider-man-spider-verse, verfügbar am 30.08.2022)

[33] The Game Awards (2017): The Winners 2017 (https://thegameawards.com/rewind/year-2017, verfügbar am 05.08.2022)

[34] Unity Technologies (o.J.): Cuphead von StudioMDHR (https://unity.com/de/madewith/cuphead, verfügbar am 08.09.2022)

[35] Unity Technologies (o.J.): Unity (https://unity.com, verfügbar am 08.09.2022)

[36] Vasseur, Thomas (2018): Art Design Deep Dive: Using a 3D pipeline for 2D animation in Dead Cells (https://www.gamedeveloper.com/production/art-design-deep-dive-using-a-3d-pipeline-for-2d-animation-in-i-dead-cells-i-, verfügbar am 05.08.2022)

Bildquellen

[37] 1.1 Grafik im Pixel Art Style:
https://opensea.io/assets/matic/0x2953399124f0cbb46d2cbacd8a89cf0599974963/106825514307926736650122659039617408678747571874305663381446828667610700185601, verfügbar am 21.09.2022

[38] 1.2 Abbildung der Etappen der Pipeline von Dead Cells & 1.3 Aus 3D-Daten generierte Normal Map: https://www.gamedeveloper.com/production/art-design-deep-dive-using-a-3d-pipeline-for-2d-animation-in-i-dead-cells-i-, verfügbar am 21.09.2022

[39] 2.1 Die vier apokalyptischen Reiter von Albrecht Dürer, 1511:
https://www.nationalgeographic.de/geschichte-und-kultur/2022/03/wer-waren-die-reiter-der-apokalypse, verfügbar am 21.09.2022

[40] 2.2 Darstellung der benannten Line Art-Typen in Rot gekennzeichnet: Outlines (links), Inlines (rechts): https://www.kindpng.com/imgv/hTbRRJ_thumb-up-mickeys-hand-mickey-mouse-thumbs-up/, verfügbar am 21.09.2022

[41] 2.3 Stufenweiser Aufbau von Cross Hatching-Linien:
https://thepostmansknock.com/the-beginners-guide-to-crosshatching/, verfügbar am 21.09.2022

[42] 2.4 Darstellung zu direkter und indirekter Beleuchtung:
https://www.scratchapixel.com/lessons/3d-basic-rendering/global-illumination-path-tracing, verfügbar am 21.09.2022

[43] 2.5 Ausschnitt aus dem Animationsfilm "Chihiros Reise ins Zauberland":
https://www.kinopolis.de/ab/filmdetail/chihiros-reise-ins-zauberland/21372000012PLXMQDD, verfügbar am 21.09.2022

[44] 2.7 Kunstwerk im Vector Art Style:
https://www.pinterest.de/pin/8022105578505860/, verfügbar am 21.09.2022

[45] 2.12 Visualisierung Farbton (Hue), Farbsättigung (Saturation), Farbwert (Value):
https://www.virtualartacademy.com/three-components-of-color/, verfügbar am 21.09.2022

[46] 2.13 Ausschnitt aus dem Cartoon Steamboat Willie:
https://filmschoolrejects.com/mickey-mouse-steamboat-willie/, verfügbar am 21.09.2022

[47] 2.14 Malerei von Altamira, ca. 13.500 v.Chr.:
https://www.deutsches-museum.de/museumsinsel/ausstellung/altamira, verfügbar am 21.09.2022

[48] 2.15 Paper Mario: https://xboxplay.games/de/news/paper-mario-the-origami-king-wie-man-das-shuriken-toss-minispiel-spielt-9471, verfügbar am 21.09.2022

[49] 2.17 Smear-Effekt bei Hilda Berg aus Cuphead:
https://twitter.com/studiomdhr/status/728706574928121856, verfügbar am 21.09.2022

[50] 2.20 Multiplane Camera: https://ftp.afilm.com/blog/PAT_Multiplane2_01.jpg, verfügbar am 21.09.2022

[51] Studio MDHR (2017): Cuphead
Folgende Abbildungen wurden aus dem Computerspiel Cuphead entnommen:
4.1 Verschiedene Art Styles für Hinter- und Vordergrundelemente in Cuphead
4.2 Der Spieler-Charakter und ein Gegner aus Cuphead
4.3 Elliptischer Schatten unter dem Charakter in Cuphead
4.4 Highlights (markiert durch weiße Pfeile)
4.5 Bloom-Effekte in Cuphead
4.6 Ausschnitt aus Cuphead
4.7 Geschwindigkeits-Linien in Cuphead
4.8 Abbildung eines Levels aus Cuphead, folgend thematisierte Objekte sind vergrößert dargestellt
4.9 Drei VFX aus Cuphead
4.14 Abbildung der gemeinten Plattformen in Cuphead

Anhang

Referenzübersicht

Art Style

Abbildung .1: Referenzen in Bezug auf den Art Style

Abbildung .2: Referenzen in Bezug auf den Art Style

Animation Style

Abbildung .3: Referenzen in Bezug auf den Animation Style

Rubber Hose

Abbildung .4: Referenzen in Bezug auf den Rubber Hose Style

VFX

Abbildung .5: Referenzen in Bezug auf die VFX

Cross Hatching Shader

Untergruppe für Linien-Textur

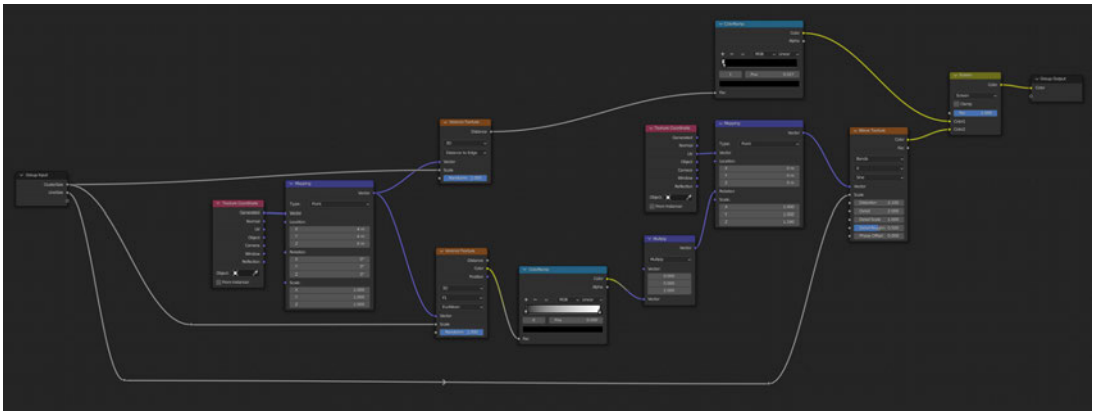


Abbildung .6: Untergruppe für Linien-Textur

Untergruppe für kleine zusätzliche Linien

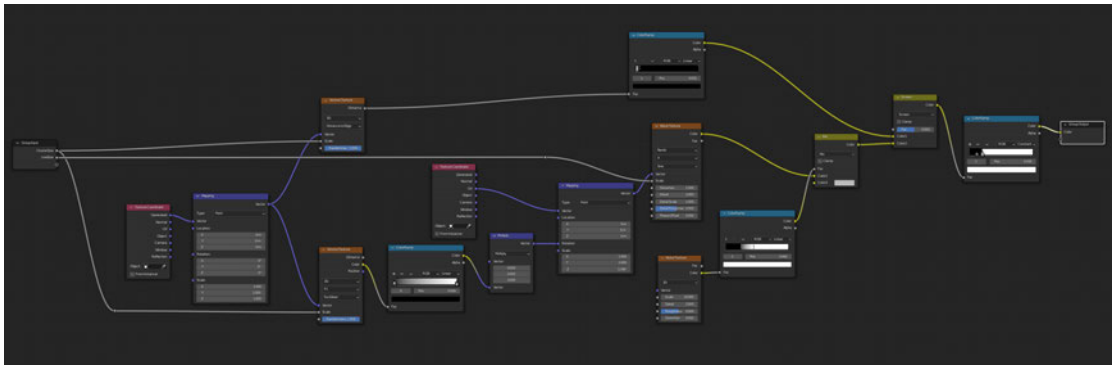


Abbildung .7: Untergruppe für kleine zusätzliche Linien

Untergruppe für Interaktion mit Licht

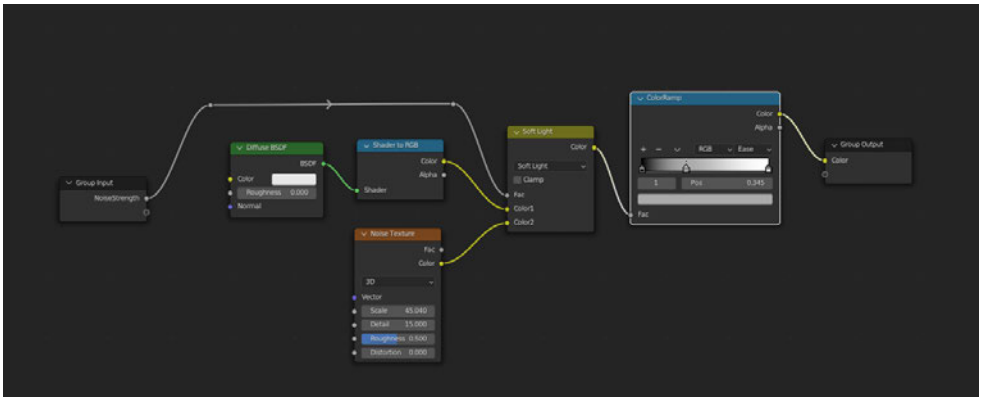


Abbildung .8: Untergruppe für Interaktion mit Licht

Finaler Shader (alle Untergruppen zusammengefügt)

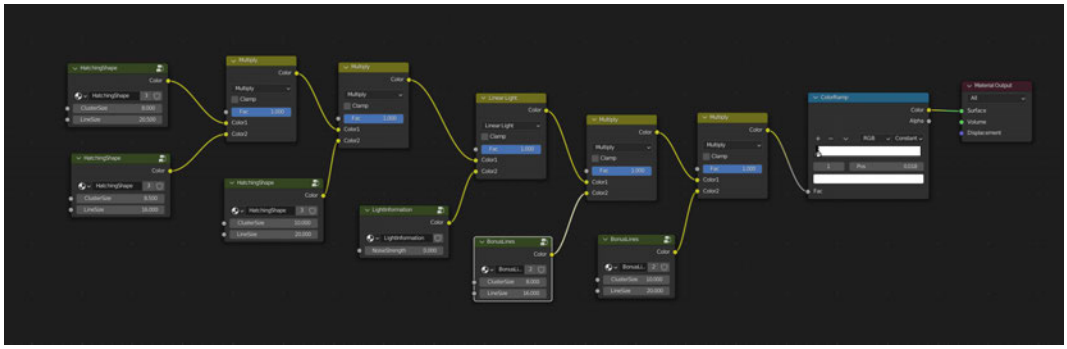


Abbildung .9: Finaler Shader (alle Untergruppen zusammengefügt)

Shader zur Simulation von Wasserfarbe

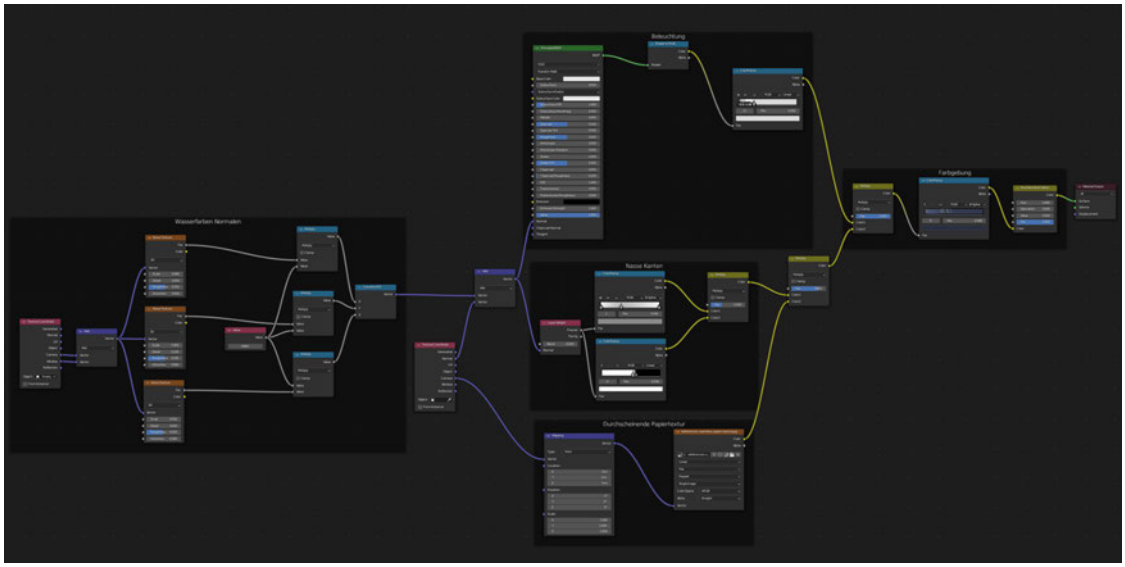


Abbildung .10: Shader zur Simulation von Wasserfarbe

Charakter Animationen

Idle-Animation

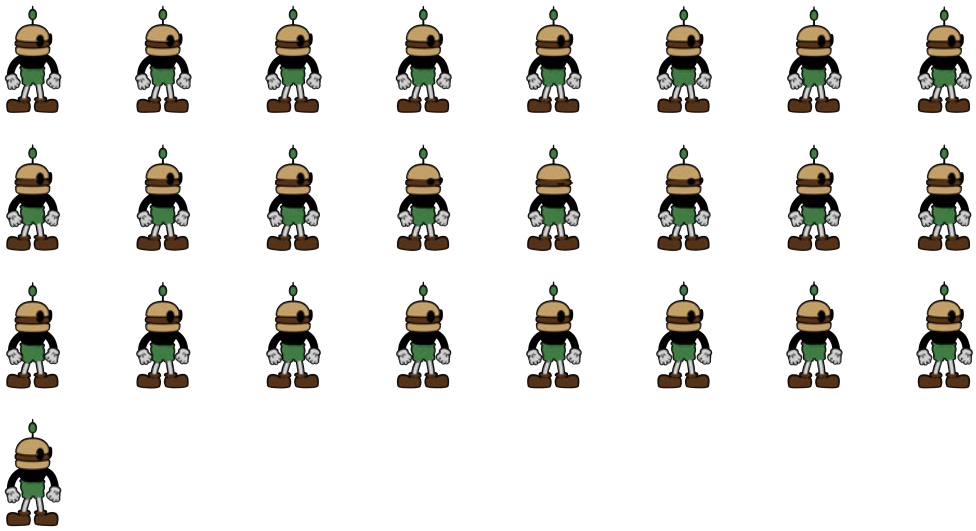


Abbildung .11: Vollständige Animations-Frames der Idle-Animation

Run-Animation

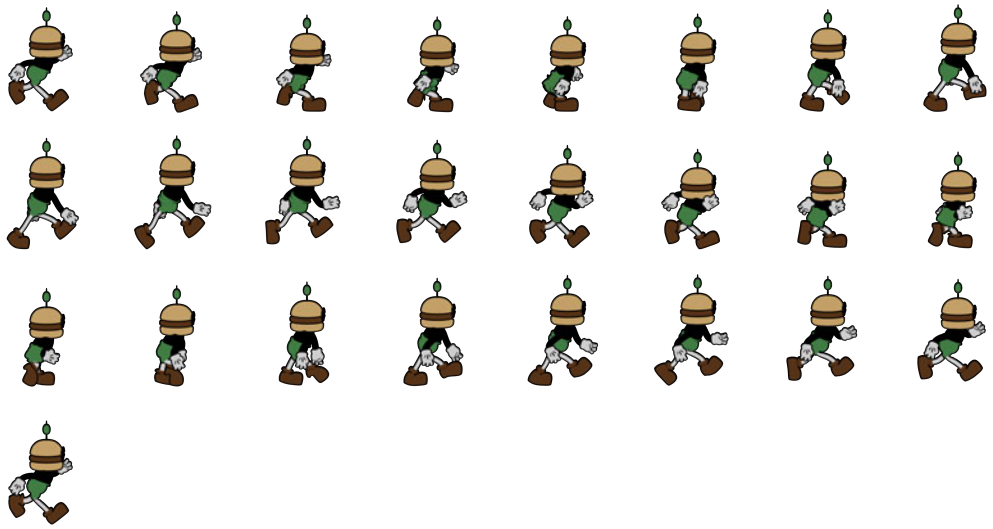


Abbildung .12: Vollständige Animations-Frames der Run-Animation

Dash-Animation

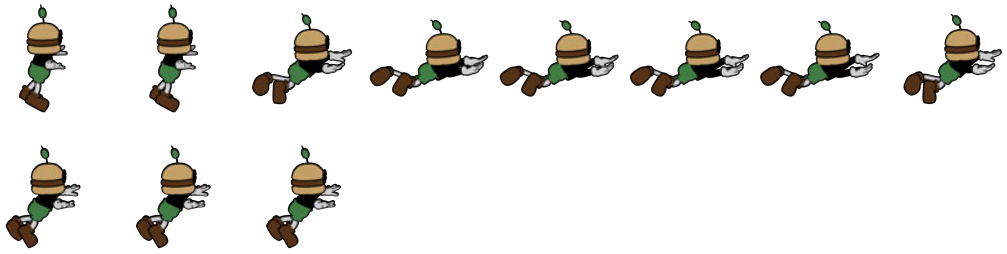


Abbildung .13: Vollständige Animations-Frames der Dash-Animation

VFX-Animation

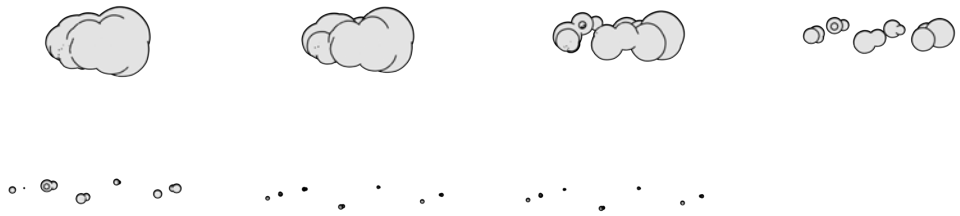


Abbildung .14: Vollständige Animations-Frames der VFX-Animation

Parallax-Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity-Skript | 0 Verweise
public class Parallax : MonoBehaviour
{
    //Referenz auf die Kamera der Szene
    public GameObject camera;
    //Erwähnter Multiplikator zur Einstellung der Effektstärke
    public float parallaxMultiplier;

    Unity-Nachricht | 0 Verweise
    void Update()
    {
        //Verschieben des Objekts, auf dem dem das Script liegt auf der x-Achse.
        //Beim x-Wert des Vektors wird die Bewegung der Kamera mit dem Parallax-Multiplikator verrechnet
        transform.position = new Vector3(camera.transform.position.x * parallaxMultiplier, transform.position.y, transform.position.z);
    }
}
```

Abbildung .15: Vollständige Animations-Frames der VFX-Animation

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Mittweida, den 26. September 2022