Einfluss von Software auf den Designprozess im Animationsfilm

Maximilian Penzinger

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Digital Arts

in Hagenberg

im November 2019

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 25. November 2019

Maximilian Penzinger

Inhaltsverzeichnis

Kurzfassung

Eine wichtige Grundlage im digitalen Film ist die gewählte Software. Diese entscheidet in allen Phasen der Produktion aufgrund der technischen Grundlage und der zur Verfügung stehenden Tools maßgeblich über den Look eines Filmes. Dabei steht eine breite Palette an Technologien, Werkzeugen und Programmen zur Verfügung, die große Unterschiede in der Art ihrer Verwendung und den daraus resultierenden Auswirkungen auf die visuelle Erscheinung eines digital kreierten Werks beinhalten können – sei es im Animationsfilm oder in anderen Bereichen der digitalen Kunst. Obwohl die verwendeten Programme im finalen Bild nicht immer erkennbar sind, beeinflussen sie zumindest den Arbeitsprozess des Animators. Die Ausprägung dieses Einflusses kann dabei stark variieren. Als digitales Medium simuliert Software gleichzeitig eine Vielzahl an analogen Technikern und Materialien, bringt jedoch auch gleichzeitig eigene Charakteristika und Werkzeuge mit sich. Die Rolle des Künstlers wird dabei an den Software-User übergeben. Abhängig vom gewählten Programm stellen viele Software-Pakete jedoch auch teil-autonome Systeme zur Verfügung, die dem User Arbeit abnehmen sollen. Dies kann eine deutliche Arbeitsersparnis bringen, gleichzeitig geht hier aber auch Kontrolle über das visuelle Endergebnis verloren. Komplexe Prozesse werden im Hintergrund des Programms vom User verborgen. Der Animator oder User selbst beeinflusst oft lediglich die Parameter, von denen das Bild abhängt. Viele der Design-Entscheidungen trifft somit nicht immer der Künstler selbst, oder er trifft sie nur unbewusst über die Tools des Progamms. Je konkreter und fortgeschrittener die Tools innerhalb der Software sind, umso weniger Kontrolle über den visuellen Stil bleibt oftmals. In manchen Fällen wird Software und insbesondere Künstliche Intelligenz somit selbst zum Autor visueller Werke.

Abstract

The chosen software is an important basis in digital film. On every single phase of production the technical foundations and the available software tools largely decide upon the visual look of a film. At the same time, a broad pallet of technologies, tools and programs is available to the user. These can have big differences in the way we use them and the resulting visual influence of a digitally created piece of art. Even though the used programs aren't always visible in the final film, they mostly do influence the workflow of the animator. The different emphasis of this influence can thereby vary heavily. Software as a digital medium simulates a multitude of analogue techniques und materials, yet it entails its own characteristics and tools. The role of the artist is thereby given to the software user. Depending on the chosen program many software packages provide autonomous systems that should save the animator a great deal of work. This can benefit the required time for a film production but control over the final visual result can also be lost in the process. Complex procedures in the background of a program are hidden from the user. The animator himself sometimes influences solely the parameters that the image depends on. Many design choices are not taken by the artist or are sometimes taken only involuntarily by the tools of a program. The more specific and advanced tools are inside of a software, the less available control over the visual style may be left. In some cases, software and particularly Artificial Intelligence becomes the author of visual art.

Kapitel 1

Einleitung

Als Werkzeug bietet Software so viele Möglichkeiten, wie kein Medium zuvor. Sie ist fester Bestandteil unserer Kommunikation, unseres Arbeitslebens und auch alle anderen Bereiche des menschlichen Lebens werden maßgeblich von digitaler Technologie beeinflusst. Dabei spielt ihre Anwendung in der Kunst nur eine Nebenrolle in ihrem großen Einsatzfeld. Dennoch basiert ein Großteil des heutigen Film- und Kunstschaffens auf Software und Computercode. Illustration, Animation, Design und Videoschnitt, jeder Bereich, der zuvor für analoges Arbeiten stand, ist heute durch eine digitale Arbeitsweise ersetzt. Gleichzeitig nehmen wir den Computer und die zugehörige Software als gegeben an. Gerade deshalb ist die Analyse unserer Verbindung als Künstler zur Software eine interessante Basis für das Verständnis digitaler Kunst. Kaum ein Bereich erhielt durch die Digitalisierung dabei so ein großen Wachstum wie der Animationsfilm. Wo vor 20 Jahren noch Einzelbilder von Hand auf Papier gezeichnet wurden, spielt heute der Computer eine essentielle Rolle. Arbeitsprozesse wurden vereinfacht und beschleunigt, die Zahl der digitalen Möglichkeiten scheint heute beinahe endlos. Neue Tools zur Animation, Texturierung, Generierung und zum Rendern von Inhalten entstehen dabei in rasantem Tempo. Diese Masterarbeit soll sich deshalb auf das Feld der 3D-Animation fokussieren und Fragen über die Herkunft und die Rolle der verwendeten Technologien beantworten.

Zugunsten einer besseren Lesbarkeit wird in dieser Arbeit zudem auf geschlechtsspezfische Formulierungen verzichtet. Die verwendete männliche Form bezieht sich somit immer auf weibliche und männliche Personen gleichermaßen.

1.1 Zielsetzung

Diese Arbeit soll verschiedene Zugänge zu Computersoftware als Werkzeug digitaler Kunst untersuchen und Unterschiede verschiedener Programme und Technologien in ihrem allgemeinen Workflow und ihrem Einfluss auf das

visuelle Endresultat aufzeigen. Es soll gezeigt werden, wie die unterschiedlichen Möglichkeiten von Animationssoftware den Stil aktueller Strömungen prägen und wo auch bewusst gegen diese Strömungen gearbeitet wird.

Als theoretische Grundlage für die Analyse soll unter anderem das Buch *Software Takes Command* [4] und weitere Arbeiten von Lev Manovich dienen. Anhand der dadurch gewonnenen Erkenntnisse über die Software als künstlerisches Werkzeug sollen einige weit verbreitete Software-Pakete und deren Architektur und Grundform analysiert werden. Dabei sollen außerdem verschiedene weitere für dieses Feld relevante Technologien und der Umgang mit diesen analysiert werden. Ziel dieser Arbeit ist es außerdem, die Rolle des Animators und in diesem Fall Software-Users zu erörtern und aufzuzeigen, welchen Stellenwert autonome Softwaresysteme im Arbeiten mit Animation haben können. Häufig vorkommende Prinzipien sollen analysiert und verglichen werden. Verschiedene Formen von Software und der jeweilige Zugang zur Animation sollen analysiert und verglichen werden. Dabei sollen sowohl Probleme, als auch Vorteile des jeweiligen Programms beschrieben werden. Einige Fallbeispiele in Form von Filmen und Projekten, die mit den beschriebenen Programmen umgesetzt wurden, sollen die Analyse dabei unterstützen.

1.2 Forschungsfrage

In diesem Kontext ergeben sich folgende Fragestellungen:

> Inwiefern wirkt sich technologischer Fortschritt und die heutige Technologie im Allgemeinen im Bereich der Animation – beispielsweise in Form von Animations-Software und Plugins – auf den kreativen Prozess und das finale Bild aus? Wie wird mit diesem Einfluss umgegangen und welche Rolle hat diese Entwicklung auf verschiedene Bereiche und Gattungen des Animationsfilms?

1.3 Aufbau dieser Arbeit

Zu Beginn dieser Arbeit sollen einige theoretische Grundlagen erörtert und Begrifflichkeiten definiert werden. Sowohl der Softwarebegriff an sich, als auch die Rolle des Computers im Laufe der letzten Jahrzehnte werden dabei aufgezeigt. Für verschiedene theoretische Bereiche, die im Zusammenhang mit Software relevant sind, sollen in Kapitel 2 Grundlagen geschaffen werden. Dazu gehört nicht nur der Begriff des Mediums, sondern auch die Relevanz der Schnittstelle zwischen Software und User – das Interface.

Dies bildet eine Basis für verschiedenste Bausteine von Software in Kapitel 3. Prinzipien moderner Programm-Struktur sollen aufgezeigt und mit

den theoretischen Grundsteinen in Verbindung gebracht werden. In Kapitel 4 beginnt schließlich die Analyse spezifischer Software-Pakete. Dabei sollen vor allem die Besonderheiten im Umgang mit dem jeweiligen Programm und die Unterschiede zu anderen gängigen Methoden untersucht werden. Kapitel 5 vervollständigt das vorherige Kapitel um Technologien, die nicht unbedingt unter dem Begriff einer 3D-Software fallen, aber dennoch interessante Aspekte im Arbeitsprozess aufbringen. Der Bereich der Künstlichen Intelligenz erhält im Umfeld aktueller Medienkunst besondere Aufmerksamkeit und soll hier ebenfalls Platz finden. Zuletzt soll ein Vergleich der verschiedenen Zugänge und Ideen im abschließenden Kapitel erfolgen.

Kapitel 2

Software Studies

In unserer heutigen Kultur sind digitale[1] Medien kaum noch wegzudenken. Fast jeder Bereich unserer Gesellschaft wird mittlerweile von Computersoftware beeinflusst oder gesteuert. Dabei wurden im Zuge der Digitalisierung eine Vielzahl an Technologien, die im Laufe des 19. und 20. Jahrhunderts aufkamen, durch digitale Software ersetzt [4, S. 4–6]. Auch unsere Kommunikation, Unterhaltung und ein Großteil unserer Arbeit beruhen auf digitalen Technologien. So hat sich Software in fast jeden Bereich unserer Gesellschaft verbreitet:

> „What electricity and the combustion engine were to the early twentieth century, software is to the early twenty-first century.“ [4, S. 2]

Dieser große Wandel macht sie zum zentralen Bestandteil unserer heutigen Kultur. Die große Bandbreite ihrer Anwendungsgebiete verankert solche Software-Lösungen tief in unseren sozialen, wirtschaftlichen und kreativen Strukturen. Dennoch, oder vielleicht auch genau wegen ihrer universellen Rolle in unserer heutigen Welt, scheint sie fast unsichtbar. Als Medium und theoretisches Forschungsgebiet erlangten Software und digitale Medien bis Anfang des einundzwanzigsten Jahrhunderts kaum Aufmerksamkeit. Zu neuartig und zu schwer greifbar wirkte das Digitale zu seinen Anfängen der 90er-Jahre. Erst durch Pioniere wie Friedrich Kittler, Matthew Fuller und Lev Manovich fand Software als eigenständiges Medium vermehrt Beachtung.

[1] Als Gegensatz zum Begriff des „Digitalen“ steht das Wort „analog“, das in diesem Zusammenhang „kontinuierlich“ oder „stufenlos“ bedeutet [36].

2.1 Der Software-Begriff

Doch wie lässt sich der Begriff „Software“ eingrenzen? Zum einen wird Software als ein Set aus Daten und Anweisungen definieren, mit denen spezifische Aufgaben erledigt werden können. Dieser funktionale Teil von Software und der damit verbundene Output grenzt sie ab von einfachen Datensätzen oder Algorithmen. Software ist an sich auch ausführbar und besitzt dadurch einen aktiven und einen inaktiven Zustand. So könnte man Software ebenfalls als variablen Teil eines Computers sehen. Die Hardware könnte dann als der invariable Teil beschrieben werden. Beide Teile stehen dabei in Wechselwirkung [54].

Jedenfalls ist der Software-Begriff sehr breit definiert und reicht von sehr rudimentären, beinahe unsichtbaren Elementen wie dem Prozessorcode oder einem Betriebssystem bis hin zu verschiedensten Anwendungen als Schnittstelle zum Benutzer. Als System-Software wird dabei Computer-Code bezeichnet, der eine Plattform für andere Software bildet. System-Software stellt die Grundvoraussetzungen für das Arbeiten. Sie dient somit als Schnittstelle zwischen der Hardware und den eigentlichen Programmen, mit denen der User in Interaktion steht. Diese weiteren Programme, die auf der Basis der System-Software aufbauen, können widerum zur Applikations-Software gezählt werden .

Applikations-Software ist User-orientiert und ermöglicht diesem verschiedenste Anwendungen. Auf diesen Bereich des Software-Begriffs und dabei im speziellen auf sämtliche für die Produktion von computeranimierten 3D-Animationsfilmen relevante Software und Computertechnologie fokussiert sich diese Arbeit. Auch wenn sie genau genommen Teil des Gesamtprozesses in der Filmproduktion sind, werden andere Typen von Programmen wie etwa Verwaltungs- und Kommunikationssoftware oder System-Software nicht zur genaueren Analyse herangezogen. Vielmehr soll der Fokus auf das kreative digitale Arbeiten gelegt werden, wobei eben hier 3D Software wie Maya, Houdini oder Cinema 4D als Werkzeug des Filmschaffenden gesehen werden. Ihre Einflüsse auf den Arbeitsprozess und die Arbeitsweise eines Users mit dem Programm, sowie die Eigenarbeit die Software an dieser Stelle oft unabhängig vom User leistet sollen genauer untersucht und verglichen werden.

Eine ähnliche Unterscheidung in Hardware und Software ist in analogen physischen Medien häufig nicht möglich. In diesem Zusammenhang analysiert Lev Manovich, dass ein früherer analoger Film-Projektor gleichzeitig die Hardware, als auch die Rolle heutiger Medien-Player-Software in einem einzelnen Konstrukt übernimmt [4, S. 91–93]. Ein Buch hat, im Gegensatz zu heutigen digitalen Medienformen, eine feste Reihenfolge und Anzahl der Seiten. Der flexible und variable Software-Teil digitaler Medien ist in analogen Medien fixiert und nicht vom Benutzer veränderbar, ohne das Medium an sich (physisch) zu ändern.

2.2 Software als Medium

Ein Medium[2] kann durch zwei Teile definiert werden: Das eigentliche Material, auf dem die Information festgehalten wird – in der Malerei ist das beispielsweise eine Leinwand oder Papier – , und ein oder mehrere Werkzeuge, die verwendet werden, um die Information aufzuzeichnen – das können Pinsel, Feder oder andere typische Utensilien sein [4, S. 204–210]. Hier, könnte man argumentieren, unterscheiden sich digitale Medien grundlegend von analogen Medien. In der digitalen Kunst stellen Software und Computercode sowohl das Material, in dem Informationen als Daten gespeichert werden, als auch das eigentliche Werkzeug in Form eines Software-Tools dar. Gespeichert wird in Form von Datensätzen und -strukturen, beziehungsweise auf kleinster Ebene als Binärcode. Die von der Software zu Verfügung gestellten Werkzeuge können unterschiedlich aussehen. Grundsätzlich wird aber auch hier häufig auf das Vorbild analoger Werkzeuge zurückgegriffen (vgl. Abschnitt 2.4). Gleichzeitig könnte man bei genauerer Analyse des Digitalen die zugrunde liegenden Datenstrukturen als das Material sehen, aus dem ein digitales Medium besteht. Diese unterscheiden sich je nach Medium und können sich als File-Formate wie png oder obj manifestieren. Die Werkzeuge werden im digitalen schließlich durch Software beziehungsweise Algorithmen ersetzt.

Obwohl die Grunddefinition eines Mediums also gegeben ist, sieht Lev Manovich den Begriff eines *digitalen* Mediums in seiner Analyse generell kritisch. Software bezeichnet er vielmehr als das eigentliche Werkzeug eines digitalen Mediums, das auf verschiedenen Mediencontent angewandt wird [4, S. 152]:

> The "properties" of digital media are defined by the particular software as opposed to solely being contained in the actual content (i.e., inside digital files).

Er bezeichnet das digitale Medium weiter als „Resultat einer allmählichen Entwicklung und Ansammlung einer großen Anzahl verschiedener Software Techniken, Algorithmen, Datenstrukturen und Interface Konventionen". Dabei unterscheidet Manovich bei digitalen Techniken zwischen einer *mediums-abhängigen* und *mediuims-unabhängigen*[3] Form. Während erstere eben einem konkreten Medium zugeschrieben sind, – beispielsweise nennt er die Technik des Extrudierens, die ausschließlich an einem dreidimensionalen Objekt angewandt werden kann – funktionieren mediums-unabhängige Techniken allgemein, werden jedoch von Medium zu Medium unterschiedlich umgesetzt. Als Beispiele nennt er hier etwa das Kopieren, Einfügen und Suchen

[2] Den Begriff des „Mediums" beschreibt Knut Hickethier ganz allgemein als „alle Mittel, derer wir uns beim Kommunizieren bedienen" [3, S. 7].

[3] Vom englischen *media-dependant* beziehungsweise *media-independant* übersetzt [4, S. 113].

von Inhalten, das je nach Aufbau eines Mediums eine Unterschiedliche Auswirkung haben kann (man kann sowohl nach einem Wort in einem Text, einem Objekt in einer 3D-Szene oder nach einem Muster als Bildinhalt suchen).

Die Fluidität der digitalen Software unterscheidet sie als Medium schließlich ebenfalls vom Analogen. Während ein analoges Medium durch seine physische Beschaffenheit in Funktion und Materialität beschränkt ist, werden digitale Werkzeuge laufend verändert. Im Gegensatz zum Analogen erreicht das Digitale hier nie eine finale Form, sondern passt sich ständig den aktuellen Anforderungen und Entwicklungen an.

2.3 Materialität eines Mediums

Sowohl im Analogen, als auch im Digitalen bringt jedes Medium auch verschiedene Techniken[4] mit sich. Noch vor einigen Jahrzehnten gab dabei das jeweilige Medium, in dem gearbeitet wurde, maßgeblich das resultierende visuelle Erscheinungsbild vor. Während im Bereich der Grafik früher hauptsächlich Lithografie und Offsetdruck verwendet wurden, arbeitete man im Film mit analogen Kameras, oder Techniken wie der Cell-Animation oder Stop-Motion. Jedes dieser Materialien bringt eine Ästhetik mit sich, die im Resultat fast immer auch das gewählte Medium erkennen lässt. Von Software generierte Bilder oder Videos haben allerdings für sich stehend keine wirkliche, physische Materialität und besitzen zu allererst nur ihre digitale Form [4, S. 113]. Im Gegensatz zum analogen Material weisen diese Datensätze allein keine eigenständige Qualität auf. Sie speichern nur die vorhandene Information und können erst durch weitere Verarbeitung materialisiert werden. Diese digitale Form ist ein Alleinstellungsmerkmal in der Kunst. Dennoch besitzen digitale Filme oft einen Stil, der sie auch als solche erkennbar macht. Hier werden die verwendeten digitalen Tools im finalen Bild sichtbar. Deutlich sichtbar ist diese bewusste digitale Materialität beispielsweise in Nikita Diakurs Film *Ugly* [23] (vgl. Abschnitt 4.2.1) oder David O'Reillys *Please Say Something* [18] (vgl. Abb. 2.1). Es scheinen unter anderem Interface-Elemente oder andere digitale Artefakte im finalen Bild auf und geben somit die digitale Herkunft der Bilder deutlich preis. Vor allem O'Reilly betont in seinem Essay *Basic Animation Aesthetics* [48] die bewusste Entscheidung hinter dem gewählten digitalen Stil:

> „The film makes no effort to cover up the fact that it is a computer animation, it holds an array of artifacts which distance it from reality, which tie it closer to the software it came from. This idea is in direct opposition to all current trends in animation, which take the route of desperately trying to look real, usually by rea-

[4] Eine Sammlung an Techniken und Tools wird dabei als eine Technologie bezeichnet.

Abbildung 2.1: Der Film *Please Say Something* [18] von David O'Reilly verwendet bewusst digitale Artefakte wie das Aliasing an harten Kanten als Stilmittel.

> listic lighting and rendering, or by forcing a hand-made or naive appearance."

Typische Merkmale des Digitalen können etwa an einer mathematischen Präzision in Form, Farbe oder Bewegung festgemacht werden. Aber auch Artefakte der digitalen Tools können auf ihren Ursprung hinweisen. Im Regelfall gelten diese Eigenschaften aber als fehlerhaft und werden in den meisten Produktionen vermieden. In der Avangrade der digitalen Kunst werden hingegen gerade diese Fehler[5] und die digitalen Artefakte häufig als deutliche Thematik in der Kunst untersucht. Software bietet also sowohl die Möglichkeit eines realen, imitiert analogen Stils, als auch ihre inhärente Digitalität. Die Möglichkeiten scheinen dadurch fast grenzenlos.

2.4 Vom Analogen zum Digitalen

Als relativ neues Medium existiert Software erst seit den 1940er Jahren. Bis zur ersten kreativen und grafischen Anwendung der Computertechnologie dauerte es noch einmal fast 30 Jahre. Dennoch läuft der technische Fortschritt im Bereich der Computergrafik seither rasend schnell. Es ent-

[5] Nicht umsonst wude als Thema des Ars Electronica Festivals 2018 *ERROR—The Art of Imperfection* gewählt [38].

stand also innerhalb einiger weniger Jahrzehnte ein völlig neues und zu Beginn fremdartiges Medium. Aus diesem Grund immitiert heutige Software zu großen Teilen analoge Workflows, die schon viel länger bestanden, als moderne digitale Werkzeuge. Die Ähnlichkeit zur bekannten Funktionalität der analogen Tools brachte das noch recht fremde Medium etwas näher an bereits existierende Formen der grafischen Kunst.

In fast allen Programmen, die heute zur Verfügung stehen lässt sich dieses Prinzip zu großen Teilen erkennen. Adobe Photoshop [25] beispielsweise bietet sowohl Pinsel, als auch Buntstift, Stempel und Radiergummi als digitales Werkzeug an und auch die Ästhetik vieler digitaler Filme und Bilder greift auf analoge Grundsätze zurück [4]. Auch in ihrer Funktionsweise gleichen die digitalen Werkzeuge den analogen meist recht stark. Die Art der Verwendung ist somit intuitiver und die Materialität des Bildes bekommt oft einen analog-wirkenderen und damit vertrauten Charakter. In ihrer Grundidee zielen viele dieser Tools also darauf ab, analoge Medien zu immitieren und digital zu replizieren.

> „If today we consider all the digital media created by both consumers and by professionals—digital photography and video shot with inexpensive cameras and cell phones, the contents of personal blogs and online journals, illustrations created in Photoshop, feature films cut on Avid, etc.—in terms of its appearance digital media indeed often looks exactly the same way as media before computers.“ [4, S. 59]

Je weiter man von digitalen, zweidimensionalen Bildern in Richtung des dreidimensionalen Bewegtbildes schreitet, umso technischer werden auch die verwendeten Werkzeuge. Hier kann sich oft nicht mehr an der Funktionsweise analoger Tools orientiert werden. Digitale Lösungen werden erstellt, ihre Verwendung ist allerdings oftmals weniger intuitiv. Einige Dinge sind uns aber auch mittlerweile aus allen digitalen Bereichen geläufig und deshalb einfach verständlich. Im Zusammenhang des Analogen sind diese jedoch keineswegs eindeutig. Vielmehr sind sie in der heutigen Generation jahrelang erlernt.[6]

Schlussendlich hat das digitale Arbeiten aber einige Vorteile, die im Analogen kaum umsetzbar wären. Zum einen benutzen fast alle digitalen Animationen Keyframes – sofern die Objekte und Bilder nicht prozedural generiert werden (vgl. Abschnitt 5.1). Ähnliches wurde schon zu den Anfangszeiten Disneys praktiziert. Damals wurden die Keyposen einer Animation vom Animator persönlich gezeichnet, die dazwischenliegenden Bilder dann aber unter großem Zeitaufwand von Assistenten vervollständigt. Dadurch wurde gewährleistet, dass die Grundbewegung der Animation und der Ausdruck

[6] Personen die im Umfeld digitaler Technologie aufgewachsen sind bezeichnet man daher als *Digital Natives*.

des Animierten Charakters oder Objekts über den gesamten Film hinweg stimmt. Erst durch diese „Inbetweens“ wirkt die Animation schließlich aber flüssig.

Diesen Prozess des *Inbetweenings* übernimmt in der digitalen Animation im Normalfall der Computer. Des User setzt Keyframes, deren Werte schließlich interpoliert werden. Auch wenn eine reine Interpolation häufig von Hand nachgebessert werden muss, erleichtert diese Technologie den Prozess deutlich. Es muss nicht mehr jeder Frame von Neuem gezeichnet werden. Außerdem gewährleistet die Software, dass Proportionen, Ausdruck und Farbe des Charakters von Bild zu Bild erhalten bleiben. Dies kann allerdings durchaus auch Nachteile haben. Eine Frame für Frame handgezeichnete Animation zeichnet sich oft durch deutlich stärkere Emotionalität und Dynamik aus. Diese Ausdrucksmöglichkeiten müssen im Digitalen oft mühsam in ein Rig integriert werden, um mit der analogen Variante konkurrieren zu können.

2.5 Technische Limitierung

Die Limitierungen des digitalen Mediums sind in vieler Hinsicht weitaus geringer und die Möglichkeiten reichen deutlich weiter. Ganz im Gegenteil verleiten viele moderne Tools immer mehr zu einem realistischen Look, und das schlichtweg aus dem Grund, dass die Möglichkeiten dazu immer leichter Verfügbar sind:

> „The problem is that there is simply too much power and very little control, essentially you get too much for free. Other forms of animation have benefited from their inherent limitations, but largely these do not exist with 3d.“ [48]

Es lässt sich somit auch ein stilistischer Trend in aktuellen Animationen erkennen. Durch die mittlerweile recht große Bandbreite an Tools zum realistischen Rendering wird die Grenze zum vollkommen real-wirkenden Bild immer kleiner. Auch die immer besser werdenden Renderer tragen zu diesem Trend bei. Es entstehen zunehmend realistischere Animationen, wobei es tendenziell und vor allem im kommerziellen Film immer seltener zu einer Stilisierung kommt. Die große Zahl an bereits vorhandenem Bildmaterial in unserer heutigen Kultur und dem Internet macht es außerdem schwer, neue Stile für sich zu erfinden:

> „By the end of the twentieth century, the problem became no longer how to create a new media object such as an image; the new problem was how to find the object which already exists somewhere.“ [5, S. 401]

Ein starker Kontrast lässt sich zwischen Avantgarde und Mainstream erkennen. Während kommerzielle Filme sich stark an den gut verfügbaren

Tools zur Animation orientieren, wird in der Avantgarde oftmals ein anderer Weg gewählt. Hier werden Konzeption, Ästhetik und Stil des Films deutlich vor die technische Umsetzung gestellt. Somit spiegelt sich der technische Fortschritt und die dadurch verfügbaren Tools auch immer in aktuellen Trends und Strömungen wider. Es besteht die Gefahr, die gestalterischen Möglichkeiten an dem zu messen, was durch die Software vorgegeben wird.

2.6 Interface

Um Analysieren zu können, wie Software auf den User Einfluss nimmt, muss auch die Schnittstelle zum User genauer betrachtet werden. Diese bildet das GUI[7], welches dem Benutzer die vorhandenen Daten und Tools grafisch sichtbar zur Verfügung stellt. Gleichzeitig vereinfachen Interfaces komplexe Algorithmen zu einfachen Symbolen und Buttons. Als Digital Natives verwenden wir intuitiv Interfaces und müssen und zu den meisten gängigen Symbolen und Funktionen kaum Gedanken machen. Tools wie ein einfaches Auswahl-Werkzeug oder der Mauszeiger an sich sind uns allgemein bekannt. Ihre Funktion ist in allen Programmen gleich oder ähnlich zu verwenden und lässt somit wenig Spielraum für Interpretation. Auch andere GUI-Elemente wie das öffnen von Drop-Down-Menüs oder Fenstern ist heute Standard in jedem Programm. Lev Manovich beschreibt hier das Interface als „Teil des kognitiven Modells eines Benutzers“ [5, S. 63–68]. Abhängig von der Usability des Programms und der allgemeinen Vorerfahrung mit Software und gelingt es einem Benutzer mehr oder weniger gut, ein Interface beim Arbeiten gedanklich auszublenden und intuitiv zu benutzen. Gleichzeitig besitzen Buttons in einem Interface von sich aus eine Materialität und imitieren in ihrem Aussehen und ihrer Funktionalität oft das Analoge (vgl. Abschnitt 2.4). Ihr häufig dreidimensional wirkendes Aussehen simuliert eine darunter liegende Mechanik und erweckt dadurch den Anschein eines zuverlässigen Hebels. Diese Funktion ist in analogen Systemen eindeutig mit ihrer Funktion verbunden. Im Digitalen wird genau das nur imitiert, da die darunterliegende Funktion sehr wohl variabel und viel komplexer als von außen sichtbar sein kann [2, S. 32]:

> „Interface buttons disguise the symbolic arbitrariness of the digital mediation as something solid and mechanical in order to make it appear as if the functionality were hardwired: they aim to bring the solid analog machine into the interface.“

Wie viel wirklich hinter einem jeweiligen Button im Interface steckt wird beispielsweise in Houdini deutlich. Unter dem Menüpunkt *Particles* kann ein *Source Particle Emitter* erstellt werden. Bei einem Blick in das zugehörige

[7] Graphical User Interface, im folgenden nur als Interface bezeichnet.

Node Netzwerk wird deutlich, wie viel an Information durch einen einzelnen Klick im Hintergrund erstellt wurde. Dies gilt für eine Vielzahl der bereitgestellten Funktionen in heutiger Software und vereinfacht dem Benutzer das Arbeiten mit komplexen Algorithmen und Programmen.

Heute tritt dieses analog-mechanische Aussehen in Interfaces weniger auf, als es noch vor einigen Jahren zu beobachten war. Das Design aktueller Programme und Betriebssysteme ist mittlerweile schlichter und einfacher geworden. Dies könnte einerseits durch eine größere Übersichtlichkeit oder andererseits auch durch die mittlerweile für viele selbstverständliche Funktion eines Interfaces und seiner Funktion erklärt werden. Benutzer finden sich heute auch ohne die simulierte Mechanik digitaler Buttons zurecht. Allgemein laden uns Buttons also ein, sie zu drücken und navigieren den User dadurch automatisch durch das komplexe GUI-System eines Programms. Je intuitiver und klarer ein Button, umso einfacher und klarer wird auch die Nutzung der darunterliegenden Funktion im Programm.

2.7 Die Rolle des Künstlers

Welchen Einfluss Software auf einen kreativen, künstlerischen Prozess nimmt, ist schon seit den Anfängen erster digitaler Technologien Thema zahlreicher Forschung [1]. Während Software also als Tool verschiedene Funktionen bietet, ist der User und somit der Künstler für die Grundidee und die eigentliche Umsetzung eines Werkes verantwortlich. Diese beiden Rollen – die der Software als Werkzeug und die des Users als Künstler – stehen in ständigem, engen Zusammenspiel. Das eine ist vom anderen abhängig. Das Tool wird dabei zu einer Erweiterung des Künstlers, sowohl körperlich, als auch geistig [6, S. 63–64]. Körperlich, weil Pinsel, Radiergummi und andere Werkzeuge im Analogen wie im Digitalen uns zusätzliche physische Möglichkeiten bieten, Kunst zu schaffen oder zu verändern. Aber auch geistig entsteht eine Verbindung von Tool und Artist. Das Verwenden des Tools passiert beim Arbeiten intuitiv. Wir denken dabei nicht explizit über unser Werkzeug nach, sondern sehen es als eine logische Erweiterung unseres Körpers. In der Kunst ist somit nicht nur die Idee und Vision des Künstlers, sondern auch die Erforschung und das Experimentieren mit dem verwendeten Tool, und ein Reflektieren darüber ein häufiger Ansatz. Die Rolle des Künstlers liegt also neben dem intellektuellen Inhalt eines Werkes auch in der Umsetzung über ein Tool.

Kapitel 3

Architektur digitaler Software-Tools

3.1 Spezialisierung vs Modularität

Während die Verwendung verschiedener Medien-Programme und das Erstellen von digitalen Inhalten ein vertieftes Wissen im Bereich der Programmierung erforderte, werden viele Programme heute zunehmend benutzerfreundlich. Komplexe Vorgänge werden hinter einem Interface versteckt und automatisiert. Gleichzeitig werden die Möglichkeiten zur Generierung von Bildinhalten immer komplexer und vielseitiger. Heute gibt es eine sehr breite Palette an Tools für den digitalen Film. Während Pakete wie Maya eine durchaus große Tiefe an spezialisierten Tools bieten, zielen viele andere Programme auf eine eher vielseitige, jedoch nicht weniger umfangreiche Verwendung der einzelnen Tools ab. Diese beiden Gruppen verfolgen teils recht unterschiedliche Software-Ideologien.

Programme, die stark in die Tiefe gehen, glänzen meist in sehr spezifischen Problemstellungen. Ihr Anwendungsgebiet umfasst vor allem komplexe Simulationen mit hohem Realitätsgrad, die an vielen Stellen zum Einsatz kommen und deshalb von einer Grundästhetik ausgehen können. Es wird für bestimmte Bereiche jeweils ein bestimmtes Tools zur Verfügung gestellt. Ein Beispiel hierfür ist Autodesk Maya [27], das in Abschnitt 4.1 noch nähere analysiert werden soll. Für die Simulation von Flüssigkeiten bringt Maya das Paket *Bifrost*, für Fell und Haare *XGen*. Diese Tools sind in ihrem jeweiligen Bereich sehr gut ausgebaut und bringen zahlreiche Optionen mit sich. Sie sind jedoch nicht immer mit anderen Tools innerhalb der Software kompatibel. Außerdem erfordert diese Art von Tool meist ein sehr präzises Setup, das bei einem späteren Arbeitsschritt oft nicht mehr verändert werden kann. Bei Änderungen der Grundgeometrie für eine Fell-Simulation mit XGen muss beispielsweise ein Großteil des vorhandenen Setups neu gebaut werden und zum selben Ergebnis zurück zu kommen. Die Arbeitsweise ist

also stark destruktiv. [1] Kleine Änderungen sind in diesen Tools oft mit viel Aufwand verbunden.

Eine Software-Architektur, die dieser Problematik entgegenzuwirken versucht und aktuell immer häufiger verwendet wird, ist das Node-basierte arbeiten (vgl. Abschnitt 3.5). Hier bleiben die Tools stark variabel. Diese Arbeitsweise ist meist stark flexibel und modular aufgebaut. Einzelne Funktionen können beliebig wiederverwendet und verknüpft werden. Dadurch entsteht potentiell eine höhere Fehlerquelle. Nicht jede Node macht in jedem Zusammenhang Sinn. Dieses Risiko wird beim Node-basierten Arbeiten dem User überlassen. Dennoch ergeben sich zahlreiche Möglichkeiten innerhalb des Programms. Änderungen können auch zu einem weiter fortgeschrittenen Zeitpunkt noch ohne größeren Aufwand vorgenommen werden, indem die betroffene Node geändert oder ausgetauscht wird.

3.2 Scripts

Eine Möglichkeit, vorhandene Software Pakete in ihrer Funktionalität zu erweitern, sind Scripts. Durch Scripting können Objekte in der Software mit Code erstellt oder manipuliert werden, aber auch fast alle anderen Funktionen, die auch über das programminterne Interface erreichbar sind, können aufgerufen werden. Scripts werden in fast allen größeren Software-Paketen unterstützt. Oft sind auch verschiedene Script- und Programmiersprachen verfügbar. Häufig kommt dabei Python zum Einsatz, in Maya gibt es zusätzlich die sogenannte *Maya Embedded Language* – kurz MEL. Zur Verfügung stehen fast alle Funktionen, die auch über das programminterne Interface erreichbar sind.

Der Funktionsumfang reicht auch hier von kleinen Anweisungen bis zu umfangreicheren Scripts, die ihre eigenen Interfaces mitbringen. An dieser Stelle ist die Grenze zu einem Plug-In nur noch gering. Oftmals werden mit Scripts repetitive Tasks innerhalb des Programms automatisiert. So können Abläufe, die im Programm häufig vom Benutzer benötigt werden, im Menü als einfacher Button bereit gestellt werden. Darunter liegt lediglich die selbe Art von Script-Code, wie auch bei allen anderen bereits vorhandenen Menü Punkten. Die Scripting-Schnittstelle bietet dem User in Programmen somit ein mächtiges Tool um Software beliebig zu erweitern. Sowohl eine Kenntnis über die verwendete Programmier- oder Scriptsprache, als auch ein umfangreiches Wissen über die verwendete Software und ihre Funktionen stellen hier jedoch eine wichtige Voraussetzung dar. Nicht jeder User ist also in der Lage, Scripts zu seinem Vorteil zu nutzen.

[1] Als destruktives Arbeiten beschreibt man Befehle in einem Programm, die nicht mehr auf vorherige Arbeitsschritte schließen lassen. Im Gegensatz dazu lässt ein nondestruktiver Workflow auch zu einem späteren Zeitpunkt Änderungen an beispielsweise der Grundgeometrie oder anderen Parametern zu.

3.3 Plug-Ins

Eine zusätzliche Möglichkeit, die gebotenen Werkzeuge einer Software zu erweitern sind Plug-Ins. Sie arbeiten mit vorhandenen Schnittstellen und kreieren dadurch neue Funktionen. Der Vorteil von Plug-Ins ist, dass nicht-vorhandene Funktionalitäten von externen Quellen hinzugefügt werden können. Der User beziehungsweise die Community der Software-User kann sich also benötigte Tools selbstständig erweitern [49]. Ein vertieftes Wissen über die darunterliegende Funktionsweise wird im Normalfall nicht benötigt. Die meisten Plug-Ins arbeiten mit einem einfachen Interface hinter dem oft recht komplexe Funktionen stehen. Im Gegensatz zu den selbst erstellten Scripts sind es hier also oft auch umfangreichere Aufgaben, die von Plug-Ins gedeckt werden. Die Bandbreite reicht von kleinen visuellen Änderungen – das Plug-In *ColorKey* für Maya ermöglicht beispielsweise ein einfärben von Keyframes, was das Arbeiten an komplexen Animationen stark erleichtern kann – bis hin zu Render-Engines und komplexeren Simulations- oder Animations-Tools wie *Phoenix FD* oder *Krakatoa*, die eine verbesserte Partikelsimulation bieten.

In den meisten Fällen müssen Plug-Ins vom User selbst installiert werden. Während vieles in Foren kostenlos zur Verfügung steht, gibt es auch immer mehr Anbieter von erwerblichen Plug-Ins. Oft sind Plug-Ins aber auch im Lieferumfang der Software enthalten und können nach belieben aktiviert und deaktiviert werden. Maya organisiert beispielsweise einen Großteil seiner Funktionsbausteine als Plug-Ins, die im zuständigen Menü erreichbar sind. So kann die Core-Applikation bewusst klein und effizient gehalten werden, ohne die Funktionalität einzuschränken.

3.4 Layers

Eine Funktion, die schon seit langer Zeit fester Bestandteil vieler Medien Programmen aller Art ist, sind Layers. Dabei wird dem User ermöglicht, statt auf das Gesamtbild, auf einzelne Layers, also Bildebenen zuzugreifen. Lev Manovich sieht Layer dabei als „Container für die Einzelelemente einer Composition“, und als wichtiges Tool zur Unterteilung von Information in einzelne Bestandteile [4, S. 142–143]. Dies vergleicht er sowohl mit generellen Konzepten in der Computerprogrammierung, – auch hier wird im Regelfall Softwarecode in logische Teile gespalten – er sieht aber auch deutliche Parallelen zu Techniken der Cel Animation. In Adobe Photoshop wird außerdem durch den *Layer Mode* (in der deutschen Fassung übersetzt als ‚Mischmodus‘) definiert, wie einzelne Ebenen vom Programm zu einem einzelnen Bild kombiniert werden sollen. Häufigste Variante der Kombination ist ein Überlagern mit Transparenz – im Programm lediglich als *Normal* bezeichnet. Andere Optionen wenden unterschiedlichste Berechnungen bei

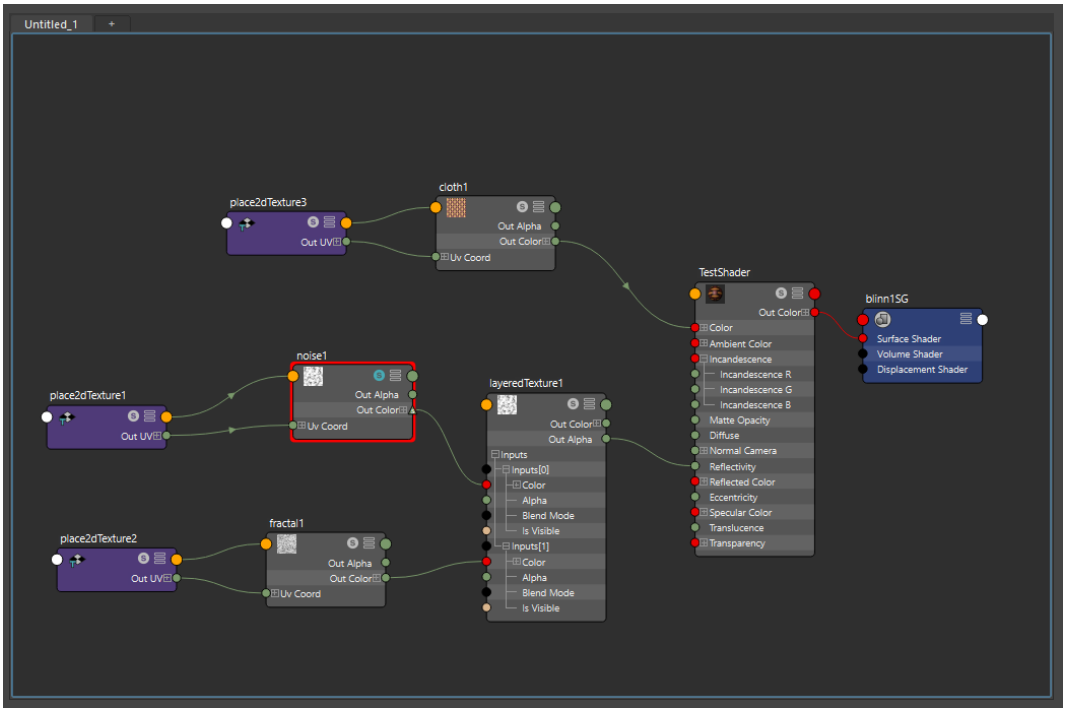


Abbildung 3.1: Screenshot aus Maya 2018 [27]. Einfaches Node System für einen Shader im Maya Hypershade.

der Kombination der Pixel an um zum Endresultat zu kommen. In jedem Fall kommt es durch eine Ansicht mit Layern in einem Programm zu einer Abstraktion der Bildinformation. Das Arbeiten fokussiert sich hier auf das Anpassen und spätere Zusammenfügen der Einzelteile. Die meisten Filter und Effekte können problemlos auf einzelne Layer angewendet werden. Andere erfordern hingegen eine Anwendung auf das Gesamtbild um korrekt zu arbeiten. Erst beim Exportieren oder auf konkrete Anweisung des Users werden die Bildebenen zu einem Gesamtbild zusammengefügt. Dabei gehen die Informationen der einzelnen Layer jedoch normalerweise verloren. Das Zusammenfügen ist in der Regel irreversibel.

3.5 Nodes

Mit Nodes besitzen viele Programme ein System, das ursprünglich aus Datenstrukturen vieler Computerprogramme bekannt ist. Während einfache Datenstrukturen wie Bäume oder Listen am Computer für das Speichern in Datenbanken zum Einsatz kommen können, verwenden die meisten 3D-Programme ebenfalls solch ein komplexes Netzwerk an Nodes. Die Nodes sind dabei ein Speicherort für Datensätze und können widerum Verbindun-

gen zu anderen Nodes haben. Es entsteht ein Datenfluss von einem Knotenpunkt zum nächsten. Obwohl im Hintergrund vieler Programme große Teile auf Nodes basieren, wird dieser Workflow nicht immmer auch an den User übergeben. In Maya beispielsweise wird erst im Node-Editor sichtbar, welche Systeme wo als Nodes aufscheinen und wie sie mit anderen Nodes verbunden sind. Herkömmliche Arbeitsschritte können dabei ganz normal über das Menü vorgenommen werden. Im Vergleich dazu ist das Arbeiten in Mayas Hypershade, wie auch ein Großteil der User-Schnittstelle von Houdini [32] oder Nuke [33] fast ausschließlich Node-basiert.

Es können verschiedenste Nodes erstellt werden, die jeweils über unterschiedliche Inputs, Parameter und Outputs verfügen. Während manche Nodes dabei lediglich das Speichern von Daten übernehmen, sind andere für komplexe Verarbeitungsschritte der Daten zuständig. Durch dieses Node-Netzwerk passiert also eine Abstrahierung ähnlich wie die der Layer in Programmen wie AfterEffects [24]. Dies kann zu einer besseren Übersicht und einem einfacheren, klareren Arbeitsprozess im Programm führen. Oftmals ist diese Arbeitsweise auch mit einem non-destruktiven Workflow gekoppelt, da die Ansicht des Node-Netzwerks eine einfache Darstellung und einen schnellen Zugriff zu vorhergehenden Arbeitsschritten ermöglicht [4, S. 307–312].

3.6 Rendering

Ein weiterer Prozess der als wichtiger Grundbaustein im Arbeiten mit 3D-Software gilt, ist das Rendering. Hier werden die in der Szene erstellten dreidimensionalen Objekte zu einem finalen zweidimensionalen Bild berechnet. Diese Berechnungen erfolgen aufgrund verschiedener Parameter, die der User konfigurieren kann. Das visuelle Ergebnis ist dabei stark obstruiert und oft nur schwer erkennbar. Je nach Komplexität der Szene kann der Renderprozess hier durchaus lange dauern. Ein sofortiges Ergebnis ist in vielen 3D-Softwares nicht möglich. Diese Verzögerung und das nicht immer vollständig kontrollierbare Ergebnis haben also deutlichen Einfluss auf den Workflow eines Users. Nur beim Rendern in Echtzeit (siehe Abschnitt 4.5) ist ein völlig freies Gestalten möglich, da hier Auswirkungen verschiedener Parameter im Programm gleichzeitig sichtbar sind. Generell ist der Renderprozess ein Schritt, der an sich zur Gänze vom Computer und der Render Engine übernommen wird. Vergleicht man dies zu ähnlichen analogen Prozessen, bei denen Einzelbilder per Hand gezeichnet und koloriert wurden, geht auch hier deutlich Kontrolle an die Software im Tausch gegen einen schnellen Arbeitsprozess verloren. Dennoch sind moderne Shader und Renderer so weit entwickelt, dass ein einfaches Arbeiten und solide Ergebnisse der Regelfall sind.

Kapitel 4

Analyse gängiger 3D-Software-Pakete

Im folgenden Kapitel werden einige Beispiele moderner 3D-Animationssoftware analysiert und verglichen. Dabei soll vor allem auf die für den Workflow relevanten und interessanten Aspekte der Software hervorgehoben und ihre Vor- und Nachteile im Arbeiten als Animator innerhalb der Software erfasst werden.

4.1 Autodesk Maya

Eines der größten aktuell erwerblichen Software-Pakete bietet Autodesk Maya [27]. Als Fusion verschiedener Animationsprogramme wurde es bereits Ende der neunziger Jahre von der Firma *Alias Systems* bzw. *Wavefront* begonnen.

Lange Zeit galt Maya neben Softimage [28] und 3DsMax [26] als das am weitesten entwickelte 3D Programm und war deshalb schon in sehr frühen Jahren der Computergrafik weit verbreitet. In enger Kooperation mit Disney wurde für die Produktion des Films *Dinosaurier* [13] intensiv an Mayas Aufbau, den individualisierbaren Schnittstellen und seiner Grundfunktionalität gearbeitet. Durch diese schnelle und sehr gute Entwicklung des Programms wechselte bald ein Großteil der Studios dieser Zeit zu Maya, das damit endgültig den Industriestandard erreichte. Für diese bedeutenden Entwicklungen im Bereich der Computergrafik erhielt Maya Anfang des 21. Jahrhunderts mehrere Academy Awards, unter anderem je einen Award für die Entwicklung der *Subdivision Surfaces* und des *Maya Fluid Effects Systems*. 2004 wurde Maya schließlich von Autodesk aufgekauft und seither stetig weiterentwickelt. Auch viele der großen Konkurrenten der damaligen Zeit – darunter auch Softimage und 3DSMax – wurden nach und nach von Autodesk akquiriert. Während Softimage nach einer erfolglosen Neuauflage des Programms unter dem Namen Softimage XSI 2015 schließlich vollstän-

dig eingestellt wurde, wird 3DSMax bis heute von Autodesk weiterentwickelt und gilt nach wie vor als solides und häufig genutztes Modelling- und Animations-Programm. Aufgrund seiner starken Ähnlichkeit und der parallelen Entwicklung zu Maya soll 3DSMax in dieser Arbeit jedoch keine nähere Analyse gewidmet werden.

Auch heute hat Maya seine Position in der Industrie noch behalten, zahlreiche andere Hersteller machen dem Programm jedoch allmählich in verschiedensten Bereichen Konkurrenz. Aufgrund seiner langen Tradition prägt Maya aber bis heute maßgeblich den Standard-Workflow im 3D Bereich [44].

Als deutlicher Vorteil des Programms ist in jedem Fall der große Umfang zu sehen. Maya ist sowohl mit komplexen Fluid- und Hair-Systemen, als auch mit dem integrierten Renderer *Arnold* ausgestattet. Außerdem besitzt es umfangreiche Poly-Modeling Tools und eignet sich bestens für komplexe Charakteranimationen. Auch die von Anfang an gut integrierten Schnittstellen ermöglichen großen Studios die individuelle Erweiterung des Programms nach eigenen Anforderungen. Dabei werden oftmals große Teile des Programms durch eigene Funktionen ersetzt.

Viele der in Maya verfügbaren Funktionen und Tools wurden extern entwickelt, schließlich aber von Autodesk aufgekauft und als Plug-Ins integriert. Dazu gehören unter anderen das Motion Graphics System *MASH* von Mainframe oder der CPU Renderer *Arnold* von Solid Angle, der als vollwertige Render Engine bis auf wenige Restriktionen im Programmumfang mitgeliefert wird. In vielen Teilen des Programms ist deutlich merkbar, dass unterschiedliche Ansätze für die jeweiligen Werkzeuge gewählt wurden. Neben den verschieden aussehenden Interfaces wie etwas in MASH ist diese Tatsache etwa auch bei der nicht kohärenten Selektionsreihenfolge diverser Werkzeuge erkennbar (vgl. Abb. 4.1). Jedes in Maya verfügbare System funktioniert nur in sich geschlossen problemlos. Eine übergreifende Funktionalität zwischen den jeweiligen Tools besteht allerdings meist nicht. Generative Zugänge – wie etwa in Cinema 4D (Abschnitt 4.2) – sind in Maya nur schwer möglich. Auch die prozeduralen Noise-Texturen sind eher veraltet und schränken hier modernere Möglichkeiten deutlich ein. Außerdem wird in vielen Bereichen deutlich, dass manche Programmteile bereits stark veraltet und überholt sind, aufgrund der nötigen Kompatibilität aber noch im Programm behalten werden. Dies lässt manche Menüs sehr überladen und unübersichtlich wirken.

Der grundsätzliche Aufbau von Maya ist eher strikt und restriktiv. Viele Tools benötigen perfekt valide Geometrie und führen bei fehlerhaftem Einsatz zum Absturz des Programms. Diese strikte Arbeitsweise kann im Vergleich zu anderen neueren 3D-Programmen recht mühsam wirken, hat aber in manchen Bereichen den Vorteil, dass diese strikte und exakte Arbeitsweise im Programm auch sehr genaue Kontrolle über das Endergebnis zulässt. Für große Produktionen, in denen die Konzeption und das Design eines Films ohnehin vorab passieren, kann die strenge und genaue Arbeits-

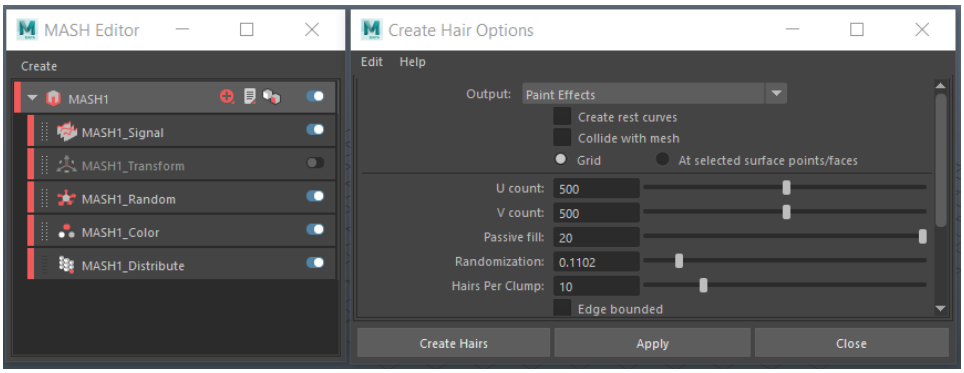


Abbildung 4.1: In dieser Abbildung wird die unterschiedliche Handhabung verschiedener Programmteile deutlich. Während *MASH* im linken Fenster ein völlig eigenes Layout verwendet, sieht man bei Mayas *nHair* ein ansonsten eher typisches Menüfenster.

weise in Maya jedoch von Vorteil sein. Jeder Schritt wird genauestens vom User vorgenommen. Viele Arbeitsschritte funktionieren lediglich destruktiv, was ein Nachträglichen abändern – abgesehen von einem einfachen Rückgängig machen der zuletzt ausgeführten Befehle – unmöglich macht. Obwohl auch einige non-destruktive Ansätze zu finden sind, funktionieren diese nach einigen danach erfolgten Befehlen nur noch selten korrekt. Die Inputs im Channel Editor zeigen verschiedene angewandte Operationen mit ihren zugehörigen Parametern. Das Abändern dieser Werte ist zwar grundsätzlich möglich, führt jedoch oft zu Komplikationen oder invalider Geometrie.

Zusammenfassend kann man Maya nach wie vor als den Standard der dreidimensionalen Animationssoftware sehen. Seine lange Tradition und Prägung dieses Bereichs, aber auch die einfache und gut dokumentierte Schnittstellenanbindung und das individualisierbare Interface bieten sowohl großen, als auch kleineren Studios die nötigen Mittel. Eine breit gefächerte Palette an Tools macht Maya zu einem vielseitigen und soliden Programm. Viele Altlasten und die strikte Funktionsweise vieler Tools machen ein schnelles kreatives Arbeiten mit vielen Iterationen innerhalb kürzester Zeit jedoch schwierig. Hier zeigen viele der in den folgenden Abschnitten genannten Programme deutlich mehr Potential.

4.2 Maxon Cinema 4D

Ein Programm das vor allem im Bereich der Motion Graphics zahlreiche Anwendung findet ist Maxons Cinema 4D [30]. Cinema 4D zeichnet sich durch eine verhältnismäßig einfache Bedienung aus, die dem Interface der Adobe Programme wie etwa Photoshop deutlich ähnelt. Vieles basiert also auf Layern und kann so auch reversiert werden. Crashes der Software sind im

Vergleich zu anderen Programmen eine Seltenheit. Zahlreiche Tools arbeiten non-destruktiv und können beliebig kombiniert werden. Diese eher einfache Bedienung macht Cinema 4D besonders benutzerfreundlich. Die Nähe zu den Adobe Progammen wird auch durch die Anbindung zu Adobe *AfterEffects* [24] deutlich. Dabei wird eine nahtlose Integration von 3D-Assets in AfterEffects angeboten. Weniger umfangreich sind in Cinema 4D hingegen die Funktionen für Charakter-Animationen und insbesondere die verfügbaren Polymodeling-Werkzeuge wie etwa der UV-Editor [43].

Einen großen Fokus legt Cinema 4D außerdem auf Simulationen und prozedurale Generation von Meshes. Hier sind es vor allem die vielen zu Verfügung stehenden Deformer, die gerade im Bereich der Werbung oder Motion Graphics viele visuelle Möglichkeiten und Effekte bieten. Eine breite Palette an generierbaren Noise-Texturen verstärkt die generative Natur des Programms. Funktionen wie die Motors und die soliden und performanten Dynamics im Programm machen die Simulation verschiedenster Vorgänge einfach. Hier lädt das Arbeiten mit Cinema 4D zu einem selektiven künstlerischen Prozess ein, bei dem immer wieder Parameter abgeändert werden können bis das wünschenswerte Ergebnis erreicht wird. Dabei kommt man recht schnell zu einem visuell ansprechenden Ergebnis. Viele Iterationen innerhalb kürzester Zeit sind möglich. Der User gibt einen Teil der visuellen Entstehung jedoch an Cinema 4D ab.

4.2.1 Nikita Diakur

Ein Film bei dem große Teile der Animation eben bewusst der Software überlassen wurden ist *Ugly* [23] von Nikita Diakur. Im Film werden dynamische Tools der Software für die Bewegung der Charaktere verwendet. Nur über wenige Controller wird schließlich auf die Bewegung zugegriffen, der Rest wird von der Software simuliert. Auch die Kamera und verschiedene Spezialeffekte werden auf diese Weise gesteuert. Die Programmoberfläche wird im Film bewusst in Form von Controllern und Interface-Elementen ersichtlich gemacht. Ähnlich wie in *Please Say Something* entsteht so eine recht charakteristische Stilistik, die deutlich von digitalen Artefakten und Merkmalen geprägt ist. Neben den oft sichtbaren Wireframes und Controllern (siehe Abb. 4.2) sind auch bei den Texturen der Wasser- und Feuereffekte die einfache Geometrie auf eine sonst unübliche Art und Weise erkennbar. Durch diese dynamische und generative Arbeitsweise ergibt sich ein völlig anderer Zugang zum Programm und dessen Möglichkeiten [53]. Es werden bewusst Limitierungen innerhalb des Films geschaffen um einen eigenen visuellen Stil zu erzeugen. Nikita Diakur versucht so wenig wie möglich in die eigentliche Animation einzugreifen [37]. Vielmehr verwendet er die integrierten Möglichkeiten von Cinema 4D zur Simulation einfacher Szenen. Herkömmliche Keyframe-Animation findet hier nur selten Platz. Es werden Rahmenbedingungen festgelegt, in denen der Simulation schließlich freien

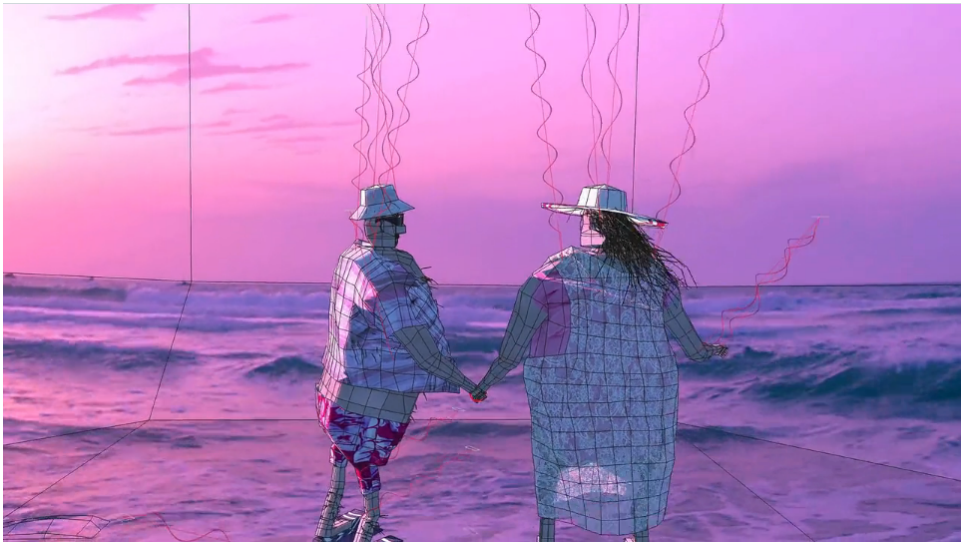


Abbildung 4.2: Ein Ident von Nikita Diakur im Stil seines erfolgreichen Kurzfilms *Ugly* [23] für den TV-Seder MTV.

Lauf gelassen werden kann. Dieser Szenenaufbau wird von Nikita Diakur solange angepasst, bis ein interessanter Output entsteht. Verschiedene Parameter beeinflussen die Animation, anstatt exakt von Keyframes definieren. Es bleibt ein großer Spielraum für die simulierten Objekte, die sich nicht immer völlig nach Plan bewegen. Den Prozess dokumentiert er dabei häufig auch auf Social-Media Plattformen und postet zahlreiche Outtakes und fehlgeschlagene Simulationen, die seinen typischen Animationsstil deutlich zeigen.

4.2.2 Zeitguised und f°am Studio

Das Kollektiv Zeitguised und das zugehörige f°am Studio arbeiten ebenfalls mit Maxon Cinema 4D. Trotz des sich stark unterscheidenen Stils verfolgen sie ein ähnliches Prinzip. In ihrem aktuellsten Film *Influencers* [15] wird unbelebten Objekten durch die prozedurale Arbeitsweise mit Deformern und anderen Effekten Leben eingehaucht. Auch hier wird vieles generiert, und somit zu einem hohen Grad der Software überlassen. Durch den modularen Aufbau des Programms entstehen viele der Formen neben der eigentlichen Konzeption hier durch Experimente. Gleichzeitig arbeitet Zeitguised in enger Kooperation mit Maxon, die sie auch für die Produktion von *Influencers* beauftragt haben. Der Film entstand mit der Beta Version des Release 20 von Cinema 4D. Durch solche Kooperationen können Softwareentwickler zeitgerecht Feedback für entwickelte Tools und gewünschte fehlende Funktionen im Programm einholen.

Abbildung 4.3: Der Kurzfilm *Spring* [20] wurde zur Gänze mit dem Open-Source Programm Blender produziert.

4.3 Houdini

Als absoluter Vorreiter im Bereich der Visual Effects und des prozeduralen Arbeitens in 3D gewann SideFX Houdini [32] im Laufe der letzten Jahre immer mehr an Beliebtheit.

Als deutlichen Unterschied zu den meisten anderen hier genannten Programmen arbeitet Houdini – wie auch das Compositing Programm Nuke [33] von The Foundry – fast ausschließlich mit Nodes (vgl. Abschnitt 3.5). Die Node-basierte Arbeitsweise bringt viele Vorteile und wurde deshalb auch von vielen anderen Programmen bereits teilweise aufgegriffen. Node-basiertes Arbeiten hat beispielsweise den großen Vorteil, dass ein Großteil der Arbeitsschritte in Form der Nodes erhalten bleibt und im Nachhinein verändert werden kann, ohne spätere Arbeitsschritte wiederholen zu müssen. Im Gegensatz dazu arbeiten beispielsweise viele der Operatoren und Funktionen in Maya destruktiv.

Da in den meisten Film-Produktionen im Laufe der Zeit Änderungen anfallen, erspart non-destruktives Arbeitenviele wiederholte Arbeitsschritte und fördert ein Experimentieren mit Parametern der einzelnen Nodes.

Gleichzeitig ist die Lernkurve in Houdini sehr steil, auch für User anderer 3D Programme. Viele Konzepte sind exklusiv in Houdini und müssen erst erlernt werden. Für ein schnelles Arbeiten im Programm müssen außerdem zuerst zahlreiche Nodes verstanden und erlernt werden. Dabei sind nur wenige von ihnen selbsterklärend. Dafür bietet Houdini ein außerordentlich umfangreiches Toolset für procedurales Modelling, Animation, Texturing, Lighting, Simulation und mit Katana auch einen Renderer, der deutlich mit anderen geläufigen Render Engines mithalten kann.

4.4 Blender

Als am häufigsten verwendete Open-Source 3D-Software stellt Blender [29] eine gute alternative zu anderen Programmen dar. Das frei verfügbare Programm wird beierts seit 1994 entwickelt und häuft seither eine immer größer werdende Community an. Genau das ist auch der große Vorteil des Programms. Auch wenn das Entwicklerteam hinter Blender deutlich kleiner ist und somit nicht alle Tools dem aktuellen Standard entsprechen entwickelt die Community eine vielzahl an individuellen Werkzeugen die den Foren kostenlos zugänglich sind. Dadurch entsteht nicht nur ein Mehrwert im Funktionsumfang des Programms. Auch die Dokumentation des Programms und der zahlreichen Funktionen ist somit basierend auf der Arbeit der Community einfach verfügbar [42]. Als Nachteil ist bei Blender noch der etwas anders umgesetzte Interface- und Programmaufbau. Viele Menüs wirken zu Beginn schwer zugänglich und auch die Selektion im Viewport ist deutlich anders gelöst als es viele User vermutlich gewohnt sind. Alles in allem bietet Blender jedoch als frei verfügbares Programm also alle nötigen Werkzeuge, ist auf allen Plattformen verfügbar, leicht erweiterbar – sowohl selbst als auch durch die große Bibliothek an user-generiertem Content – und entwickelt sich mit beachtlichem Tempo voran.

Als aktuellstes Projekt des Open-Source Programms zeigt der Film Spring [20] (Abb. 4.3) die Relevanz des Programms im Animationsbereich. Da auch der Film auf dem Open-Source Prinzip basiert, wurden sowohl die Blender-Files, als auch eine Reihe an umfangreichen Making-Of Videos und Posts veröffentlicht. Auch hier beruft sich Blender deutlich auf sein Community Prinzip. Neue Funktionen im aktuellen Release können durch das zur Verfügung stehende Projekt analysiert und dann selbst in Form eigener Projekte umgesetzt werden.

4.5 Unity

Ein neuerlicher Trend im Animationsbereich ist das rendern in Echtzeit – oder auch Realtime Rendering [55]. Dabei werden die Einzelframes einer Animation so schnell generiert, dass der Film eben in Echtzeit abgespielt werden kann. Besonders nützlich ist das im Designprozess eines Animationsfilms, da sämtliche Renderzeiten wegfallen und die Auswirkung jeder Änderung fast imminent sichtbar wird. Neben Unreal Engine [35] ist es vor allem die Game-Engine *Unity* [34], die in den letzten Jahren einen großen Fortschritt in diesem Feld verzeichnen konnte.

Dabei ergeben sich in der Pipeline einer solchen Engine zahlreiche interessante Aspekte und auch viele Vorteile. Einerseits fallen viele Arbeitsschritte zumindest teilweise weg. Der Renderprozess wird beispielsweise minimiert. Der Benutzer erhält sofortiges Feedback auf Änderungen in der

(a) (b)

Abbildung 4.4: Während das Projekt *Adam* [9] in Abb. (a) die Renderpower von Unity demonstriert, zielt der Film *Sonder* [19] in Abb. (b) auf eine viel stärkere Stilisierung ab.

Szene, anstatt ein langwieriges Rendering abwarten zu müssen. So kann der User weitaus freier und intuitiver in der Software arbeiten und Parameter einfach auf gewünschte Weise anpassen. Der kreative Prozess wird von der Software weitaus weniger unterbrochen.

Aber auch Nachteile bringt das Rendern in Echtzeit mit sich. Einerseits ist die dafür nötige Software noch nicht ganz auf dem nötigen Stand, den aktuelle Animationssoftware mit sich bringen sollte. Der Background in der Spieleentwicklung wird hier in Unity und Unreal Engine recht schnell deutlich. Auch wenn beide Programme laufend neue Features für das Erstellen von Animation in der Engine entwickeln, gibt es noch zu viele Hürden um einen reibungslosen Workflow zu ermöglichen. Ein weiterer Nachteil ist, dass die Performance in der Engine hoch gehalten werden muss. Somit können weder aufwändige Texturen oder Simulationen, noch hochaufgelöste Models oder komplizierte Rigs verwendet werden um eine flüssige Darstellung zu ermöglichen. Es entsteht ein enormer technischer Aufwand, wodurch der kreative Anteil einer Produktion deutlich in den Hintergrund geraten kann. Obwohl diese Einschränkung keinen visuellen Nachteil bedeuten muss, – aktuelle Projekte in Unity zeigen deutlich, dass auch in Realtime eine beinahe real wirkende Grafik erzielt werden kann – entsteht hier ein immenser Aufwand, um jeden Vorgang in der Animation performant halten zu können. Auch der Schritt einer Postproduktion, der in vielen Filmen einen sehr wichtigen Arbeitsschritt für den finalen Look darstellt, fällt beim Realtime-Rendering weg oder muss in die Render-Pipeline eingebaut werden, um von einem wirklich in Echtzeit abspielbaren finalen Ergebnis sprechen zu können.

Als einer der ersten in Echtzeit gerenderten Filmprojekte zeigte der Film

Adam [9] (Abb. 4.4 a) im Jahr 2016 erstmals die verfügbaren Möglichkeiten von Unity auf. Sowohl die hoch-qualitative Grafik, als auch die beeindruckenden Rendereffekte rückten Realtime-Rendering erstmals ins Licht der Animationsfilms. Diesen Look zu erreichen bedeutet jedoch einen immensen Aufwand. Geometrie, Texturen, Simulationen und Effekte – alles muss gut überlegt in die Engine eingebaut werden, um die gwünscht Performace zu erreichen. Ist dieser Aufbau aber einmal erreich, sind an vielen Stellen Änderungen im Shader oder im Licht-Setup jederzeit probelmlos möglich. An anderen Stellen sind aber große Caches und vorgerenderte Elemente zu finden, die den eigentlichen Vorteil der großen Flexibilität und der starken Geschwindigkeit eines Realtime-Renderings zunichte machen. Hier fällt der Workflow zurück in die Bahnen einer herkömmlichen Produktion.

Auch der Film *Sonder* [19] (Abb. 4.4 b) verwendet Unity als Animations-Software, geht jedoch stilistisch eine gänzlich andere Richtung. Sonder zeichnet sich durch einen malerischen, comichaften und stark zweidimensional wirkenden Look aus. Zum Feld des Realtime-Renderings zählt dieser Film genaugenommen nicht, da hier die Einzelframes aus Unity – wie auch in herkömmlichen 3D-Programmen üblich – exportiert wurden. Ein Abspielen des Films direkt in Unity ist also nicht möglich. Einige Vorteile bietet das auf Realtime ausgerichtete Arbeiten, beziehungsweise das Arbeiten in einer Game-Engine dennoch.

4.6 Zusammenfassung

Vergleicht man schlussendlich die verschiedenen verfügbaren Tools, wird deutlich, dass sich ihr Funktionumfang nur leicht unterscheidet. Grundfunktionen zum Modelling, Rigging, für Animation und Simulation sind in jedem der beschriebenen Pakete ausreichend vorhanden. Auch wenn die Qualität der Tools variiert und einige Programme deutliche Schwerpunkte in ihrer Tool-Palette aufweisen, kann man als User mit jeder Software zum Ziel kommen. Unterschiede finden sich hingegen in der Anwendung dieser Funktionen. Der Aufbau der Pakete und somit der grundlegende Workflow im Programm unterscheidet sich teils deutlich. Während Cinema 4D vergleichsweise einfach zu verwenden ist, besitzt Houdini mit seiner Node-Architektur ein für Animatoren sehr mächtiges Repertoire, das jedoch auch eine durchaus große Einstiegshürde darstellt. Der User muss die einzelnen Nodes kennen und verstehen, um eine sinnvolle Anwendung zu gewährleisten. Ist ein Node-Netzwerk jedoch einmal erstellt, können einzelne Parameter des Aufbaus beliebig verändert und angepasst werden um zum finalen Bild zu kommen. Dies unterscheidet Houdini stark von der destruktiven Arbeitsweise Mayas. Hier findet man statt einem modularen Aufbau, eine große Menge an eigenständigen Tools, von denen jedes einer sehr bestimmten Funktion gewidmet ist. Diese Funktion ist bei vielen Tools klar ersichtlich. Um die zahlreichen

Parameter und die In- und Output-Werte, die dabei im Hintergrund vom Programm verarbeitet werden, muss sich der User in diesem Fall nicht unbedingt kümmern – die Möglichkeit dazu besteht aber mit dem Node-Editor dennoch. Auch wenn dieser eine weitaus geringere Funktionalität aufweist, als etwa in Houdini.

Ein weiterer wichtiger Punkt, an dem sich Software stark unterschieden kann sind die jeweiligen Schnittstellen. Zwar weisen alle Programme eine Unterstützung für Plug-Ins und Scripts auf, mit denen der Funktionsumfang erweitert werden kann, die Open-Source Arbeitsweise von Blender hat hier jedoch deutliche Vorteile. Da hier jeder Programmteil bis zum elementarsten Source-Code ersichtlich ist, wird der Community von Blender das erstellen hilfreicher Zusatzfunktionen und somit eine Erweiterung des Programms stark vereinfacht. Im Gegensatz dazu sind Plug-Ins für Maya und Cinema 4D oft kostenpflichtig und somit nicht problemlos für jeden User zugänglich.

Schließlich unterscheiden sich die Programme auch in der Art und Weise, wie sie dem User beim Designprozess unterstützen. Programme wie Cinema 4D zeichnen sich durch ihre einfache Handhabung als besonders benutzerfreundlich aus. Dadurch wird ein Experimentieren im Programm gefördert. Oft ergeben sich so Designentscheidungen mitten im Arbeitsprozess, an anderen Stellen wird wie bei Nikita Diakur die Animation auf spielerische Art und Weise über Simulationen dem Programm überlassen.

Der Workflow wird vor allem bei komplexeren Szenen oder aufwändigen Lichtsituationen und Shadern oftmals durch lange Rechen- und Renderzeiten erschwert. Je schneller der User Feedback vom Programm bekommt, umso einfacher kann er darauf reagieren und die Szene anpassen. Dauert ein Arbeitsschritt hingegen lange oder führt zu mehreren Fehlern bis ein gewünschtes Resultat erreicht wird, wird der Designprozess stark beeinflusst. Der User kann hier seine Vorstellung und Idee oft nicht vollständig oder nur mit großen Zeitaufwand umsetzen. An dieser Stelle bieten aktuelle Trends im Echtzeit-Rendering einen interessanten Zugang. Noch sind die verfügbaren Programme jedoch nicht vollständig auf die Produktion eines Animationsfilms auf diese Art und Weise ausgelegt. Vieles muss angepasst werden und der große Fokus auf die nötige Performance kann eine große Einschränkung bedeuten. Abgesehn davon wäre ein Echtzeit-Feedback für viele Arbeitsschritte jedoch eine große Erleichterung. Gerade bei Lighting und Shading digitaler Szenen ist nur so ein einfacher und schneller Workflow mit vielen Iterationen innerhalb kürzester Zeit möglich. Auch wenn moderne Grafikkarten-Renderer hier einen deutlichen Fortschritt aufweisen können, sind sie von einem Feedback in Echtzeit teils noch weit entfernt. Komplexe Geometrien und Simulationen werden aber auch in den nächsten Jahren nicht in Echtzeit möglich sein.

Kapitel 5

Weitere relevante Technologien

Im Folgenden sollen Technologien behandelt werden, die nicht unter den Begriff der 3D-Software im herkömmlichen Sinn fallen. Viele dieser Tools werden jedoch in Kombination mit 3D-Programmen verwendet und zeigen interessante Zugänge und Workflows auf, deren Vorteile und Implikationen, aber auch Nachteile in diesem Kapitel untersucht werden sollen.

5.1 Generative Tools: Processing

Als generative Tools lassen sich Werkzeuge zusammenfassen, durch die mittels Computercode Geometrie und Formen dynamisch erzeugt werden können. Diesen Ansatz bieten Programme wie Houdini an (vgl. Abschnitt 4.3) und Maya an. Häufig wird dabei als Programmiersprache Python verwendet, Maya bietet zusätzlich auch MEL als vereinfachte Scriptsprache an. Während der generative Ansatz in Houdini und Maya sich häufig auf das Erstellen von Objekten beschränkt, wird in anderen Bereichen gänzlich generativ gearbeitet. Dazu zählt vor allem *Processing* [51], das im Folgenden genauer beschrieben werden soll.

Processing ist eine Programmiersprache, die im Grunde ausschließlich für grafische Anwendungen zum Einsatz kommt. Zu Processing zählt außerdem eine Open-Source Grafik Bibliothek und eine integrierte Entwicklungsumgebung. Da Processing nicht wirklich unter den Begriff der Software fällt, wird sämtlicher Output hier zur Laufzeit mit Code generiert und verändert. Dies hat den Vorteil, dass recht einfach mit verschiedenen Schnittstellen auf User-Input oder auch Sound reagiert werden kann. Aus diesem Grund kommt Processing auch sehr oft für Installationen zum Einsatz. Im Gegensatz zu einer 3D-Software ist hier allerdings ein weitaus technischerer Ansatz nötig. Dennoch kann man durch viele online verfügbaren Bibliotheken schnell zu einem Ergebnis kommen. Durch den Open-Source Ansatz sind auch hier –

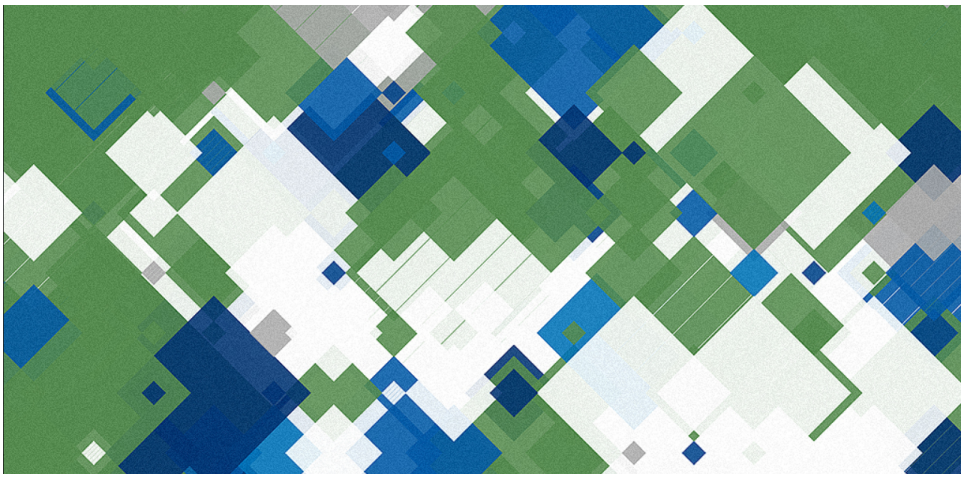


Abbildung 5.1: Mit Processing generierter Stillframe aus dem Projekt *Tennis Deconstructed* [21] für Nike von Joshua Davis.

ähnlich zu den Sourcefiles bei Unity und Blender – zahlreiche in Processing umgesetzte Projekte frei online verfügbar.[1]

Da sowohl die digitale Arbeitsfläche, – in diesem Zusammenhang meist als Canvas bezeichnet – als auch die Werkzeuge und Objekte alle von Grund auf vom Artist erstellt werden können, unterscheidet der Generative Ansatz deutlich vom Arbeiten mit vorgefertigten Software-Tools. Dem User bleibt völlig Kontrolle über seine Arbeit, gleichzeitig muss eben auch alles vom User selbst erstellt werden. Nur wenige Grundfunktionen werden von Processing selbst zur Verfügung gestellt. Die Komplexität von Werkzeugen in einer Animationssoftware wird aber nicht erreicht. Hier kommt jedoch die Open-Source Methode von Processing und die dazugehörigen Community- und Nutzer-basierten Bibliotheken zum Einsatz. Software-Artist Casey Reas, der gemeinsam mit Ben Fry maßgeblich für die Entwicklung von Processing verantwortlich war, stellt in einem Essay folgendes fest [52]:

> Through the tools that I make, modify, and use, I sometimes feel like I am in control and sometimes I feel like my ideas are heavily biased by software that others have made.

Der Artist erstellt hier also selbstständig seine benötigten Tools, bestimmt ihren Umfang und ihre Funktion und teilt sie schließlich häufig auch mit anderen Nutzern. Erstellte Tools können schließlich in verschiedenen Variationen oder in Kombination mit anderen Bausteinen wiederverwendet werden, um eine große Reihe an Output in sehr kurzer Zeit zu generieren. Diese Art der Anwendung von Software-generierter Kunst macht sie so wandelbar und für verschiedenste Zwecke einfach anpassbar. So kann auf bereits

[1] Besonders nennenswert ist hier die Webseite https://www.openprocessing.org/.

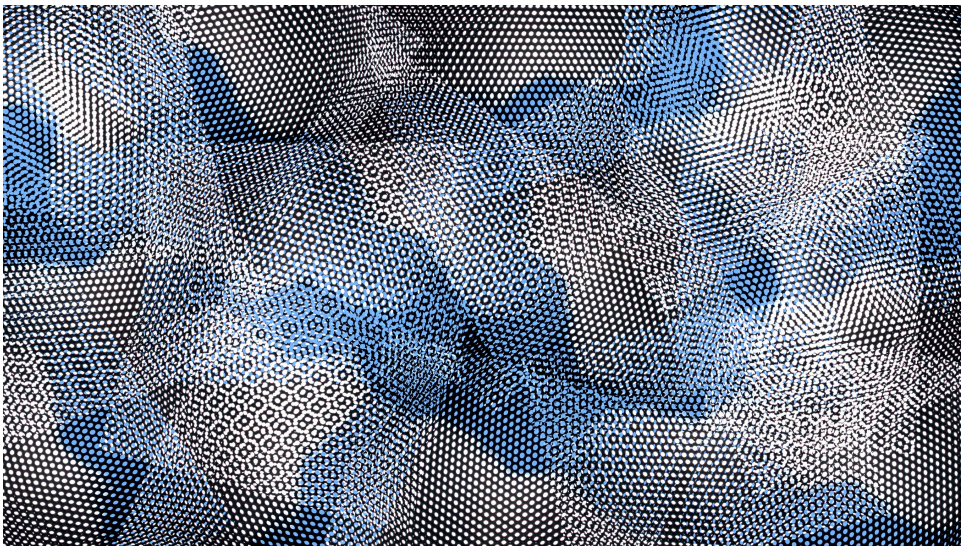


Abbildung 5.2: Ein Bild aus Casey Reas Chronograph Variations [10] für die Fassade des New World Center in Miami.

geschriebenem Code aufgebaut werden. Der Funktionsumfang von Processing ist dadurch vollkommen wandelbar und von der Community bestimmt. Von dieser werden Trends gesetzt, verwirklicht und geteilt, um wiederum andere User zu inspirieren. Processing definiert sich also als digitales Medium auch über seine Verwender und deren Beiträge zu seinen Tools. Obwohl Blender auf einem ähnlichen System aufbaut, ist dieser fast vollkommen modulare Aufbau mit keinem aktuell erhältlichen Animationsprogramm vergleichbar.

Durch das dynamische Erstellen von Objekten unterscheiden sich Arbeiten, die auf diese Weise entstanden sind, stark von anderen Filmen. Es wird nur selten linear gearbeitet. Während eine Animation mit Keyframes genau geplant werden kann, häufig einen exakten Ablauf erfolgt und dabei oft eine von Charakteren vorangetriebene Geschichte erzählt, beschränkt sich Processing vielmehr auf den Bereich der Motion Graphics. Hier sind es nur einzelne Parameter für die Erzeugung des Bildes, die sich verändern und somit eine Dynamik, Bewegung und Veränderung im Bild erzeugen. Das generative Arbeiten ist vielmehr von einem stetigen visuellen Wandel geprägt. Eine Erzählung findet hier auf viel abstraktere Weise statt. Der Zugang zum entstehenden Bild erfolgt lediglich über den Computercode und ist daher ohne ein mit dem Viewport moderner 3D Software vergleichbares Interface deutlich stärker abstrahiert.

5.2 Künstliche Intelligenz

Einen besonderen, aktuellen Trend bilden Systeme Künstlicher Intelligenz[2], die ebenfalls zur generativen Kunst gezählt werden können. Für die Definition einer Künstlichen Intelligenz gibt es verschiedene Ansätze. Gemeinsame Eigenschaft bleibt dabei aber immer das Entwickeln eines Frameworks aus Computercode, das möglichst effizient komplexe Sachverhalte bearbeiten kann. Die Aufgaben für AI-System reichen dabei von sehr spezifischen Tätigkeiten wie dem Spielen von Schach oder dem chinesischen Spiel Go bis hin zu breiteren Anwendungen wie ChatBots, die auf jeden möglichen Input reagieren können, ohne diesen im Vorhinein zu kennen. Andere Systeme wie die OpenAI [47] zielen wiederum auf eine generell einsetzbare Künstliche Intelligenz ab, die verschiedenste Aufgaben bewältigen könnte. Hier ist eine Unterscheidung in drei Kategorien möglich [7, S. 1020–1040]. Als *schwache* Künstliche Intelligenz (engl. ANI = artificial narrow intelligence) bezeichnet man Systeme, die für eine ganz bestimmte Aufgabe erstellt wurden und diese auch erfüllen. Beispiele sind die oben genannten Schachcomputer, aber auch Bremsassistenten in modernen Autos. Sie erfüllen ihre Aufgabe zwar, können aber noch nicht als wirkliche, eigenständige Intelligenz bezeichnet werden. Die zweite Kategorie, zu der ein Großteil der im Folgenden genannten Systeme gezählt werden kann, sind *starke* Künstliche Intelligenzen (engl. AGI = artificial general intelligence). Sie unterscheiden sich vor allem durch ihr eigenständigeres „Denken“ und das dadurch breitere Einsatzgebiet. Hier kann ein Vergleich zur menschlichen Intelligenz durchaus gegeben sein. Starke KIs erfassen komplexe Aufgabengebiete oder lernen aus ihren Erfahrungen. Die letzte Kategorie sind sogenannte Künstliche *Superintelligenzen* (engl. ASI = artificial super intelligence). Diese Spitze der KI Forschung würde sich durch ein völlig selbstständiges und unabhängiges Denken und Handeln auszeichnen. Durch diese starke Unabhängigkeit würde eine Superintelligenz den Menschen in allen Bereichen weit übertreffen. Mit dem heutigen Stand der Technik sind diese Systeme jedoch noch in einiger Entfernung.

Im Zusammenhang mit KI spielt auch der Begriff „Machine Learning“ eine wichtige Rolle. Machine Learning beschreibt Systeme, die aus gesammelten Daten lernen und sich dadurch selbst verbessern können. Beispiele für Machine Learning und KI-Systeme sind *Google's Deepdream* [40] oder *OpenAI* [47]. Eine spezielle Version der Künstlichen Intelligenz sind sogenannte GAN-Systeme[3]. Diese sind besonders im künstlerischen Einsatz häufig in Gebrauch. Bei einem GAN-System kommunizieren zwei unabhängige KIs miteinander. Dabei erzeugt der *Generator* in ständiger Wiederholung Bilder, während der *Diskriminator* diese evaluiert und Unterschiede zu den

[2] Der Begriff wird dabei abgekürzt als KI, beziehungsweise AI vom englischen Begriff „artificial intelligence“.

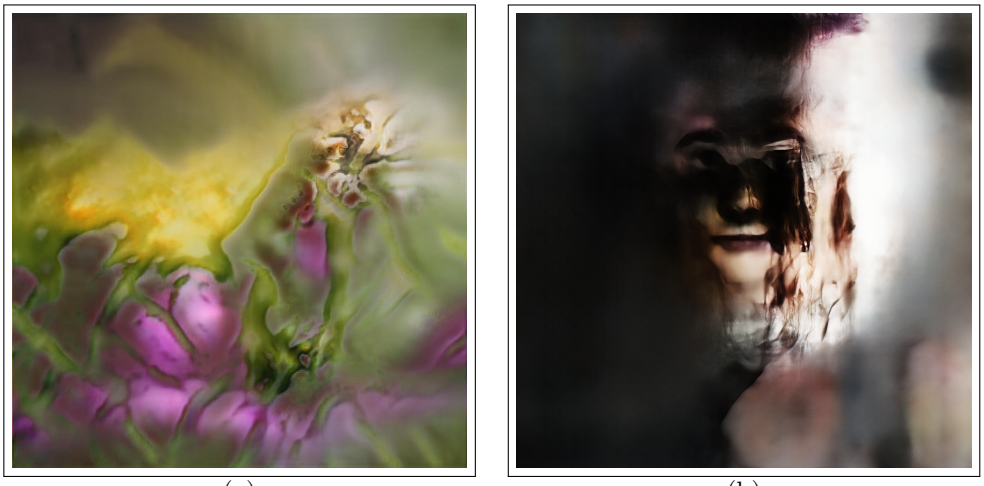
[3] GAN steht dabei für den englischen Begriff „general adverserial network“.

zur Verfügung stehenden Bilddaten feststellt [39]. Nach längerer Laufzeit werden vom Generator somit immer bessere, beziehungsweise den Grunddaten möglichst ähnliche, aber dennoch unabhängig generierte Bilder erstellt. Das System entwickelt sich laufend weiter. Der Aufbau dieser GAN-Systeme ähnelt dabei stark dem iterativen Prozess, der auch in einer herkömmlichen Produktion eines Animationsfilms üblich ist. Feedback wird laufend ins Bild eingearbeitet bis ein gewünschtes Ergebnis erreicht wird. Im Unterschied zum menschlichen Animator hat die KI jedoch keinerlei Grundwissen und Verständnis dessen, was sie zeichnet oder generiert. Vielmehr lernt die KI durch die zahllosen Iterationen stetig dazu. Das erlernte wird anschließend auch gespeichert. Hat eine Künstliche Intelligenz einmal einen wünschenswerten Output erreicht, kann dieser beliebig in einer anderen Form neu generiert werden. Bedeutende artistische Entscheidungen werden also der KI überlassen. Künstliche Intelligenzen zeichnen sich dadurch aus, dass sie eigenständig mit den ihnen zur Verfügung stehenden Daten Entscheidungen treffen können.

In der Kunst wird Machine Learning also oft zur Generierung von Bildern verwendet, die auf der Ästethik und dem Inhalt anderer Werke basieren. Diese Werke werden in großen Mengen von der KI analysiert, wiederholte Strukturen und Prinzipien werden erkannt. Mit diesem Wissen kann KI schießlich ein eigenständiges Bild erschaffen, das zwar auf dem Wissen der zugrunde liegenden Werke basiert, nicht jedoch aus ihrem Bildinhalten konkret zusammengestellt wurde. Die Werke, auf denen das generierte Bild basiert, sind in diesem nicht mehr erkennbar. Gegenüber herkömmlichen prozedural beziehungsweise von Software generierten Bildern sind bei einer KI jedoch einige interessante Aspekte zu analysieren. Als Künstliche Intelligenz wächst Software über ihren Status als Tool und Medium allein hinaus. Vielmehr wird sie selbst zum Rezipienten und gleichzeitig auch zum Autor.

> „Infact, AI acts as an art theorist, an art historian or a film scholar who is also repeatedly studies many works in some area of culture to find their common patterns.“ [45]

Als Medium, könnte man also argumentieren, ist KI gänzlich von anderer Software abzugrenzen. Welche Rolle der KI konkret zuzuschreiben ist, kann allgemein nicht ohne weiteres definiert werden. Der Künstler übernimmt jedoch hier die Rolle des Programmierers. Sein Einfluss beschränkt sich – im Gegensatz zu beispielsweise prozeduraler Kunst, bei der ebenfalls mit Code gearbeitet wird – auf die Parameter mit denen die KI arbeitet und die Daten aus denen sie lernt. Das schlussendliche Ergebnis ist – egal ob in bildlicher oder einem anderen Form – nur schwer absehbar. Die Bedeutung dieser Technologie zeigt sich jedoch aktuell sowohl in der modernen Kunstszene als auch in vielen anderen technischen Bereichen deutlich. Aber auch andere Anwendungsmöglichkeiten von KI übersteigen die Rolle der eigentlichen Software.



(a) (b)

Abbildung 5.3: Obwohl beide Projekte einen deutlich unterschiedlichen Bildinhalten und ein gänzliche anderes Thema behandeln ist dennoch eine gemeinsame Bildsprache und Textur bemerkbar: Abb. (a) zeigt ein Bild aus den *Deep Meditations* [12] von Memo Akten , Abb. (b) ein Bild der Neural Glitches aus dem Film *Mistaken Identities* [17] von Mario Klingemann.

So kann KI mittlerweile die Qualität von Bildern (und damit sozusagen von visueller Kunst im Allgemeinen) nach eigenem Ermessen bewerten und in vielen Fällen auch verbessern [45].

5.2.1 Memo Akten

Als Pionier im Bereich der modernen computergenerierten Kunst arbeitet Memo Akten schon seit einigen Jahren mit Machine Learning Systemen. Besonderes Merkmal legt er dabei auf die Schnittstelle von Wissenschaft und Spiritualität und setzt sich selbst das Ziel, kreative Kollaboration zwischen Mensch und Maschine zu ermöglichen [12]. In vielen seiner Arbeiten – beispielsweise den Deep Meditations (Abb. 5.3a) – werden einer KI große Mengen an Bilddaten gezeigt. Schließlich werden neue Bilder generiert, die Teile des Gelernten widerspiegeln. Es entsteht eine Art Traum eines Computers, der stark vom zuvor Gesehenen abhängt, gleichzeitig aber etwas völlig Neuartiges zeigt. In den Deep Meditations lernt die KI anhand der Ergebnisse einer Bildersuche zum Begriff „everything“. Zwar ist dem neuralen Netzwerk nun das Aussehen von zahlreichen Objekten bekannt, ihre Semantik fehlt jedoch gänzlich. Im resultierenden Film morphen deshalb Blumen zu ganze Galaxien und Sterne werden zu mikroskopischen Bakterien. Es ergibt sich ein ständiger Bildwandel, in dem immer wieder bekannte Formen und Strukturen auftauchen. Auch der Sound zum Video stammt aus diesem neuralen Netzwerk. Hier wurden religiöse Gesänge als Grundlage verwendet, die im

(a) (b)

Abbildung 5.4: Zwei unterschiedliche Versionen eines mit KI veränderten Videos: Abb. (a) aus dem Jahr 2016 imitiert den malerischen Stil Claude Monets während Abb. (b) ein Bild im Stil einer Felsenküste zeigt [14].

finalen Soundtrack ineinander verschmelzen.

Auf dem selben Prinzip basierend können in Memo Aktens *Learning to See* [16] interaktiv Gegenstände unter einer Kamera platziert werden, während die Künstliche Intelligenz versucht, das dadurch aufgenommene Video live in ein gelerntes Muster zu verwandeln. So entstehen aus einem Stück Stoff und einigen Kabeln beispielsweise Flammen, Wellen oder Blüten, die von der KI zuvor erlernt wurden.

Eine weitere Arbeit von Memo Akten, die den großen Wandel seit dem Aufkommen Künstlicher Intelligenz als großes Thema in Kunst und Computerwissenschaft zeigt, ist *#EpicGanGuy2019* [14]. Beinahe dasselbe Projekt veröffentlichte Akten schon 2016 unter dem Titel *#EpicMonetGuy2016* mit den damaligen technischen Standards Künstlicher Intelligenz (vgl. Abb 5.4). Ein populäres Video des Eurovision Songcontests wird hier von zwei unterschiedlichen KIs verfremdet. In der ursprünglichen Version arbeitet Memo Akten mit einem „Style-Transfer“ Algorithmus, der den Stil eines Bildes lernen und auf andere Bilder anwenden und möglichst exakt imitieren kann. In diesem Fall stellt das Bild *Poppyfield in Giverny* von Claude Monet die stilistische Grundlage dar. Eine viel deutlichere und modernere Umsetzung schafft hingegen die neuere Version aus dem Jahr 2019. Hier wird das originale Video von Nvidias *Gaugan* [31] in eine Felsenküste verwandelt. Es entsteht hier ein viel klareres und realeres Bild mit weitaus weniger sichtbarem Rauschen und nur wenigen Artefakten.

5.2.2 Mario Klingemann

Ein weiterer Künstler, dessen Fokus in den letzten Jahren deutlich auf Künstlicher Intelligenz lag, ist der deutsche Medienkünstler Mario Klingemann. Mit seinen *Neural Glitches* (Abb. 5.3b) beschäftigt sich Klingemann mit bewusst hervorgerufenen Fehlern in GAN-Systemen. Er trainiert die Systeme erst, löscht später aber einzelne Teile des Gelernten. So entsteht eine Lücke, die die KI füllen muss. Es entstehen teils gespenstische Bilder, bei denen Teile von Gesichtern schmelzen oder ineinander fließen. Die Neural Glitches sind für ihn dabei eine Technik, mit der er eine Reihe an Werken entwickelt hat. Das bekannteste ist dabei die Filmreihe *Mistaken Identities* [17]. Hier setzen sich in fast zwei Stunden langen Filmen drei Gesichter immer wieder neu zusammen, wandeln sich, ändern ihre Struktur und ihr Aussehen und verschwinden dann allmählich wieder. Auf die Frage nach seiner Motivation, Kunst mit einer KI zu generieren antwortet Klingemann in einem Interview:

> „I want to see interesting images, different images, and I want to understand this meaningful idea of what information aesthetics are. [...] It's not just about the pixels themselves, it's about what they mean and the story they tell." [50]

Auch hier ist der Ansatz einer kreativen Leistung durch die KI erkennbar. Das neurale Netzwerk versucht, für sie unverständliche Formen zu vervollständigen. Durch diese Glitches entstehen Klingemanns Bilder. Neben den Experimenten mit Bildmanipulation durch KI arbeitet Klingemann auch an Sortieralgorithmen, die Bilder anhand ihrer visuellen Merkmale kategorisieren sollen. Teil dieser Forschung ist das Projekt X Degrees of Separation, das Klingemann 2016 für Google entwickelt hat. Die Applikation ermöglicht es, für zwei gewählte Objekte der Kunstgeschichte eine Interpolation zu generieren aus anderen Werken, die den beiden gewählten Werken visuell ähneln. Es entsteht eine Reihe an realen Bildern, die von einem zum nächsten zu morphen scheinen, wobei sowohl Farbe, als auch Form der Bilder berücksichtigt wird. Die Bilder an sich werden dabei nicht verändert, der visuelle Verlauf ergibt sich durch das aneinanderreihen.

5.3 Microsoft Kinect

Eine Technologie, die erstmals eine Abbildung dreidimensionaler Bilddaten zuließ, war die 2010 veröffentlichte *Kinect* [46] von Microsoft. Durch mehrere Sensoren – unter anderem einen Tiefen- und Infrarotsensor, aber auch eine herkömmliche Videokamera – gelingt es der zugehörigen Software, eine für die Kinect typische Punktewolke zu generieren [8] (vgl. Abb 5.5). Ursprünglich zur Gesten-Steuerung der Spielekonsole Xbox 360 gedacht, begannen schon nach kurzer Zeit erste künstlerische Experimente [41]. Gerade

Abbildung 5.5: Der Film *Clouds* von James George [11] verwendet erstmals die Kinect, um dreidimensionale Bilder aus einer Kombination von Sensor- und Videodaten zu erzeugen.

weil diese Art des dreidimensionalen Scannens einer realen Umgebung zuvor nicht oder nur schwer umsetzbar war, machte die Kinect diese Technologie für die breite Masse nutzbar. So gelang es dem Videokünstler James George mit seinem Film *Clouds* (vgl. Abb. 5.5) die Punktewolke der Kinect mit den Videodaten einer DSLR Kamera zu kombinieren. Diese Technik nennt er RGB+D. Es entsteht eine dreidimensionale Aufnahme einer Szene, bei der der verwendete technische Zugang klar ersichtlich bleibt. Besonders deutlich werden anhand der Kinect also auch die großen stilistischen Implikationen bei der Verwendung eines neuartigen Mediums. Der Stil, den die Datensätze eines Kinect Videos mit sich bringen, war von Anfang an charakteristisch für das Medium. Da die Hardware eine Wolke an im Raum platzierten Punkten ausliest und an die zugehörige Software weitergibt, ist diese Punktewolke auch der grundlegend charakteristische Stil, der die Kinect auszeichnet.

Einen ähnlichen Ansatz ähnlich zu dem der Kinect zeigt der 2018 mit einer Goldenen Nica ausgezeichnete Film *TROPICS* [22] von Mathilde Lavenne. Der Film spielt auf einer Farm in Mexico und ruft mit Hilfe verschiedener akustischer Sequenzen Erinnerungen an diesen Ort hervor. Die Künstlerin verwendet dabei statt einer Kinect jedoch moderne 3D-Scanner, deren Funktion auf einem ähnlichen Prinzip basiert, um ihren Bildinhalt zu generieren. Dementsprechend deutlich ist auch die visuelle Ähnlichkeit zur Kinect-Ästhetik. Man erkennt die darunterliegende Technologie des Scannens merklich. Obwohl die Auflösung deutlich höher ist, sind auch hier die

Abbildung 5.6: In *TROPICS* von Mathilde Lavenne [22] verwendet die Künstlerin 3D-Scanner um eine Punktwolke dreidimensionaler Geometrie zu erzeugen.

gescannten Bildebenen und -punkte in weiß auf schwarzem Hintergrund deutlich sichtbar. Durch Kamerafahrten entsteht eine Art Paralax-Effekt, der die Bildtiefe deutlich erkennbar macht. Hier wird deutlich, dass beim Prozess des Scannens verdeckte Bildinhalte nicht in der Punktwolke aufscheinen, auch wenn die Kamera sich deutlich hinter gescannte Objekte bewegt. Es entsteht ein schwarzer Leerraum in den Zwischensegmenten der Objekte. Als Unterschied zur Technologie der Kinect ist zu nennen, das in TROPICS lediglich Standbilder einer Szene aufgenommen wurden. Erst durch Kamerafahrten im digitalen dreidimensionalen Raum und durch die digitale Nachbearbeitung entsteht die Bildbewegung. Im Film werden außerdem meist mehrere dieser Scans übereinander gelegt.

Im Laufe der Zeit hat sich die Art der Verwendung in der künstlerischen Auseinandersetzung mit der Kinect stark geändert. Mit dem Wissen über das verwendete Medium stiegen auch die Möglichkeiten. Dabei bleibt die ursprüngliche Materialität und das typische Aussehen der Kinectdaten immer weniger deutlich in den entstehenden Werken erhalten. Der Kinect Sensor wird heute, obwohl die Produktion von Microsoft 2017 eingestellt wurde, beispielsweise für Motion Tracking oder 3D Scans verwendet.

5.4 Vergleich

Schließlich lässt sich erkennen, dass bei der Produktion heutiger Animationsfilme Software in fast jedem Bereich unumgänglich ist. Dabei gibt es verschiedenste Ansätze mit Software zu arbeiten, oder Software für sich arbeiten zu lassen. An manchen Stellen, wie etwa der KI, passiert dies durchaus bewusst und wird als Thema häufig von Künstlern aufgenommen. Der Vergleich des Computers und der KI mit einem menschlichen Wesen ist dabei sehr oft zentrales Merkmal computergenerierter Kunst. Während erwerbliche Animations-Software ein Komplettpaket zu bieten versucht, mit dem alle Aufgaben eines Animatiors erledigt werden können, setzen auf Computer-Code basierende Tools wie Processing oder die erwähnten Systeme Künstlicher Intelligenz auf ein Entwickeln eigener Methoden und Tools. Teils übernimmt dies der Künstler selbst von Grund auf, in jedem Fall wird ihm zumindest die Möglichkeit dazu geboten. So ist es möglich, maßgeschneiderte Werkezuge zu erstellen und einzusetzen. Durch das Teilen von erstellten Modulen und Grundbausteinen kann aber auch auf bereits vorhandener Arbeit aufgebaut werden [52]. Im Bereich der Künstlichen Intelligenz ist es dadurch dem Artist überlassen, unter welchen Parametern er eine KI arbeiten lässt, wie sehr sie seinem Input und den auferlegten Regeln abhängt oder wieviel Freiraum der KI gelassen wird, um zu einem sowohl visuell, als auch inhaltlich spannendem Ergebnis zu kommen. In anderen Bereichen ist die Software eher ein Mittel zum Zweck und soll den Workflow so einfach und schnell wie möglich machen. Programme wie Maya und Cinema 4D decken sich in ihrer Funktionalität sehr stark. Vielmehr ist es der Zugang zur Software, ihre Usability und die Art der Verwendung, die Relevanz zeigen. Dabei spielt Software zwar oft keine bewusst Rolle, sie hat aber dennoch eine große Auswirkung auf das finale Bild. Der Stil eines Werks ist dabei oft untrennbar mit verwendeten Technologien verwoben. Gerade zu Beginn neuer Technologien wie etwa zur Anfangszeit der Kinect, aber auch zum Allgemeinen Beginn der Computergrafik ist die mögliche Stilistik noch stark eingeschränkt. Erst mit einem besseren Verständnis und einer deutlichen Weiterentwicklung eines Tools lässt dieses dem Artist die Freiheit, über Stil und Materialität des Endresultats entscheiden. Die dadurch entstehende Freiheit und die Vielzahl an visuellen Möglichkeiten muss aber nicht unbedingt für eine bessere Qualität des Bildes sprechen. Ganz im Gegenteil sprechen sich Künstler wie David O'Reilly bewusst für (selbst auferlegte) Restriktionen aus, um eine stilistische Kohärenz und eine interessante Materialität und Aussage im Bild zu schaffen [48].

Kapitel 6

Zusammenfassung

Schlussendlich lassen sich einige Aspekte über das Medium Software zusammenfassen. Software imitiert analoge Medien in ihrer Funktion und Materialität, führt sie ins Digitale über und erweitert diese dabei um zusätzliche Funktionen und Eigenschaften. Das Generierte imitiert oft den Stil von Medien, die schon vor dem Digitalen existierten, kann aber auch für sich zu einer neuen digitalen Materialität kommen. Diese eigene, digitale Qualität wird jedoch nicht in physischer Form, sondern über die Verwendung verschiedener digitaler Werkzeuge visuell erkennbar. Lineare Bewegungen, konstante Farbverläufe und mathematisch exakte Grundformen sind einige der Zeichen der oft zu sehr perfekt wirkenden, digitalen Form. Aber auch in Fehlern und Glitches wird das Digitale offenbart.

Abschließend lässt sich außerdem erkennen, dass Software als Medium in der Kunst sehr komplex und wandelbar ist, aber dadurch auch viele Möglichkeiten bieten kann, die andere Formen der Kunst nicht haben. Von starker Abstraktion bis hin zum absoluten Realismus halten uns heutige Programme dabei fast alle Möglichkeiten offen. Die fehlende Limitierung kann aber auch Nachteile für den visuellen Stil entstehender Arbeiten haben. Eine bewusste Stilentscheidung und die damit verbundene visuelle Einschränkung sind wichtig, um einen Film interessant und kohärent wirken zu lassen. Die verwendeten Tools werden dabei immer komplexer und verstecken aufwändige Berechnungen unter der Programmoberfläche vor dem User. So wird der Zugang zum Programm einfach gehalten. Diese digitale Abstrahierung des eigentlichen Arbeitsprozesses bleibt für den Artist aber meist unbewusst. Vielmehr verschmilzt der User beim Arbeiten mit den Software-Tools und deren Interface. Die dabei bestehende Schnittstelle ist für uns heutzutage so intuitiv, dass sie sich nicht weiter bemerkbar macht. Als Digital Native sind viele, der uns zur Verfügung stehenden Grundlagen in moderner Software selbstverständlich und damit unsichtbar. Dennoch tragen auch diese Elemente zum digitalen Arbeitsprozess bei.

Grundsätzlich wird dabei in allen Medien-Programmen versucht, eine

Balance zwischen einer vorgefertigten Funktionalität und einem modularen, flexiblen Arbeiten zu finden. Trotz der großen Menge an erhältlichen Programmen im Bereich der 3D-Animation, unterscheidet sich deren Funktionsumfang nur geringfügig. Nicht jedes erdenkliche Werkzeug kann allerdings in der Software integriert werden. Somit sind Schnittstellen für Plug-Ins und Scripts für komplexe oder spezifische Tasks wichtig. Der Workflow des Animators wird also vielmehr durch den eigentlichen Zugang zur Software beeinflusst, als durch einzelne Werkzeuge innerhalb der Software. Hier sind grundlegende Unterscheidungen in der Arbeitsweise möglich. Techniken wie das Echtzeit-Rendering bieten eine vielversprechende Technologie, die aber zum aktuellen technischen Stand noch zu viele Hürden beinhaltet. Dennoch wäre ein Echtzeit-Feedback der Software für einen reibungslosen, kreativen Prozess wichtig. Nur so kann ein Artist effizient eine Idee visuell im Programm umsetzen, ohne lange Warte- oder Renderzeiten. Für die Wahl der richtigen Software sind schließlich viele Faktoren ausschlaggebend: Neben der Grundarchitektur einer Software können auch Effizienz und vorhandene Vorerfahrung mit einem Tools zu einem schnellen und fehlerfreien Workflow in der Software führen.

Anhang A

Inhalt der CD-ROM/DVD

Format: CD-ROM, Single Layer, ISO9660-Format

Pfad: /

_thesis_DE.pdf	Masterarbeit (Gesamtdokument)
images Folder	alle in der Arbeit verwendeten Bilder
online Folder	alle Online-Quellen als pdf

Quellenverzeichnis

Literatur

[1] Joyce J. Elam und Melissa Mead. „Can Software Influence Creativity“. *Information Systems Research* 1.1 (März 1990), S. 1–22 (siehe S. 12).

[2] Matthew Fuller. *Software Studies—A Lexicon*. Cambridge, Massachusetts: MIT Press, 2008 (siehe S. 11).

[3] Knut Hickethier. *Film- und Fernsehanalyse*. Stuttgart: J.B. Metzler, 2012 (siehe S. 6).

[4] Lev Manovich. *Software Takes Command*. International Texts in Critical Media Aesthetics. New York: Bloomsbury Academic, 2013 (siehe S. 2, 4–7, 9, 15, 17).

[5] Lev Manovich. *The Language of New Media*. Cambridge, Massachusetts: MIT Press, 2001 (siehe S. 10, 11).

[6] Marshall McLuhan. *Understanding Media: The Extensions of Man*. Berkeley, California: Ginkgo Press, 2003 (siehe S. 12).

[7] Stuart Russell und Peter Norvig. *Artificial Intelligence—A Modern Approach*. 3. Aufl. New Jersey: Pearson, 2010 (siehe S. 31).

[8] Zhengyou Zhang. „Microsoft Kinect Sensor and Its Effect“. *IEEE, Microsoft Research* 19.2 (2012), S. 4–10 (siehe S. 35).

Filme und audiovisuelle Medien

[9] *Adam*. Film. Production: Unity Demo Team. 2016. URL: https://unity3d.com/pages/adam (siehe S. 25, 26).

[10] *Chronograph Variations*. Fassadenprojektion, Bilderserie. Casey Reas. 2011. URL: http://www.reas.com/chronograph_p/ (siehe S. 30).

[11] *Clouds*. Film. Regie: James George. 2012. URL: http://jamesgeorge.org/CLOUDS (siehe S. 36).

[12] *Deep Meditations*. Filmreihe. Memo Akten. 2018. URL: http://www.memo.tv/portfolio/deep-meditations/ (siehe S. 33).

[13] *Dinosaurier*. Film. Production: Walt Disney Pictures, Regie: Eric Leighton und Ralph Zondag. 2000 (siehe S. 18).

[14] *#EpicGanGuy2019*. Film. Memo Akten. 2019. URL: http://www.memo.tv/portfolio/epicganguy2019/ (siehe S. 34).

[15] *Influencers*. Film. f°am Studio. 2018. URL: https://vimeo.com/281951902 (siehe S. 22).

[16] *Learning to See*. Film. Memo Akten. 2017. URL: http://www.memo.tv/portfolio/learning-to-see/ (siehe S. 34).

[17] *Mistaken Identity*. Film. Mario Klingemann. 2018. URL: http://underdestruction.com/2018/10/28/neural-glitch/ (siehe S. 33, 35).

[18] *Please Say Something*. Film. David O'Reilly. URL: https://www.google.com/search?client=firefox-b-d&q=plaease+say+something (siehe S. 7, 8).

[19] *Sonder*. Film. Soba Productions. 2018. URL: http://www.sondershortfilm.com/ (siehe S. 25, 26).

[20] *Spring*. Film. Buch und Regie: Andy Goralczyk. 2019. URL: https://cloud.blender.org/p/spring/ (siehe S. 23, 24).

[21] *Tennis Deconstructed*. Installation und Print. Joshua Davis. 2014. URL: https://joshuadavis.com/NikeLab-21-Mercer-USOpen-Tennis-Deconstructed (siehe S. 29).

[22] *TROPICS*. Film. Mathilde Lavenne. 2018. URL: https://www.mathildelavenne.com/t-r-o-p-i-c-s/ (siehe S. 36, 37).

[23] *Ugly*. Film. Nikita Diakur. 2017. URL: https://nikitadiakur.com/0 (siehe S. 7, 21, 22).

Software

[24] *Adobe AfterEffects*. Videobearbeitungs-Software. URL: https://www.adobe.com/products/aftereffects.html (siehe S. 17, 21).

[25] *Adobe Photoshop*. Bildbearbeitungs-Software. URL: https://www.adobe.com/products/photoshopfamily.html (siehe S. 9).

[26] *Autodesk 3DsMax*. 3D Software. URL: https://www.autodesk.com/products/3ds-max/overview (siehe S. 18).

[27] *Autodesk Maya*. 3D Software. URL: https://www.autodesk.com/products/maya/overview (siehe S. 13, 16, 18).

[28] *Autodesk Softimage*. 3D Software. URL: https://www.autodesk.com/products/softimage/overview (siehe S. 18).

[29] *Blender*. OpenSource 3D Software. URL: https://www.blender.org/ (siehe S. 24).

[30] *Maxon Cinema 4D*. 3D Software. URL: https://www.maxon.net/en/products/cinema-4d/overview/ (siehe S. 20).

[31] Taesung Park u. a. *GauGAN*. KI-Applikation. URL: http://nvidia-research-mingyuliu.com/gaugan/ (siehe S. 34).

[32] *SideFX Houdini*. 3D Software. URL: https://www.sidefx.com/products/houdini/ (siehe S. 17, 23).

[33] *The Foundry Nuke*. Compositing Software. URL: https://www.foundry.com/products/nuke (siehe S. 17, 23).

[34] *Unity*. Game Engine. URL: https://unity.com/ (siehe S. 24).

[35] *Unreal Engine 4*. Game Engine. URL: https://www.unrealengine.com/en-US/ (siehe S. 24).

Online-Quellen

[36] *Analog signal*. Nov. 2019. URL: https://en.wikipedia.org/wiki/Analog_signal (besucht am 06. 06. 2019) (siehe S. 4).

[37] *Animateka Masterclass: Nikita Diakur—Ugly Aesthetics and Interactive Animation*. 2017. URL: https://www.youtube.com/watch?v=9UEQAvmZpMM (besucht am 19. 11. 2019) (siehe S. 21).

[38] *ERROR—The Art of Imperfection*. Ars Electronica Festival 2018. URL: https://ars.electronica.art/error/en/ (besucht am 08. 11. 2019) (siehe S. 8).

[39] *Generative Adversarial Networks*. Juni 2019. URL: https://de.wikipedia.org/wiki/Generative_Adversarial_Networks (besucht am 19. 11. 2019) (siehe S. 32).

[40] *Google Deep Dream Generator - Human AI Collaboration*. URL: https://deepdreamgenerator.com/ (besucht am 08. 06. 2019) (siehe S. 31).

[41] Julia Kaganskiy. *Experimental Film Clouds Combines Kinects And DSLRs To Imagine The Future Of Filmmaking*. 2012. URL: https://www.vice.com/en_au/article/z4y7ga/experimental-film-icloudsi-combines-kinects-and-dslrs-to-imagine-the-future-of-filmmaking (besucht am 19. 11. 2019) (siehe S. 35).

[42] Sarah Kutz. *Blender: pros, cons, quirks, and links*. iMeshup. Aug. 2018. URL: https://medium.com/imeshup/blender-pros-cons-quirks-and-links-3a9bf803826f (besucht am 04. 11. 2019) (siehe S. 24).

[43] Sarah Kutz. *Cinema 4D: pros, cons, quirks, and links*. iMeshup. Aug. 2018. URL: https://medium.com/imeshup/cinema-4d-pros-cons-quirks-and-links-1d5009d16c5f (besucht am 04. 11. 2019) (siehe S. 21).

[44] Sarah Kutz. *Maya: pros, cons, quirks, and links*. iMeshup. Sep. 2018. URL: https://medium.com/imeshup/maya-pros-cons-quirks-and-links-4ee1c4eeecc2 (besucht am 31. 10. 2019) (siehe S. 19).

[45] Lev Manovich. *Automating Aesthetics: Artificial Intelligence and Image Culture*. 2017. URL: http://manovich.net/index.php/projects/automating-aesthetics-artificial-intelligence-and-image-culture (besucht am 14. 11. 2019) (siehe S. 32, 33).

[46] *Microsoft Kinect*. Nov. 2019. URL: https://en.wikipedia.org/wiki/Kinect (besucht am 08. 11. 2019) (siehe S. 35).

[47] *OpenAI - Artificial General Intelligence*. URL: https://openai.com/ (besucht am 08. 06. 2019) (siehe S. 31).

[48] David O'Reilly. *Basic Animation Aestetics*. 2009. URL: http://www.davidoreilly.com/downloads (besucht am 12. 11. 2019) (siehe S. 7, 10, 38).

[49] *Plug-in (computing)*. Nov. 2019. URL: https://en.wikipedia.org/wiki/Plug-in_(computing) (besucht am 08. 11. 2019) (siehe S. 15).

[50] Antonio Poscic. *The Pixels Themselves: An Interview With Mario Klingemann*. 25. Aug. 2018. URL: https://thequietus.com/articles/25188-mario-klingemann-ai-art-interview (besucht am 04. 11. 2019) (siehe S. 35).

[51] *Processing*. Nov. 2019. URL: https://en.wikipedia.org/wiki/Processing_(programming_language) (besucht am 04. 11. 2019) (siehe S. 28).

[52] Casey Reas. *Thoughts on Software for the Visual Arts*. Feb. 2017. URL: https://medium.com/@REAS/thoughts-on-software-for-the-visual-arts-690ea7bfc8b6 (besucht am 19. 11. 2019) (siehe S. 29, 38).

[53] Tobias Revell. *Seizing the means of rendering. – Render reality?* 2016. URL: https://www.tobiasrevell.com/Seizing-the-Means-of-Rendering-Amateur-Cities (besucht am 19. 11. 2019) (siehe S. 21).

[54] Margaret Rouse. *What is software?* URL: https://searchmicroservices.techtarget.com/definition/software (besucht am 02. 06. 2019) (siehe S. 5).

[55] *What is Real Time Rendering and Why It Matters*. URL: https://www.easyrender.com/3d-rendering/what-is-real-time-rendering-and-why-it-matters (besucht am 31. 10. 2019) (siehe S. 24).