The Influence of Software on Computer Animation

Klemens Svetitsch

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Digital Arts

in Hagenberg

im Dezember 2015

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, December 2, 2015

Klemens Svetitsch

Contents

Preface

Media theorist Lev Manovich already claimed that "the logic of programming is projected to the GUI level and becomes part of the user's cognitive model of working with media inside applications" [14, p. 222]. Obviously, the ways software may influence art and animation in particular are manifold and some of these influences may be difficult to examine as well. This paper will be not exclusively about the specific animation styles that have evolved as a result of software being increasingly used for animation production over the past forty years. Nor is it going to prove that artists are in fact being manipulated or controlled by the software applications they are believing to be in control of. All of these aspects and presumptions do play an important role, however, in trying to find the connections of the technical art of software development with the artistic techniques of animation.

These connections are indeed very complex and this paper will not go into too much detail about all of the theories already established in the fields of software and animation, psychology and everything in-between, but rather a couple of theories and findings will be picked to support the network of ideas that will be offered.

The general question we will have to ask ourselves when trying to connect software and art is what short- and long-term consequences of creating and using software might be. More specifically, it should be the goal to find evidence for the impact of digital production (through software) on animation. Also, different aspects of software must be examined in relation to people and their workflows—the main ones being the user interface, the way programmers think and programming. Very important might be to evaluate the claim that the usage of software has certain side-effects on creativity, literacy and other cultural conditions, but this will be beyond the scope of this paper.

Since technology plays such an important role in the process of animation production, the history of technologies involved in creating it and the associated ways of thinking about animation are essential to understanding what might have changed with the introduction of computers. But before we can delve into the technicalities of animation, we have to deal with the nature of software, the tool that is used to shape the vast majority of today's media and thereby culture, and that is itself manipulating the ones

manipulating the masses.

As far as software is concerned, no full account of all its dimensions can be given. For one, software, as we are encountering it today, is still not comprehensively examined as an artifact in regards to the interaction and interchange processes with humans; for the other, I argue that software as a technology is still in its infancy. The reader may agree or not, but this should be sufficiently evident from the missing subtlety and sophistication of its integration into our everyday lives.

What we have to think about is not only whether software in its current state, covertly causing social and political upheaval, economic restructuring and collective reprogramming, could, apart from its apparent positive effects, also have negative ones. Also, new concepts of thought, concerning the place and purpose of the human among their tools, are necessary to cope with the accelerating technological advancement that might sooner or later do to its programmers what it is constantly doing to itself: make them redundant.

As the probably most universal, and still expanding, non-interactive art form of our time, animation shall provide the examples necessary to make the influence of software on animation evident and believable. Lev Manovich, who connects diverse aspects of culture, software and art, and Paul Wells who puts animation in a wider cultural context and also thinks about its interaction with the technologies, both indicate this relationship and are essential sources documenting and researching its history.

Abstract

Animation has always involved machines and contraptions, but with the advent of computers a totally new era of animation has started, driving both technology and software as well as different animation styles. But these strong connections between software and animation lead not only to innovation and a prospering of diversity but also to a convergence of animation styles and the disappearing of techniques. The logic of the computer and software has an impact on the software tools available to artists and thereby subtly influence what artists can and want to do and how they work.

Software is based on principles like automation and abstraction, which reflect out onto the user interfaces of media authoring applications and possibly even onto the animated works created by the artist. As the central communication platform for software and artist, the user interface decides how much power the artist has inside an application, how restricted they are and whether they get inspired by the tool or not.

Since animation programs are complex and technical pieces of software, animation artists using an application may experience significant pressure from the tool, as is normal for any tool, instead of being able to receive their imagined artwork at the click of a button. While a certain degree of manipulation from a tool is natural, software has very specific properties that lead to particular influences in how animation looks and is thought about. These can likely be connected back to the logic of programming and software development itself.

Kurzfassung

Obwohl Technologien mechanischer und chemischer Natur schon immer eine große Rolle für die Animation spielten, wurden durch die Verbreitung des Computers technologische und visuelle Entwicklungen nie dagewesener Dimensionen ausgelöst. Doch diese enge Verbindung von Software und Animation hat nicht nur Innovationen und eine aufblühende Vielfalt in der Animationsszene zur Folge, sondern auch eine Verdrängung bewährter künstlerischer Techniken und Stile. Die Logik des Computers und der Software hat Auswirkungen darauf, welche Programme Künstlern zur Verfügung stehen und damit auch darauf, welche Möglichkeiten diese durch die Software erhalten und wie sie damit arbeiten.

Software zugrunde liegende Prinzipien wie Automation und Abstraktion wirken durch alle Ebenen der Softwareentwicklung, bis in die visuelle Bedienoberfläche, und haben so subtilen Einfluss auf Animationskünstler, ihre Arbeit und ihre Werke. Die Benutzeroberfläche entscheidet, wie viel Macht dem Benutzer gegeben wird und somit, wie stark der Künstler durch das Werkzeug eingeschränkt, unterstützt oder sogar inspiriert wird.

Wie auch andere Werkzeuge, drängen Animationsprogramme den Künstler in bestimmte Richtungen, statt auf Knopfdruck das vorschwebende Ergebnis zu liefern. Auch wenn diese Manipulation bis zu einem gewissen Grad natürlich und häufig erwünscht ist, hat Software auch Einflüsse, die für den Benutzer besonders schwer erkennbar sind und die teilweise bis in technischen Details der Softwareentwicklung zurückverfolgt werden können.

Chapter 1

Introduction

1.1 Technology and Technique

In history, there have been quite a lot of definitions and notions about the meaning of the term technology, starting out from the original Greek word *techne* to the industrial age when it received many new connotations. There have also been lots of disputes about the correspondences of the English terms "technology" and "technique" and the German "Technologie" and "Technik". Though it should be noted that "Technik" carries similarities with both English terms and "Technologie" doesn't seem to be quite the same as "technology", it shall not be discussed here any further because I simply find that there is a lot of confusion about the German terms and that "technology" and "technique" are more accurate to describe the differentiation I am intending to make. Without having delved into the depths of their previous connotations I decided to use these terms as follows, based on how they are currently understood [48]:

- A **technique** is a procedure (an algorithm) with the purpose of reaching an objective.
- A **technology** is a collection of techniques that by means of an artifact can fulfill a task not fulfillable by humans or optimize a process performed by humans in an unprecedented/unequaled way. It is thereby set to the goal of controlling/harnessing certain natural forces.

Technology relates to technique in that way because generally we identify devices as technology whose task requires an intricate understanding of how the task is performed and of the matters involved in it. While a technique can be applied by anything that is physically equipped to do so, a technology can only be developed by an entity capable of imagining what the technology is supposed to do and of implementing it physically.

A few simple examples should serve to clarify the distinction and to point out that it is not always as unambiguous as the definitions suggest: An artist using paint and brush to create a painting applies a technique to accomplish

this. The structure of an airbrush takes advantage of the physical behavior of liquids and air to allow an artist to apply color onto a surface in a very consistent manner. A teacher makes use of certain techniques to keep their students busy in class and to motivate them to study. Face recognition is using several algorithms that are implemented and embedded in another hardware technology (e.g. a computer or a camera).

Evidently, the two definitions are based on certain implicit assumptions and have a couple of implications. First, technique per se is a purely virtual construct existing in our heads and, in order to manifest, requires a medium that it can be applied to. Consider artisans or artists who are working with physical materials to transform them into something useful or decorative. But virtual things can also be subject of techniques, for example learning and memory techniques that transform words on paper into memories and, hopefully, understanding.

Another important observation about the term technology is that it denotes both, a virtual concept and the physical artifact exploiting it and making it accessible and useful to humans. The artifacts incorporating these "technologized" techniques are called tools because people are using them in order to do things faster than without or to do things they would not be able to do without. The following conclusion is important: *A technology is basically the concept behind the performance of a tool that more or less radically reduces the necessity and possibility of human intervention.*

1.2 Tools

Humans have been building tools for 2–3 million years [49]. The main reason to do this are physiological limitations—that are our size and strength—that we had to compensate for. A tool is therefore an extension of the body and the mind, expanding the possibilities and powers of the user.

Tools can be very different in many ways. What we commonly understand as tools are mostly devices that are operated physically, by hand, like a hammer or a brush. But there are also other tools that are—though they are operated through mediation by a physical device—not physical themselves and not actually perceived as such, like computer programs. There are also big differences in complexity, as far as concerns construction as well as operation. For example, while it is relatively easy to build a hammer and to use it to drive a nail, one might say it can be much more difficult to use a paint brush in a way to produce what one has in mind. Again, it is a completely different thing to construct a factory or a car than to operate them. When it comes to the relationship of the user and their tool, an asymmetry of powers can be assumed natural and unavoidable. This asymmetry mirrors the tension between the two roles people can take towards tools: the role of the toolmaker and the role of the tool-user.

This dichotomy seems mainly to be the result of social and psychological mechanisms that, though having efficiency as their asserted goal, are causing people to separate into groups defined by certain skills and interests. The separation which is deeply fixed inside the minds of everybody living in a so-called "civilization" is a separation of professions. Although technically the sword does not stand between the smith and the swordsman, a blunt blade certainly can. Still, the tool also connects the maker with the user and their relationship is shaped by how both approach each other and the tool itself. Only if both are concerned with the thinking of the other can the tool really be powerful.

Tools are the most important subjects of techniques. Around every tool—be it a paint brush or a 3D animation package—a whole collection of techniques accumulates over time, with some going out ahead and others falling into oblivion. Precisely these techniques are what defines the tensions not only between tools and users but also among the users themselves. The result are target groups which systematically divide users by their skills and intentions. While seemingly connecting people with each other, tools separate the makers from the users and the professionals from the amateurs.

It is regarded necessary that a goal exists in order that a technique can be developed. If a task is performed without following some sort of procedure or strategy it is usually not called a technique. It is however not necessary that the manifestation of a technology (e.g., a piece of machinery) has an actual goal or purpose in order for it to appear as such, because intent and purpose cannot necessarily be deducted from appearance. Rather, we sometimes deem things as technology because of their morphological features which reflect into our assumptions of their underlying complexity. This might (perhaps intentionally) result in the situation that a piece of art is seen as technology but not as art despite its being anything but technology.

1.3 Interplay: Artist and Tool

The three essential kinds of reflection artists are practicing are the reflection of themselves where they are trying to show their feelings and conceptions, the reflection of their environment like culture and society and the reflection on their tools and media. Some will argue that tools are actually part of the artist's environment but I would suggest that there is a difference between an examination of, for example, social dynamics or political developments and the exploration of the properties of a tool.

At first glance, one feels tempted to rank the three categories for idealistic reasons, but this does not make any of them less relevant. On the contrary, it underlines the natural misconception that every person is prone to due to their civilized cultural upbringing. It is conveyed to us that social and

personal issues should be the topmost priorities of a good person and this is enforced by general affirmation through mediation mechanisms. The tool is not culturally seen as part of a person but as a separate entity which makes it less likely that people reflect and critique on them as they do on themselves. Well-known media philosophers and theorists like Marshal McLuhan or even Sigmund Freud have made sufficiently clear that the tool extends the body and also integrates with it in our minds [7, p. 48], [17, p. 63]. Thus, tools should be just as much the subject of art as they are subject of public discussion.

Earlier I pointed out that the asymmetry between user, tool and medium is cause for a tension and I argue it is this tension that the artist is trying to overcome by working with their tools. For many artists the foremost wish may be to realize their visions—idealistic or deeply personal—but to feel one's tools and materials and to experiment with them, just like the baby playing with its hands and feet, feeling them and getting to know them, is an approach just as valid. It is these two approaches—to prioritize form (the tool in the broadest sense and what it can do) or the idea (message)—that Scott McCloud explains in his book *Understanding Comics* [16, p. 179f.].

1.4 Media

The way we experience the world, understand it and learn from it, is by means of sensory perception. We perceive configurations of matter and structure the visual, auditory etc. information corresponding to our existing concept of the world—built from our previous experiences—which new ones must fit into.[1] However, the brain is also capable of connecting together ("associating") different experiences or phenomena, and in cases where this bond becomes explicit and consolidated, "signs" are created according to semiotic studies [2, p. 2]. For what is usually understood as a sign in semantics[2], like words or symbols, particular properties and proportions must exist between the sign and its meaning.

Let us consider several examples from popular media types. In written language, words are purely abstract while their meaning can be immensely diverse but almost always involves some kind of abstract notion of a real-world phenomenon, artifact or a concept.[3] The meanings of words like "tree", "home", "hypotenuse" or "theory" are learned and require understanding and the ability of abstraction. Pictograms on street signs, product packages or in written documents resemble physical objects and thereby evoke an

[1] No doubt, this is not going to change until technologies have been developed that can emulate impressions directly to our brain, thereby bypassing our sensory system.

[2] Semantics is the study of meaning, or more specific, of signifiers and the signified, the artifacts signs are associated with [50].

[3] Words that exist due to grammatical peculiarities and might negate this statement are not considered relevant here.

association with them [51]. Visually, they are no different than words, from an objective point of view, but in the context of the visually perceptible real world they are more strongly connected with their meanings than words.[4] Then there are also static and moving images like photographs in magazines or film on television.

Media are constructs that possess the ability to carry signs. They can be used to transport messages from one place or time to another, but more importantly from one entity to another. Since all media are physical or physically based, they also influence the messages they are carrying by their inherent limitations. Any artifact capable of holding a sign is a medium but media are very different in terms of what signs they can carry and what effect they have on the message.

The actual purpose of media is to transport information that is generated in one point to another. In personal communication this is clear. When looking at mass media like television, movies or news sites however, another effect becomes apparent. Mass media technologies make possible the separation of content production from consumption, thereby reducing the overall cost of distribution (in a financial but also general sense). By a historical comparison, bards, messengers and story-tellers have become redundant because of YouTube as the global bard, television as the centralized news messenger and movies as oligarchic story-tellers.

1.5 An Account of Animation

If technology is the endeavor to make the hard easy and the impossible possible, the goal of animation would be to make that which is impossible to see visible, that is to show a world that is incompatible with reality. In the first statute of the ASIFA it reads: "[...] animation cinema creates the occurrences using instruments different from those used for automatic registration. In animated films, the occurrences take place for the first time on the screen" [65].

Admittedly, scientific instruments like microscopes, telescopes and x-ray machines also have the purpose of making the invisible visible and this parallel should not be dismissed lightly. In fact, scientific research is vividly connected to animation as firstly, it required massive technological progress in order to even enable animation production—this is still far too obvious for digital animation—and secondly, animation is the main means of visualization used by scientists to demonstrate their findings. This was one of the earliest drivers of the development of computer animation as the most general and powerful technology in animation.[5]

[4] Since words have an auditory representation, their pronunciation, they are also strongly rooted within our memory due to this double-association.

[5] Edward Zajac created one of the earliest computer animated films in 1961, which

As opposed to expensive instruments, due to its nature as a simple sequence of images, animation can be consumed by practically anyone in the whole world. Nevertheless, to support the above statement about media centralization, only few are capable of making it. That is because animation takes a lot of effort and expense to produce. It is even more expensive than live-action movies and therefore the decision which format to use for telling a story is critical. In effect, in most cases when animated movies are made, the narrative and visual language make use of the extraordinary capabilities of animation.

Abstract animation may be at the core of the concept of animation because it is the prime example of phenomena that cannot be seen in reality or shown in any other way but by means of animation technology. Most people, however, seem to be averse of the disruptive nature of the majority of abstract animations and therefore different, more figurative themes dominate in the animation industry for the sake of entertainment. There are now various typical places in movies where the "animation aspect" has been made to excel: locations can be surreal, magical or futuristic; people can do impossible things, survive deadly attacks, dissolve and reassemble; animals can talk and take the place of humans as main characters; imagined magical creatures like dragons or even more exotic ones are used.

Animation is a playground for the imagination. The previous restrictions of live-action filming are gone and with photo-realistic visual effects most of nowadays' blockbuster movies can be called "hybrid" at the very least. Apart from extremely visible effects created to transform the imagery altogether, compositing techniques are used to alter images in ways that make it difficult to identify them as computer-manipulated.[6] Looking at animated feature movies, one might observe that the quality of the narrative *generally* is very high and oftentimes critical points of view and themes are taken up.

1.6 Animation from the Software Perspective

When we think about how animation, with all its sub-forms, and software, with its plethora of shapes, permeating culture deeply, interact and continue to form new expressive forms, it is clear that no uniform way of characterizing these interaction mechanisms and mutual influences can be described. Rather, a certain hierarchy of involvement of software in the creation of cultural products must be developed, not only to distinguish certain levels of impact but also the different types of applications used. The following

simulated a satellite orbiting the Earth [22, p. 151]. James Blinn, whilst working for NASA on visualizations of the Voyager missions, solved major computer graphics problems [23, p. 48ff.].

[6] These manipulations are called *invisible visual effects* [81]; further examples can be found in this video: https://www.youtube.com/watch?v=clnozSXyF4k, referenced in [69]

scale is based on the prevalent terms or "genres" used in the industry[7] and uses their language to describe how deeply software is integrated into their production; the examples given (whether of technique or technology) are to be seen as major drivers of developments, not as the only ones:

1. **Analog/Traditional**: Computers are involved not at all or in ways or degrees that are insignificant to the effective outcome of the process. For example, computers might be used for accounting or project management. Whether the writing of a screenplay on a computer already has measurable impact on the outcome, is arguable. Examples for this mode of work are all animated films prior to the spread of digital image processing software, like Ivan Sutherland's *Sketchpad* (1962) [23, p. 41ff.].
2. **Digital Post-Production**: While the production (animation) itself is done in purely traditional ways, computers are used to enhance the images. Examples are the stop-motion technique where the photographs are graded and artifacts from the manual process removed, or cel animation with a digital finishing step.
3. **Digital 2D**: With Disney's *Computer Animation Production System* (CAPS) came a set of tools that allowed for large parts of the old *ink & paint*-based process to be replaced by digital workflows. Though parts of the skills required to create good animation (like drawing and mastering the principles of animation) were still necessary, other tasks connected to the previous cel-medium, like painting and inking, were abandoned (together with the corresponding workforce). The new medium led to lower production costs and the animators (solo artists in particular) being able to focus on the essential aspects of animation (image and motion) rather than a laborious and difficult process, but of course, knowledge about certain crafts were lost as well.
4. **Visual Effects**: They are used to manipulate and enhance "live-action" footage, to extend and complement the recorded "real". This category is itself so extensive that it is difficult to set it in relation to the other modes, but although it came chronologically before the above digital 2D animation practices, the general degree of computer usage in the VFX industry (nowadays) results in a far greater exposure of live-action movies to side effects of the digital.
5. **3D/Full-CG**: After the integration of computer-generated imagery into live-action footage, the logical next step was to create the whole movie inside the computer, which was successfully done for the feature format in 1995 and for the short form several years earlier. In cinemas, the 3D-film as the now-prevalent form of entertaining animation has

[7] I consider *Genre* a valid designation because the categories are based on a common understanding in the industry and the employment market of the involved workflows and technologies. Genre is usually associated with an agreement of a market.

replaced the cartoon and its visual style, while adopting large parts of its language. Since computers play a primary role in the creation of 3D films and they have ubiquitous areas of application, it will be necessary here to examine practices linked to the usage of software and computers on the one hand, and the changes of practices taken over from previous working methods on the other.

6. **Procedural Animation**: While the digital 3D-animation approach still involves the manual placing of key-frames on a timeline and the manual creation and manipulation of virtual objects, once programming is accepted as the more powerful and universal tool in creation, the artist is liberated from the repetitive processes of manual animation and the rules specified in the programming code are left to reign over the details of motion and shape. In procedural animation, the principles and the knowledge developed over decades of manual animation practice are formulated in code and integrated as basic rules (if they are at all desired), together with the general implementation—instructions and procedures—of the artistic vision. The artist is now no longer the user of a tool, building their world piece by piece or frame by frame, but the conductor of a spectacle which they can watch and make changes to (if they dare), in service of the machine's logic.
7. **Virtual Reality**: With devices that make possible the immersion of oneself in a virtual world comes the necessity to create the diverse alternate realities on the large scale. Animation is interactive and generative, ultimately: real. The frame is finally abolished and the logic of the computer is made completely transparent to the user who adapts to the new paradigms, rendering unnecessary and unlikely the critical examination of the underlying mechanisms.

1.7 The Logic of the Virtual

Any artist will confirm that the blank page, the emptiness, cannot be the sole ground for creation, but rather, the environment, the "eco-logic" of the surroundings are necessary for inspiration and innovation, ultimately something that we might call creativity. As intimidating as the blank page can be, lacking the media and tools to realize one's vision can be just as frustrating. To deal with this situation is the actual goal of the digital, to level out all preconditions and bring all tools down to a single, blank page on which anything imaginable, therefore **virtual**, can be created.

The logic of virtuality—de-construction and re-construction—sometimes seeming to be the driving forces of all human endeavors, is also the principle of art and animation. The power to create a world after one's own vision is inspiring as well as terrifying in its ability to transcend all limitations that we experience in real life. But in reality, the limitations of the computer

are just as present as the limitations of the physical. There is a fundamental difference between the two, however, that is to the traditional artist, physical objects are potentially source of inspiration, part of the art or even a tool, while the digital artist, whose philosophy is to deconstruct and recreate, can draw inspiration only from the digital world or face the obstacles of having to engage deeply with technical matters and limitations in order to simulate the non-digital. In nature one has an arsenal of materials and shapes at their disposal; in digital one always has to rely on other people's work or one's own ability to return to one's senses after a journey into technicalities and details, to a broader perspective of the real once more.

1.8 The Structure of this Paper

So now we have looked at several terms that are essential to the discussion about the influences of software, as it will be held here. After an introductory chapter about **Software**2, the circumstances of its production and its properties, multiple perspectives will be taken up, one after the other revealing a tiny part of the image that we might have of the cultural and industrial factors of software.

As the most visible part of software, first the **user interface**3 will be discussed in regards to what (subliminal) influence it has on the user/artist. Then a closer look will be given on the workings and mechanics of the **tools and operations**5 inside software themselves and how they steer us in our creative work. Software also has a major impact on our **workflows**6 and, consequently, on how we think about our own work and how we plan our lives. Last, some **visible effects of software**7 on animation (or animated film) will be pointed out and their roots discovered.

Chapter 2

Software

Although the term **tool** has already been described in depth, when it comes to software it will be used here in two different ways. One is to emphasize the character of software or a software application as a tool to the user to create something, as has already been established. The other artifact we can identify as a tool in the software context is, of course, the functionality, operation or "tool" inside an application. Everyone should be familiar with "tool-palettes" from Photoshop or almost any other media authoring software with a GUI.

2.1 Classifying Software

This paper will primarily focus on the professional software tools used in the animation scene but it is nevertheless interesting to have a look at what software is used in general and how.

Here are a couple of properties that may be useful when we are trying to classify a piece of software. We may consider whether it is for

- **professionals** or **consumers**,
- **visible** or **invisible**,
- **graphical** or **textual**, or for
- **general** or **specific** purposes.

These are only some of the possible distinctions we might make, but they are the most important to find out how an application interacts with the user.

Whether an application is intended to be used by people of a certain **profession**, like animators or technicians, or by just anybody finding it useful, is of great importance to how it is supposed to be designed. This shall be discussed in detail later on.

Visible software applications are all around us. They are in computers, smart-phones, smart-watches, glasses, cars, TVs, ATMs, inside our internet

browsers. But invisible programs are even more so. The internet is home to countless web-servers delivering content to us, bots and crawlers continuously scanning the web for information, and analysis mechanisms observing our behavior on the web. Even on desktop computers and smart-phones most processes running are invisible, so-called daemons. Most of them perform simple and useful tasks like drivers that provide an interface with hardware devices, but there are also dangerous Viruses and Trojans which are just as invisible to the human eye as all the little helpers we appreciate.

In the early days of computers it was at first impossible and later still difficult to build applications with a **GUI** (which would encompass anything that is not text) because of the limitations of the hardware. Graphical elements like buttons or draggable windows have been made possible by inventions like the mouse and hardware components capable of handling the additional computational strain caused by the complex visual representations. The command line interface, though, has the advantage that the developers can focus entirely on the functionality of a program as opposed to its looks, which is why most programming environments default to a non-graphical project on start.

The specialization of an application usually doesn't have an immediate effect for the user because they are often only interested in solving the imminent problem at hand. What kinds of problems the software addresses in general is irrelevant in this case if only it provides the proper solution. Quite relevant, however, is the quality of the user interface and the tool implementations, which is in some way coupled with the degree of focus the developers can raise to attend to the small details in the application. Large applications like 3D-suites and compositing programs often aspire to satisfy a plethora of needs and thereby miss one of the key requirements that is to provide the tools the user wants with the greatest efficiency possible. Having one hundred unneeded tools doesn't make up for a single tool missing at the wrong time.

Highly specialized applications like SynthEyes[1], Dragonframe[2] or xNormal[3] follow the strategy of giving the artist exactly the tool set they need to complete a specific job. Apart from offering all of the tools one will most likely be looking for, they have the advantage of being very small and cheap in comparison to the "Swiss Army knives" of 3D or animation.

One of the core conflicts between user and developer is brilliantly brought to the point by Kostas Terzidis: "[...] the programmer is able to provide those tools that are believed to be needed [... but ...] is unable to provide the means to create the tools that are not believed [...] to be needed" [24, p. 77].

[1] https://www.ssontech.com/
[2] http://www.dragonframe.com/
[3] http://www.xnormal.net/

2.2 The Properties of Software

2.2.1 Learning to Use Software

Any heavy software user will agree that software can have a major influence on the way we think and act. Those not overly familiar with computer programs may take as a proof that it usually takes people not having used computers before a rather long and wearisome time to get accustomed to the ways of thinking required to successfully operate a computer and the different applications on it. While it may be argued that learning a new application takes time because new features, command locations and tools must be learned, this argument alone is not sufficient to explain the difficulties emerging at the first contact with a computer.

The process involved in learning to work with a computer in general—system settings, the file browser and the different mechanisms essential to navigation—is much more complex than what is happening in the brain for the second, third or n^{th} application. The brain has to adjust to the machine's interaction principles and logic which are very different from anything we experience in the physical world and the interaction with humans.

Consumer software has to make it easy for people to get accustomed with it and a number of different features are usually implemented to accomplish this. One of them is the group of "auto"-operations whose purpose is to shorten a complicated or technical manual process by automating it based on a number of input parameters and providing the inexperienced user with only a simple button, virtually saying "Start".[4]

2.2.2 Automation

Automated processes can be life-savers in many every-day tasks with computers, be it batch-renaming, auto-formatting or a spell-check. However, there is a distinction to be made between different kinds of these tools. Operations that can be simply called useful are the ones that take work off our hands that is purely repetitive and tedious and that not necessarily requires human attention or intervention. They can also do work that emerged only as a result of limitations of the computer, for example the conversion of data from one format into another. As long as no actual intelligence is needed, everything is fine.

The tools of the second kind appear quite similar to the first ones in the way that they also automate tedious processes. The nature of them, however, is that sometimes it occurs to us when looking at the results of their work, that they have not arrived at the exact result that we expected. Now it may happen that most of the time a certain spell-check or match moving tool

[4]The 3D match moving application SynthEyes indeed provides such a button, labeled "Auto", but overall it targets professional users and requires experience to operate.

does everything as we wanted, but then there is always the exception. And in those cases it is often difficult to manually correct the mistakes.

Such tools which abstract processes that cannot be fully automated cause an issue that becomes especially severe when people begin to **not** learn how to perform the task manually any more. At a certain point a tool may be sufficiently sophisticated and intelligent to make human intervention dispensable, but still, in most cases a tiny probability for mistakes remains. That is why users need first to have a basic understanding of the processes and and why the tool must provide enough information and control to the user so they can steer the process in the direction they want.[5]

2.2.3 Abstraction

One of the primary principles of the procedural programming paradigm is that it supports and enforces the abstraction and automation of processes. In fact, these two are very similar in the way that once a process has been automated it can be abstracted by substituting it with a signifier that stands for the whole process. The notation of algorithms is very similar to this as usually only the steps necessary to understand it are included, whereas self-evident basic steps are grouped together under a code word. In contrast to human languages which are hard to abstract because of their complex and diverse semantic expressiveness, programming languages are designed to allow for definition and substitution. Without this, with the low level at which programming languages technically operate, it would not be possible to build large applications because a programmer would not be able to handle the gigantic program structure.

Abstraction is a very useful concept because it allows the programmer to think in junks that can be efficiently handled by the brain. By packing a whole algorithm together into the notion of a **function** whose outcome is predictable but not necessarily easy to compute, the programmer can again put together an even more complex program that in the end will be represented by just a single word. As wonderful as it sounds—being able to stack finished parts on top of each other without having to rebuild the whole thing each time—in reality all algorithms and often whole applications are implemented many times. The reason is that software structures tend to be complicated and hard to grasp and one never knows in beforehand how particular components will be working together. A very common problem is that structures, due to changed requirements, suddenly need to possess a flexibility that was not planned for.

[5] In a bit different context Kostas Terzidis stresses as well that technical understanding (of algorithms) is essential to gaining control over software as complex and automated as CAD applications [24, p. 78].

2.2.4 Parameterization

The next step, logically following the concept of abstraction which takes complexity but also control away from the programmer, is the introduction of parameters. A parameter is a piece of information that is used to control the workings and outcomes of a function. For example, to a program that builds skyscrapers we could pass a number that indicates how many floors should be built, or we could create a simple function called "sort", uniting all available sorting algorithms, and, by passing a parameter, state which algorithm should be used. The diversity of ways parameters can be used to control aspects of software, technology or art is restricted only by the underlying platform [14, p. 220ff.].[6]

The first kind of parameters simply define what subject (we might call them data or media) the function (technique) shall be applied to. This can be tricky because the data to be transformed might have several representations used by different applications and therefore a conversion must take place. Some applications—which are also, non-technically speaking, just complex functions with lots of parameters and options—take it on them to perform this themselves; others need to be fed the exact right data types.

The second kind of parameters are actually used to transform the operation itself. When changing the size of a Photoshop brush, the size of a blur filter, the maximum distance of a vertex-merge-tool or the resolution in render-settings of a 3D package, we tell a procedure how it should do its work and we expect to observe a predictable effect on the result.

But functions, tools and operations are not parametric in the first place. Making them so can be a difficult task because decisions have to be made about what aspects of the process are supposed to be outside-controlled and by what means and to what extents this control can be given. From a visual point of view the principles of abstraction and parameterization allow to build a system that obeys certain rules and that can be manipulated easily within its constraints [22, p. 95]. If abstraction makes repetition possible, parameterization adds to it the possibility of non-uniformity because control can be exerted on the individual manifestation of the idea[7] and not only on the general concept. In the computer context this enables not only the creation of procedural artworks but also complex crowd-simulation systems that try to copy realistic dynamics of large masses of people by letting individuals move according to precise, but nevertheless seemingly organic, rules.

Since the author's intention when creating a parametric system is crucial to its power, evidently, conflicts arise between the actual implementation

[6] Here I am not referring to operating systems but to any environment that allows for the implementation of a parametric procedure, like programming languages or media applications with tools for automation.

[7] http://en.wikipedia.org/wiki/Theory_of_Forms

and the user's desire. Every parameter that is exposed adds possibilities on the one hand but can also cause confusion on the other. Usability is a core interest of technical development and it usually stands against the power or controllability of the product. The easiest to use devices are the ones with only a single button and the clearest applications are the ones with few menus and easily accessible and comprehensible functionalities. But this restrictiveness takes away a lot of the flexibility people need and want to create something they can really feel connected to because it reflects their complex reality.

2.2.5 Customizability and Mutability

In many games it is possible to change one's avatar and name, clothing and equipment, sometimes even when those thing are not relevant to success or goals of the game. Lev Manovich sees such kinds of superficial customization critically: options are chosen from fixed sets and at best items are combined from fixed inventories. Some applications let the user change the color of the user interface, the size or type of fonts, icons or let them place and arrange widgets on a screen. But every question, choice or artwork ends in a process of picking from options—they are discrete, digital. He argues that this is the "new logic of computer culture" where nothing is built from the ground up but rather every new piece is a combination of existing, or even worse, ready-made parts. Libraries from which prepared assets or projects can be picked constitute a mixed blessing because whether they are actually helpful or holding back creative initiative after all depends on how one approaches these gifts and tools in general [15, p. 124].

In an economical context a degree of efficiency has to be achieved and working long hours with the software is a natural part of people's jobs. Professional users are going to demand flexibility and adaptability of the user interface and tools to their needs. In contrast to beginners who are easily overwhelmed by overly complex interfaces, they are working with their software tool every day and know every one of its features by heart.

As indicated in [3, p. 13] there are some software tools that are very difficult to make adapt to a workflow in a particular production environment whereas others embrace or even encourage **customization**. The features offering faked individuality to the user (like themes or skinning), which Manovich refers to, should not be confused with this kind of deep adaptability some applications allow in order to optimize the usage and workflow with the interface. For professionals to be able to efficiently work with an application it is in most cases indeed necessary that they can easily adjust the accessibility of tools and their settings.

If the options are consciously used it is also helpful if presets and saving and restoring of configurations of tools are supported in order to speed up usage of frequently used settings. Additional functionalities like those

necessary for embedding the application into an existing pipeline should also be easy to be integrated, especially because compatibility with other applications is a crucial quality professional software must live up to.

By nature software applications are more like sealed machines (**black box**) running some magic than like a juggler who one can watch doing their amazing stunts. There are several ways of retrieving information about the status of code execution. Code can be installed inside the application that writes log files to the disks or shows information on the screen, some applications can sort of penetrate other programs and read data from their memory (hacking). If code is compiled into machine code there isn't really a way of directly intervening with the execution on a high level. If, on the other hand, the source code is interpreted at run-time instead, as in the case of JavaScript, the interpreter can indeed make performance measurements, for example, or allow debugging.

The two most common ways of breaking with the inflexibility that applications cannot be altered after their compilation are available in almost every major media authoring application: scripting and plug-ins.

Plug-ins

They are pieces of software that interface with a host application and usually can access its functionality on some level (through an API). Plug-ins extend the capabilities of the application and can be installed and removed at any time. Data created with the operations of a plug-in often depends on the specific plug-in to be installed on the machine where the file is opened, however. Large media applications like 3D and compositing software depend highly on the plug-in mechanism because many users require special functionality that cannot be included in the default set of tools, or the application would become too big and possibly unstable. By delivering the software with a number of basic features the core installation stays minimal and every user can add their preferred plug-ins later on [77].

Many plug-ins provide automation and preset options that the base applications do not have. There is a variety of plug-ins for *After Effects*, for example, that simplify and speed up diverse processes of creating motion graphics and complex animations. The strategy or workflows intended by a software manufacturer might thereby be overridden, allowing a deep customization of the application according to the design of another software developer. After its installation into the software a plug-in practically becomes part of the main software, as far as the user is concerned.

Scripting

With "scripting", code is executed by an interpreter entirely inside the application itself, which has the advantage that it is much easier to use and

deploy than plug-ins. It is supported mostly by bigger software packages like Maya[8] or Nuke[9] and can be a powerful tool in the hands of artists and pipeline designers because they usually allow access to at least all commands available through the visual interface, if not more. These commands are provided through an API (application programming interface) which is tailored and unique to the host application and therefore requires proper familiarity with its features and peculiarities. Against the ease-of-use also stands the fact that applications may offer their own scripting languages to control it.[10] Although some particular languages like Python and Javascript have become increasingly popular for this precise purpose, when dealing with many different packages (which is common for studio pipelines), one might have to learn several programming languages to implement the same functionalities in all of them.

Versions, Updates and Patches

Software has a peculiar property most other media have not—the tendency to make itself redundant.[11] There is not only good software and bad software like there are good TV shows and bad TV shows, rather even good pieces of software are permanently reinventing themselves, rendering their older versions or even their older functionalities outdated and expendable.

This property that software constantly mutates and evolves is innate and completely integral only to digital data due to the fact that only data, as a technology, can be seamlessly replaced at least on the application level. Versioning and updating are observable not only in data structures that feature this capability like tables, documents and databases, but also in games, office applications and operating systems. Since they typically consist of an enormous amount of code they are especially prone to errors and failure. Frequent updates are the inevitable consequence.

But with these updates come not only improvements and praised features but also changes that make it extremely difficult for users to develop consistent workflows and to work consistently over a longer period of time. New (better) file formats, altered (improved) features and the removal of features and support for file formats lead to loss of data that has been created with an earlier version of a program and not been converted to the new format. They frequently lead to disorientation after, perhaps automatic, updates or even downright inappropriateness of the application [71].[12]

[8] http://en.wikipedia.org/wiki/Autodesk_Maya

[9] http://en.wikipedia.org/wiki/Nuke_(software)

[10] Maya uses MEL (Maya Embedded Language), 3dsMax uses MaxScript.

[11] We still read books from the 18th century and watch movies from the beginning of the 20th, but nobody uses software that is more than a couple of years old.

[12] Such a case is Apple's initial release of *Final Cut Pro X* which, allegedly, made scores of professionals stay with the earlier versions or switch to other alternatives. Updates to the application were offered after discontent was expressed [72].

Dynamic Programming

The overall goal of the above structures is to make software more dynamic, to add and remove features as needed. But programming languages can also have "dynamic" features themselves that enable the programmer to think more immediately and work more expressively and faster. Dynamic features make use of methods like Just-In-Time compilation or interpreting, all of which cause overhead at run-time. But with computer power steadily increasing it is now becoming possible to introduce new dynamic capabilities into productive software despite the need of good performance. At the moment expressive languages are primarily used for the implementation of high-level logic or prototyping because they are still too slow to run operating systems or animation software on. The underlying consideration of abstract languages is however that eventually their code can be automatically compiled and optimized in such a way that the resulting machine code will actually run just as fast or faster than manually tweaked low-level code. When this point is reached it would mean a decoupling of technology from logic because now, any valid formulation in any programming language imaginable would lead to the same result and the technical, more difficult to learn platforms would disappear.

2.3 Programming

Our foremost goal here should be to examine the mentality of the programmer, to get a notion of how they think and why they might be doing what they are doing. Programming languages and "programming" are their means of solving problems and achieving unforeseen things and therefore we first have to deal with the nature of programming itself. In general, programming could be called the embodiment of the human urge to imagine. There is currently no system (except for the universe itself maybe) with a similar capability to create what has not been before. Of course, the beautiful difference between the computer and the universe (in our opinion at least) is that we can control it and create the things **we** like. The fact that what the computer creates isn't actually "real" but virtual, ironically is irrelevant from an experiential point of view because the brain makes it real anyway. This may be one of the reasons an examination of software is really necessary. The following shall provide a brief insight in what the basis of this imagination "virtualization" system is.

The process of creating instructions for a mechanical or electronic machine to follow is called programming. This can happen through configuration of mechanical parts, hard wiring processing circuits, authoring alphanumeric instruction files or inside visual programming environments.[13]

[13]There might also be other ways of programming which the author is not aware of but

All of them have very distinct qualities for the programmer as well as the machine and involve very different ways of working. Since the first two are not immediately relevant for the topics of software and animation, and since I am not particularly familiar with them either, I shall continue to explain the basic principles of alphanumeric and visual programming. Both of them are important for artists and programmers, though visual programming has not gained much traction in professional software development so far.

Note that this discussion deals specifically and only with the programming of electronic digital computers as they are used today in many shapes and which John von Neumann created the foundation for. Also, this is not supposed to be an account or summary of any sort of the enormous field of computer science or the circumstances of software production, but rather its purpose is to convey a general idea of what it is like to program to those who have not done this before.

Alphanumeric program texts (or codes, as they are called) consist of commands[14] that instruct the computer to perform certain actions. This way of programming has been used ever since the first digital computers in the 1940s and is still predominant. It is based on the physical structure of the computer which consists of a CPU capable of very basic operations regarding mainly calculation (like addition and multiplication), data transfer (reading and writing chunks of data) and control structures. The latter make "decisions" depending on the data available to the computer possible. Of course, the computer does not decide—the programmer does—which means that it is the programmer's duty to account for every possible case before "runtime" (or "real-life") data is even available.

2.3.1 Programming Languages

The categorization of certain artifacts in informatics, which actually all have the purpose of instructing and controlling the computer, into "programming languages", "protocols" and "file formats", is in fact arbitrary. The terms, again, only serve us to infer their mode, area and scope of application, which indicates that their actual difference is the size of the window they open upon the functionality of the underlying structures [4, p. 170].

Although all Turing-complete programming languages[15] are functionally equivalent, each language is designed for specific problem sets and thus encourages distinct modes of thought [4, p. 170]. Evidently, the structure of programming languages must be highly dependent on the types of problems that are supposed to be solved with them. They are the result of the negotiation between technical considerations, as not to obstruct the work of the

they are not really relevant here.

[14]The term "command" is used here in the sense of any written symbol or combination of symbols that give the computer a definite instruction.

[15]For Turing-completeness see [52].

machine unnecessarily, and user preference, for the sake of learning curves and work efficiency [4, p. 169]. Programming languages are also designed specifically to avoid mistakes by the programmer, enforcing constraints and proven working habits on them i.e. by means of syntax.

The enormous number of existing programming languages may serve as evidence of the countless different problems and preferences that are to be satisfied. It is the contemporary, accepted understanding that no single language can fulfill the requirements of all problem sets as well as the needs of all programmers. Especially when it comes to "high-level" and "low-level" programming languages, no consent can be reached because both target largely different areas of application. Which language to use is, as with human languages, a question of availability and liking.

To clear the terms up, with low level programming languages we have to work very closely to the hardware of the computer, that is to say, we take direct control of the actual features of the hardware parts. This, however, requires ways of thinking that are very far from how people are used to think about problems. First, great efforts are necessary to accustom oneself to them and second, it makes it more difficult to think about a high-level problem[16] one wants to solve if one has to also think about pointers or registers. These technicalities also cause lower level languages like C or Assembly to have an increased amount of code the programmer has to handle for the same implemented logic. And a lot of code means more chances of making mistakes. These languages give a lot of power to the programmer but also force them to do many things themselves that high-level languages would take care of [53].

On the top of the list of popular high-level programming languages are the likes of Java, C#, Python and JavaScript. They are also very different from each other but newer developments seem to indicate that languages supporting multiple paradigms are on the rise.

Programming paradigms are schools of thought that describe how instructions are formulated, how data is structured and, as a consequence, how the programmer has to think. Functional languages, for example, consider functions (procedures) as a kind of data and are thus much more dynamic because they don't depend on rigid, inalterable processes [54].

2.3.2 Programmers and Programming

In general, the nature of software is to systematically constrain the totality of functionalities of a Turing-complete language, embodied by the hardware of the universal machine[17] the digital computer is, to a subset of operations purposed for specific sets of tasks. The application's user interface provides access and control only over these operations, abstracting the overwhelming

[16] Like how an interface should adapt to the user's needs.

[17] For the universal machine, see http://en.wikipedia.org/wiki/Universal_Turing_machine.

mass of possibilities of the programming language down to a graphical or alphanumeric control language, fitted to the nature and requirements of the tasks and the preferences of the human user [5, p. 149f.]. This can be compared to a window opening the view onto a huge landscape. Sometimes one narrows the perspective by focusing on a specific part and sometimes one feels confined and wishes to broaden the view.

Against this stands the technical reality that programming languages are very rigid constructs with fixed and inflexible grammars and specific requirements. Problem solving involves thinking on several levels, or *levels of abstraction* as I would say. The way to solving a problem is like a branching structure, sort of like a fractal, so to say. One starts with thinking about a problem, dissects it, splits it up into smaller pieces: sub-problems. These, again, are split up into a number of lower-level problems, and so on. This is not to say that any of these sub-problems are actually easier to solve than higher-level problems—on the contrary, often they are more challenging—but they need to be solved before moving up. Moving upwards from a set of lower-level problems means abstracting processes and they can be thought of as a simple symbol, a function.

To conclude this notion with a metaphor: In reality, usually ready-made components (bricks) are used and assembled by specialists (masons). But if one wishes to build their house by themselves they must learn everything necessary for that. In comparison to house-building, software development is not at all standardized for a lot of reasons, three of which being that there are different areas for which software is made, that people prefer different ways of thinking and working (to repeat myself) and that the technologies are still evolving rapidly. The first is certainly true for house-building as well, but the second is normally ignored in that domain, at least for the "less abstract" tasks. Unlike the architect who does generally not lay bricks, the artist programmer writes their code themselves and thereby deeply engages with the substance of their tools and media. This, however, blocks these mental resources for higher-level considerations like deep and elaborate messages.

Personally, I would argue that this switching through the abstraction levels can also be an obstacle to getting into the experience of *Flow*[18]—especially when unforeseen lower-level problems emerge whilst working on a higher-level problem.

[18]For the theory of "Flow", see Mihaly Csikszentmihalyi's work [6].

Chapter 3

User Interfaces

The main purposes of the user interface are to provide information and to give feedback to the user. Information can be given regarding the state of the program and the data it is working on, and the options (tools and commands) available. In addition to this information of a more static kind, the user desires immediate feedback to their actions in order to be able to react accordingly. Ideally what the user perceives in the interface would, as in the physical world, be directly manipulable by hand and immediately changing. Since this is not currently possible, the devices majorly in use are the ones designed to best fit the human needs. This does mean, however, that anyone using these interaction devices with a computer will inevitable be shaped by their physical and logical nature.

For animation visual input and feedback are crucial and although animation is possible by simple text input (see JavaScript animation frameworks like Greensock[1]), a Graphical User Interface can give far more control and precision in achieving what one has in mind.

3.1 Principles of Interaction

3.1.1 Conceptual Model and System Image

Don Norman uses two significant terms in order to describe what happens when the user meets a tool or a device. The information that is available to the user from what they see, hear or from what they have experienced with other systems is called the *system image* [20, p. 31f.]. This is what the user is confronted with when they want to use a tool. It is then up to the user to make sense of this information and to use it in order to operate the tool. The understanding which the user has formed from the system image and the knowledge they have gained from their experiences are called the *conceptual model* because they do not describe the actual workings of the

[1] https://greensock.com/

system but rather how the user imagines it to work [20, p. 25ff.].

The above mentioned principles of interaction with objects are essential to the communication between designer and user. Here, the idea outlined in the introductory Section 1.2, that tool maker and user are connected through the tool itself, comes into play. When the designer's conceptual model, which is probably built even before the tool itself, does not align with the user's conceptual model, it strongly suggests problems with the design of the tool. Since the designer and the user usually come from different perspectives, it is often very difficult to create the tool in a way that conforms to both the ideas and requirements of the designer or programmer and those of the user.

This issue is not limited to user interfaces but also applies to programming environments and languages. Although programming is already considered a complex skill to acquire and accordingly a lot of effort to learn it is anticipated, even further obstacles are created by bad design of syntax, language features and application programming interfaces. Oftentimes, certain functionalities are either expected to exist or to work in a particular way but turn out to be missing, misleading or to work in a completely different manner than considered sensible by the programmer.

As best usability is provided when a system meets all the expectations of the user, some general goals of any system design should be to ensure *predictability*, *consistency*, *meaningful feedback* and *reliability*. Obstacles in achieving this might be missing constraints, missing conventions or bad communication design.

3.1.2 Conventions and Standardization

One of the problems of the computer industry is the fast pace at which new technology is developed and put on the market. The competitors aim for earliest possible release dates and new and innovative solutions. When there are several thinkable solutions to a particular problem (there always are), multiple different systems will be developed and sold by the players in the industry. A good example of the issue that follows this behavior is the diversity of 3D packages available. One must only look at the variety of software products made by Autodesk and the competing applications like *Cinema4D*[2], *Modo*[3], *Blender*[4], *zBrush*[5]. It becomes clear that very different ways and strategies can be found to work with objects in 3D space. The annoying or even frustrating part appears when several of these programs are to be used at the same time because almost no standardization whatsoever has been reached in order to simplify transition between them. The most obvious example for this is the navigation in the viewport, for which

[2] http://www.maxon.net/en/products/cinema-4d-studio/
[3] https://www.thefoundry.co.uk/products/modo/
[4] https://www.blender.org/
[5] http://pixologic.com/

every application comes up with its own flavor. Like with other technologies like cars, high-definition video, web technologies and so on, agreements for standardization will be made for a large number of aspects. Best practices identified by leading professionals in the field, as well as general usability considerations should be the base of this possibly lengthy process [20, p. 248ff.].

3.1.3 Input

As beings with a focus on our visual senses we prefer visual feedback to our actions. While we cannot yet (sufficiently precisely) control the computer with our eyes, the direct feedback we get e.g. from moving the mouse and thereby the mouse cursor provides us with a qualitative feeling of manipulation. Despite its popularity and seemingly effortless handling there are repercussions to the usage of the mouse, physical as well as psychological. For example, the mouse constrains our radius of action and possibly our distance from the computer. It requires that we watch the screen closely and have to be aware of the changing position of the cursor. When we want to trigger an action we must turn our focus to the area where we want to navigate and it takes precision and time to reach the target. The mouse cursor is limited to a single position, so it is impossible to work in multiple areas at once or to activate multiple items simultaneously. There is no agreement as to how many buttons the mouse should have; a single button is certainly simpler but with the growing amount of operations we want to perform (and quickly) it may be helpful to have more options directly at hand on the mouse.[6]

3.1.4 Feedback

After input has been given to a system to perform a certain action, the user will expect the system to carry out that action. It is a common scenario in media applications that an operation is triggered by shortcut or through the menu but no results can be observed right away. If the user cannot verify that the input has been received and that the intended effect has been achieved, they might assume malfunction of the system or an error on their own side. In order to prevent such frustration, information about the state of the system must be provided as well as feedback about the reception of input. Especially in applications with a graphical interface it happens at times that disturbingly long delays occur until feedback is given. In some cases this can lead to wrong judgments of to which action a certain effect can

[6] Maya uses the middle mouse button for the dragging of certain items, while Blender uses the right mouse button for selecting and the left for placing the “3D cursor”[55]. The pop-up menu triggered by the right mouse button in most operating systems appears at the position of the cursor to be easily accessible.

be attributed. Too much feedback, on the other hand, is to be avoided as well because it has the same effect as noise, that is to drown any distinguishable signal in an overload of information. Without useful feedback, no experience and expectations of the system can be formed and no mastery of a tool can be accomplished [20, p. 23ff.].

3.2 The GUI and Invisibility

The principal job of the Graphical User Interface is to make things visible. As mentioned, it visualizes data, states, available commands and options and perhaps also gives hints and tips at what can be done or done better. But another thing the GUI is good at is *hiding* the complex system behind it.

While the GUI can be great for achieving particular results it has been designed for, it also prevents the user from doing other things. This, however, is not perceived as a constraint as such but rather becomes apparent only when the user forms intentions that do not comply with the designer's original concept of the work process.

For particular decisions or tasks the user wants to perform, knowledge might be required that is not available through the GUI. In some cases, this information can be obtained from manuals or documentations of scripting interfaces and APIs, but for very large applications, these tend to be incomplete or difficult to comprehend. Without proper understanding of the underlying logic and workings of the system, qualified decisions cannot be taken.

3.3 Elements of the GUI

3.3.1 Windows

The window metaphor is very convenient because it allows a variety of analogies with software mechanisms and, not by chance, it has been used ever since the invention of the GUI at Xerox PARC in the early 1970s. In software, windows are built for flexibility. According to Steven Johnson they permit access to the larger information space behind the interface that is otherwise inaccessible to the human senses. Windows in file explorers provide features to organize and order the data that is stored on the computer's hard-disk in a chaotic way, which means they should be adaptable to basically any preference a user might have regarding the presentation of folder structures [10, p. 91ff.].

They are also means of separating different applications and logical units from each other. In this capacity they are especially useful because spatial structures (the "layout") are essential for our ability to remember and

quickly recall where we can find a certain tool or a piece of information. Johnson, however, suggests that the frequent moving-around, which windows encourage, makes efficient use of our spatial memory impossible [10, p. 91f.].

3.3.2 Buttons

The two established modes of manipulation—the tool to "work on" data allows a more or less intuitive real-time interaction, and the operation which performs changes according to previously defined specifications—both require some sort of trigger to initiate their activity. The button, the prevalent means of achieving this, is usually a rectangular (or rarely circular) area which is emphasized and indicated as an interactive element by a visible border, sometimes additional decoration, and an icon or a text pointing at its purpose or meaning to the user. While in the real world the artist or craftsman has to physically pick up a tool and bring it in contact with the medium (which in turn resists its manipulation) or at least pick up the medium, insert it into an automated machine and observe the medium being manipulated, the virtual medium requires no such action and provides no comparable feedback [21, p. 31f.].

As Don Norman stresses, affordances, the properties of an object that help us understand what it might be used for, are an essential factor in the learning and recognition of tools. The software button, however, in the place often not even indicating its constraints—not to be dragged or written into for example—fails to effectively communicate its meaning through its only relevant functionality, to be clicked. Ultimately, an icon or text label—tiny pieces of visual communication—in combination with the most generic interaction with the computer possibly conceivable, the click, is supposed to guide the artist in their choice of tool [20, p. 11].

Although there exists a variety of mechanisms in the computer to "trigger" processes, the deliberate action of the user necessary to activate it is what distinguishes the tool as something that explicitly exists "in service" of the user and not in their non-observance or even to their detriment. Therefore the button, in its seeming visual and functional simplicity, spites the user's belief to be in control through its feigned non-ambiguity and powerlessness as part of the larger auto-mated (self-acting) system. It might not yet be working **for** itself, but it has already been abstracted from us—pulled away from our control—which could be even more concerning [21, p. 34].

3.3.3 Shortcuts

The open-source 3D package *Blender* is a prime example of an expert-oriented software application. Most tools and operations are intended to be entered or invoked by using *shortcuts* on the keyboard (see Blender's ex-

tensive shortcut editor in figure 3.1). While beginners need a lot of feedback and visible options in order to explore the complex system, experts, who are familiar with it and have the necessary information in their head, require the utmost efficiency in using the available tools. Keyboard shortcuts can be utterly frustrating for amateurs when they are pressed by accident and possibly cause a ripple through the program which might be hard to revert without the appropriate knowledge. However, for experienced users it is important to be able to define custom shortcuts that are setup according to their workflow and easy for them to remember.

3.3.4 Hotboxes, Pie Menus and Ribbons

Context sensitive menus that are dependent on the state of the system or the location or object clicked on are very useful for effective navigation because large menu structures are cumbersome to go through repeatedly and certain systems possess such a large number of commands that shortcuts for each are exceedingly difficult to find and remember.

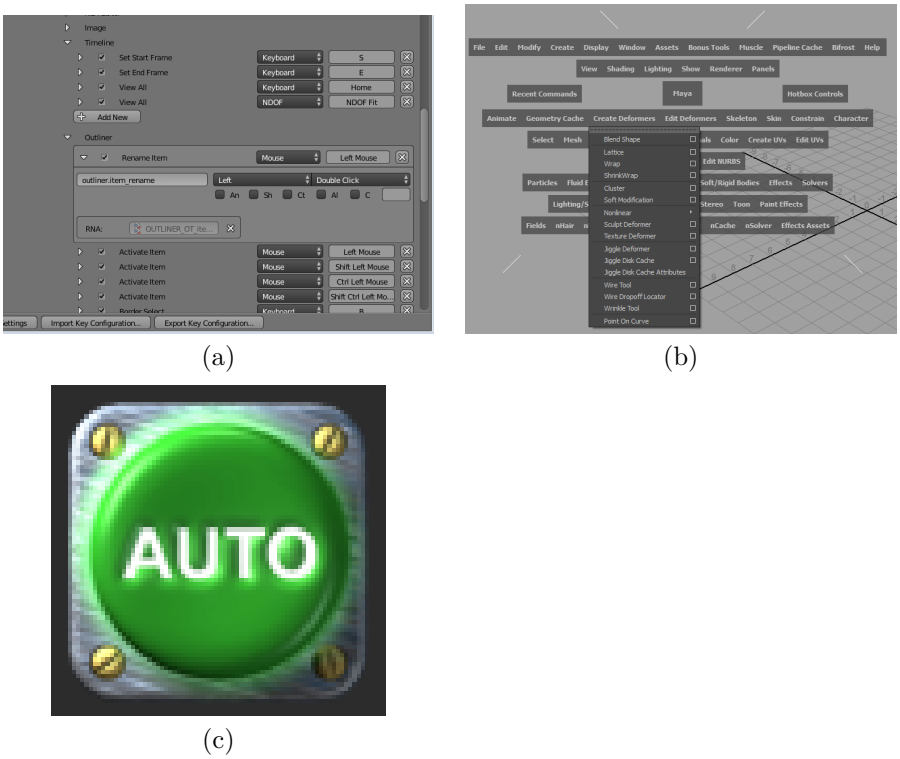
Again, different concepts serve different purposes. While the pie menu is suited for a small number of options which are arranged in a circular form around the cursor (or touch) position, the *Maya* hotbox 3.1 accommodates all menu options available in the application and supersedes the top menu bar completely, which saves screen space. Context-sensitive circular menus enable more intuitive changing of tools if the location of particular options is fixed and can be remembered. That way, the user need not read button labels all the time and can quickly switch back and forth between tools and modes which might not have shortcuts assigned.

Ribbons, as available in *3ds Max* and Microsoft products like *Word* or *Excel*, and Maya "Shelves" 3.2 are another way of providing mode-dependent information or interface elements. Shelves provide quick access to common features useful when working in a specific "mode" or workflow stage. The idea is that every workflow can be divided into smaller parts, each of which requires a specific set of tools to work with. Since shelves are customizable, they can be adapted to suit any possible requirements.

3.4 Limitation

The principle of reduction and abstraction is a common concept for programming and design. The goal is to reduce an existing base of options and operations down to a number of controls that are easier to work with. In computer software the lowest level is the hardware itself, which layers of software programming are set up upon.[7] Programming the hardware to do what

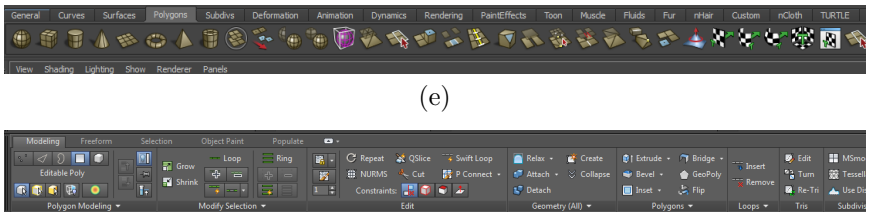
[7] This layering can happen in multiple ways. Sometimes higher-level code invokes faster lower-level code. Often higher-level languages are implemented in lower-level languages.



(a) (b)

(c)

Figure 3.1: *Blender*'s shortcut editor (a) offers detailed options to assign mouse or keyboard shortcuts to any tool or operation of the application. *Maya*'s hotbox (b) overlays all other window content and appears when the space key is pressed. *SynthEyes*' "Auto" button (c) has an almost physical quality to it.



(e)

(f)

Figure 3.2: *Maya*'s shelf and tool menu (e), and *3ds Max*'s ribbon (f) are icon collections providing faster and more contextual access to certain tools than traditional top menus do.

Crucial in the context of human-computer-interaction is not the technical implementation but the fact that different levels of abstraction exist with different strengths and limitations.

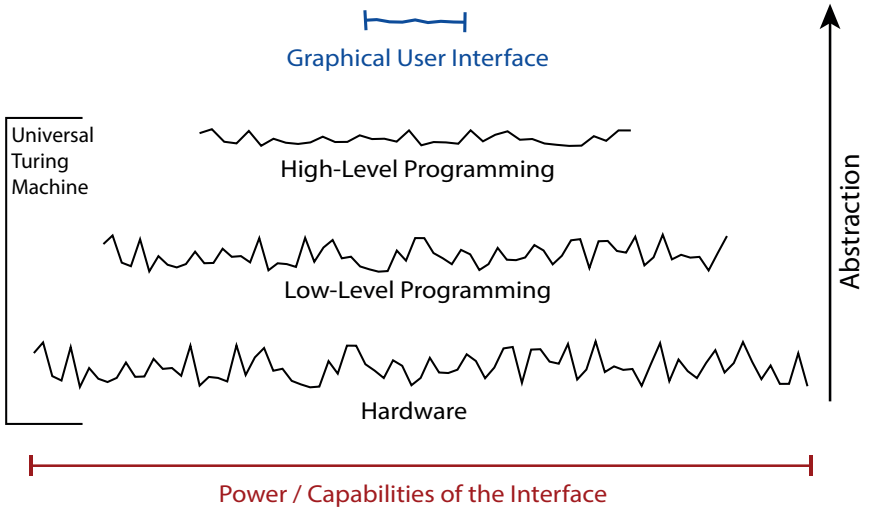


Figure 3.3: While hardware programming and real programming languages provide large sets of possibilities and a lot of flexibility (although abstraction always takes certain capabilities away), description languages like HTML and graphical user interfaces are far more restricted.

the user will be trying to achieve is, of course, not possible, not least because on the hardware level problems like heat arise that are completely different from the issues prevalent in the higher, more abstract, levels.[8] It is, after all, the goal of most computer systems to hide all influences of physics and the "real" from the computer user. Up to a certain point the layers stacked on top of each other are Turing-complete, which means that essentially they are equally powerful. The main reason to use one language instead of the other is that some are designed to make it easier for programmers to build more complex structures as higher-level tools are constructed. The critical point is reached when one layer breaks the chain of Turing-completeness. In most cases this is the graphical user interface as it is common to hide the complexity of the underlying programming from the consumer. The only exceptions are products for developers and professional users who might demand extreme flexibility to be able to extend or alter the application's functionality. Media applications like the major ones from Adobe (Photoshop, Illustrator, After Effects, Premiere), programs for 3D- and 2D-Animation or visual effects all offer scripting capabilities.

Knowledge of technical restrictions, of what is possible and what is hard, is acquired over a longer amount of time and in the course of working, trying

[8] Florian Cramer points out in [68] that data in a computer is actually analog on the lowest, physical level, which is true because voltage has continuous values. The real digitality is produced by transistors which act like switches (1 or 0; current flowing or not) [80].

and failing, with the software. However, high expectations, clear visions of the end-result and low motivation to learn the technology itself can prevent any getting closer of artist and software. The limited expressiveness and consequent informative value of the user interface on the one hand, and the insufficient training in technical thinking of the user on the other hand cause a lot of frustration since the software is a very specialized tool and the artist is usually a user with very particular demands [24, p. 77f.].

But even technically experienced people can get frustrated with a piece of software if the user interface is too restrictive and blocks the user from performing actions they consider logical and necessary. When user interfaces and functionalities are designed, developers take special care not to expose too many options and parameters to the user because these might be confusing to new users and annoying to experts if they get in the way. While to some degree this is desirable, over-compensation can also lead to serious impairments of the user's abilities to perform freely and creatively inside the application.

3.5 Software and the Artist

Since expression and implementation of their ideas are major desires of most artists, every tool the artist uses should be examined in regards to how it affects their expressive abilities. The interplay of tools and people, of purpose and intention, is intense and beyond measure but the basic principles are evident. The tool gives power to the artist and in return the artist gives up a bit of freedom and power by agreeing to the restrictions they put upon him.

The power one can gain by using software should be quite easy to determine by looking at popular software as well as the digital art scene, which has grown considerably in the past years. Both are important because one shows the state of the art of commercial production which has to withstand the enormous pressure of the global market and the other has to keep up with the speed of technological progress in order to remain relevant.

The ways software influences and changes us, on the other hand, are more subtle and creeping. When the artist uses a tool they are exposed to the effects of the various interactive processes between them. While working with the tool, they slowly take in its logic and rules, which we call *learning* [24, p. 78].

3.6 The GUI as a Medium

Tools can be used to produce and manipulate media but are they media themselves? Are they communicating a message to their users? In fact, there are several layers of communication taking place between a software tool and

its user. The first layer tells the user simply *whether to use it* or not. Some tools are inviting and inspiring because of their design and others are not. The second layer tells the user *what to do* with the tool. Software often has the problem that it is very hard to determine from its appearance (GUI) what it is actually good for or built for. The third layer instructs the user *how to use* the software. As tricky as the first two layers may seem, this is the most difficult part of communication with the user [20, p. 11].

While the first layer is trivial and relevant mostly for the economic success of a product, the second is in fact already decisive for what users will attempt to do with it. Factors in this stage are available examples of works done with the software, foremost of course professional pieces and artworks which span from mainstream applications to the more eccentric and innovative, and then the indications inside the software itself. These might be tutorials or presets coming up at first start or labels and button graphics in the user interface.

If a GUI foregrounds certain common or special features, like in an own palette or menu, it suggests to use them and try them out instead of the other ones hidden in the drop-down menus. A GUI built for accessibility like Maya, with tools fairly distributed, appears complex because it has no focus or entry point. It also makes a difference how the tools are grouped in the menus or palettes.

If they are grouped by task (like in Maya 2016) they are likely to be seen as associated with that task for a reason, making usage outside of that task less likely. Someone following a workflow that the predefined tasks do not fit into might not be happy with this default arrangement at all. The commonly used convention in software is to group operations sort of task-based under items like “File” for loading and saving, “Edit” for editing content in the broadest sense, “View” for changing settings of the content views. This seems helpful as a convention although it is entirely arbitrary and unfit for consistency.

Software working with different media or data types might also group tools by what type of content they create or operate on. Blender uses this paradigm for its *Create* tab (“Mesh”, “Curve”, “Lamp”, “Other”), Cinema has a similar grouping in their main *toolbar* and Maya 2016 has *shelves* set up in that way. This kind of configuration is very practical and actually technology oriented but it does not support the more abstract mode of working suggested by the Python-paradigm: that a tool should have a meaning and work according to it and decide its actual operation in context of the given data.

There is probably no way of sorting this problem out that has to do with how people work and think. But it might be a viable solution to provide mechanisms to switch between different modes depending on the current needs.

Chapter 4

Software Media

As pointed out in the introductory Section 1.4, *media*—be they so-called traditional or "new" media—are means of communicating content to people: From one to one, from one to many, from many to one or from many to many. These aspects of media, and new media in particular, and the problems that arise with them, have been discussed exhaustively for the past years. What has not been widely discussed are the characteristics of software as a medium communicating to billions of people all over the world. While it is obvious that communication is enabled by applications like Skype, Facebook, Twitter and YouTube, the more hidden and subtle ways software is impacting us, with its own particular agenda, are mostly overlooked. This daring and provocative presumption of a "software-agenda" is subject of the critical examinations throughout this paper.

The discussion of GUIs as a medium in the previous Section 3.6 should give a brief introduction to the ways software *in fact* acts as a medium. But as software and development become more accessible, the border between tool and medium software becomes blurred. According to Lev Manovich, the terms "digital media" and "new media" don't accurately reflect the fact that the associated media types and their creation are mainly shaped by software [14, p. 149ff.]. It is in fact the software applications used to create and view the digital data behind "new media" that are responsible for most of the properties and affordances these media expose [14, p. 149ff.].

Maybe it would be more sensible to call the kind of new media that are accessed via digital devices, and that are constantly changing and reinventing themselves, *dynamic* to express the fact that behind these media stand evolving technologies shaping the experience they offer. Software media might then be a sub-category describing media that are accessed through screen-based devices like computers, tablets and smart-phones, as they are the homes of the applications that are commonly referred to as "software". *Virtual reality* is a good example to test this formulation. In virtual reality applications the goal is not an interactive screen but full immersion and

immediate interaction. The much deeper connection it allows in comparison to traditional software makes it more "hard" than "soft" and therefore not a software medium as such, but still a very dynamic one.[1]

Lev Manovich proposes a new notion of how media are to be examined. He suggests that particular techniques for the generation, editing and access of content are the core factors of any medium. For digital media in particular, he resolves that their essential components distinguishing them from each other are the data structures and algorithms behind them which are used to create, edit and view content [14, p. 206]. Previously there were hard boundaries between technologies—like the television, the radio or the newspaper—and thereby their associated techniques (of creation, access etc.) were separated [14, p. 206ff.]. With the dawn of software this has changed dramatically and new categories to distinguish digital media from each other must be found.

4.1 Hybrid Media

Hybrid media, as Manovich defines them, are a product of the advancements in software and hardware development since the late 1970s [14, p. 199]. The roots of media hybridity lie in the power of the universal digital computer into whose language the former analog media have been gradually translated. In a general sense, all translation of real-world phenomena into a sort of virtual representation inside a computer can be called a simulation, be it planet trajectories, market behavior or a physically accurate paint brush. Formerly, techniques were implemented in hardware instruments and thereby highly restricted in their applicability by the form of their implementation and the physical medium this tool was fitted for. The transfer of these traditional media techniques into the computer meta-medium practically made their merging and cross-fertilization inevitable [14, p. 199f.].

More specifically, the computer gave the analog media software as a common basis and thereby enabled their fragmentation into smaller elements like primal algorithms and data structures. In particular, every algorithm and data structure can be related to media because, as time has proven, almost every mathematical theoretical corpus can be employed for some kind of communication purpose. It is this collection of algorithms that are the base elements and capital of all digital and hybrid media. They made the combination of previously independent media possible, leading to the emergence of media concepts never seen before [14, p. 176].

But according to Manovich hybrid media is not just the necessary conse-

[1] Of course, I am completely aware of the original meaning of software as a set of computer instructions but in the context of media I use the term to distinguish media with a definite association with a "program run in the background" from media where interaction is superseding this feeling [78].

quence of the universal digital code but also results from the gradual development of technological compatibility. File formats, modules and plug-ins for import and export or network protocols comply with conventions arising from a social context and in turn they make communication and data exchange between software solutions, facilities and individuals possible [14, p. 336f.].

Due to the computer's flexibility, collaborative strategies and modern software development platforms, the creation of a new media "species" can be done within increasingly shorter amounts of time, which explains the incredible velocity at which new media emerge (and vanish). Also, with the mouse and especially with newer technologies like touch-devices, a new factor entered the picture and stirred up the previously simple concept programmers had of software. Interactivity is at the heart of many new-media technologies because it is the immediate interaction with software and the machine that give them the life-emulating qualities people have been dreaming of.

4.2 Interactivity

In general, interactivity is the capability of a system to react to an outside stimulation with a more or less complex response that is usually expected to be somehow related to the given input, to be more or less predictable, that is. Theories of interactivity state different definitions of what should actually be called "interactive" and what is merely "reactive" or non-interactive. For computer systems, which are typically completely predictable, the delimitation of interactivity becomes slightly simpler. The ways a computer can exchange information with the human user and process it are (still) very limited and therefore real interactivity is virtually not yet possible [56].[2]

As already pointed out in Chapter 3, in comparison to command-line interfaces spatial navigation and spatial interfaces not only give the developer more options to let the user control an application but also give the user a lot more freedom in how they can generally interact with it. Therefore, more consideration has to be put into how to restrict this freedom without it being too irritating. Even today, software tends to crash or not respond when something unexpected happens (which always does with human users) and therefore special precautions must be taken to prevent that. While this may be tolerable for programmers who focus on particular use cases, it is a major drawback for people without proper technical background and experience.

With interactivity the computer added another essential feature to the

[2]The concept of (artificial) intelligence would be what we imagine the source of actual interactivity, meaning the complexity of the machine's response matches the complexity of our thinking. In this sense, the naive notion of AI includes interpretative and expressive abilities while wisely excluding the threatening powers of decision and execution.

collection of media techniques which had, for the most part, always been passive in nature. Interactivity opened completely novel possibilities to artists, ranging from websites, apps and games to virtual reality projects and large installations interacting with several people at once. Interaction with an artifact provides to the audience a deeper feeling of involvement with its nature and content, which is especially useful for art and learning environments.

The importance of interactivity in media could lead us to speak now of a fundamental division of media in interactive and non-interactive. Both may have equal rights to the attention of the audience from a cultural or pedagogic point of view but the ongoing advance of interactive technology allows no conclusion about whether there will remain a small area for "traditional media" to exist.

4.3 Popular Software

The network effect: The more members a network has, the more the members profit from it. This simple observation underlines the radical difference of traditional markets where there are limited amounts of products, and new "digital" markets where the amounts are unlimited and therefore the consumption of one person does not directly hurt another. The consequence is that people, in order to gain the most from it, choose the bigger network which supports the forming of monopolies [11, p. 197].

The traditional physical media were quite lucrative for artists as well as publishers and generated reasonable profits, but since the advent of the internet they have been constantly losing market share. They are giving way to new ways of distribution that yield no or minor profits for the artists but large ones for the distributors (platforms like online stores), putting artists under serious pressure [11, p. 175].

By making the access of media content easy and fast, "on-demand" services and streaming support and enforce casual consumption. The mechanism is designed in a way that anything the consumer asks for is delivered to them directly and—at least if they are on modern telecommunication standards—without delay. With this little effort needed, little resistance is evoked against constant consumption in the leisure time or even alongside work. Obviously this reduces the importance of a single piece of consumed content down to a fraction of its original value as it is drowned in a flood of other content and activity [11, p. 176].

For media authoring software this also has an effect as applications with larger communities of users supporting each other and contributing to the overall ecosystem built around the product are more likely to be taken up by newcomers because they promise a faster introduction and problem resolution.

4.4 Software Accessibility

Using software made by a large corporation, or anyone else than oneself for that matter, always entails dependency on that producer. Especially when it comes to the industry where producing in commercial quality is paramount, a small number of common tools dominate the market. While the individual artist potentially has the freedom to work with whatever tool they like, professional artists are very much bound to the software used and available at their workplace, not least for compatibility reasons. That way, cultural differences in production methods and techniques might even be overridden by commercial software products from a specific country, targeted at a very generic audience. Accessibility, therefore, is not only a question of how easy a tool is to use and how easily it can be obtained, but also of which tools are being made, under what circumstances and for what purpose they are made [71].

Cost

Which applications a person can make available to themselves highly depends on their financial and social context. Products developed for professional use are often very expensive, though payment-free student versions may be offered. For artists, however, many of those tools are not available because they cannot afford to buy them, nor are they entitled to student licenses. Since these professional products are very well protected and their usage also not widespread, it is sometimes difficult to acquire an illegal copy of the program as well.

Slowly, free and open-source tools like Blender start to fill in the hole that has been emerging due to the growing interest people are taking in media technologies. At first, when the rise of digital animation began, hardly any software was available to private persons at all and the few tools that existed could be used only by specialists spending most of their time taming them and sending back messages to the programmers, to remove that bug or add that feature.

Usability

Tools that are based on technologies that are only just developing, are almost never built with a focus on usability. That is because the developers' primary interest lies in the tools being functional and working well. Only when the technical and performance problems have been sorted out, further care is taken to improve the users' experience with the tool. Sometimes, the developers just don't know what the users actually need.

In open-source software applications, which are developed by people out of passion, mainly in their free time and without receiving any payment for

their work, this phenomenon can still be observed. Many otherwise powerful tools tragically lack in clear user interfaces and workflow-enhancing features.

Complexity

Another problem, which can be especially severe for artists and students, is the complexity of professional applications. With more and more software products flooding the market, even software manufacturers making tools for professionals have been put under pressure to redesign their applications to be more user-friendly and easier to learn. But until recently, these applications were created mainly with a focus on functionality in mind, making them very hard to learn and understand and also incredibly unpleasant to the eye (which might also have turned one or the other artist away).

Today, professional-level workshops and tutorials (e.g. from lynda[3], gnomon[4], digitaltutors[5]) are available for very reasonable prices, narrowing the gap students have to jump when entering the industry. Thanks to the growing community actively using animation software in all kinds of productions, there is also an increasing number of user-made tutorials on private blogs, computer graphics and animation platforms like cgmeetup[6] or on YouTube and Vimeo as well. Still, since most lessons are intended to facilitate the entry phase into an application, it can be rather tricky to find comprehensible instructions and documentation of tools that are more advanced or rarely used. This goes especially for expert-level features like scripting, which are actually aimed at a more technically skilled audience.

Hardware

Apart from this, another factor is the hardware performance consideration. Professionals and studios spend a considerable amount of money on up-to-date computers because they depend on their software running smoothly. Students and artists, however, usually don't have the means to achieve the same standard and are therefore either completely excluded from the usage of a particularly "hungry" package, or at least limited in the types and sizes of projects they can create with it.

On the other side of this is the quick entry of desktop-level applications into mobile devices. With the touch screens and ever more powerful processing units smartphones and tablets possess, they are now capable of providing capabilities in regards to creating drawn as well as 3D animation. The WebGL standard allows for 3D animation to be done even inside a browser with applications like Clara.io[7].

[3] http://www.lynda.com/
[4] https://www.gnomon.edu
[5] http://www.digitaltutors.com/
[6] http://www.cgmeetup.net/home/
[7] https://clara.io

4.5 Post-Digital

The "shift from semantics to pragmatics" that Florian Cramer diagnoses contemporary "maker movements" with can also be observed in animation [68]. Animation has always had an element of exploration of texture and materiality. Beginning with the abstract animations of Oskar Fischinger, Norman McLaren to the early digital works of James and John Whitney, the adventurous or "innovative" amongst the animators often pushed for the novelty of the visual. Experimenters aside, with digital animation a new mode entered the stage and claimed attention: **procedural animation**. Although not by definition non-narrative, procedural animation is strongly associated with abstract animation and a more liberated concept of story and dramaturgy. Its function changed from minority enthusiast entertainment to mainstream event spectacles and background visuals for music performances. As such, animation became just another language in the canon of multi- and hybrid-media contexts, like mobile apps and web platforms did a little later.

What Post-Digital means overall, as follows from the explanations by Florian Cramer, is that "the digital" (as a synecdoche for all digital technologies and behaviors elicited through the digital revolution) has transitioned from a state of being fascinating and disruptive to being special no more but rather ordinary [68]. The less comforting aspect of this development is that a world or life without digital technology is getting increasingly unimaginable.

Although the Post-Digital is supposedly an attitude encouraging the use of technologies not by default or fashion but by suitability for the specific purpose, trends in animation indicate a growing importance of the digital production methods. Alvaro Gaivoto, an animator originally trained in traditional methods, argues that "3D animation is an evolution of the animation genre" but that "the quality of story and direction has suffered" (see Section A.2). This reflects a notion resulting from the deceiving universality of the computer—the "illusion of more control over the medium" [68]. But despite or because of the greater power and freedom the digital medium indeed gives, the actual control the artist has over it decreases when the general techniques associated with the medium (or rather the super-medium, animation) are ignored or if the technology is not under one's control.

4.6 Power to the Tool

Missing knowledge of techniques and basic properties of a technology or the medium is the common source of loss of control to the tool. While the programmer might understand the capabilities of a tool or a feature as a starting point for others to explore further from, software users perceive them as unchangeable defaults, especially beginners with no overview of the

general possibilities.

Abstraction and automation unknown to the user are the main concepts used to disempower them. By delivering to them a highly automated tool with promised capabilities, seemingly giving them the powers to achieve what previously could be achieved only by professionals, the user's approach to and acting inside or with the tool are limited and under control. Behind this is the user's desire to achieve without effort; but in practice average effort can lead only to average achievement and by definition the average can only ever be "mainstream".

Presets, ready-made assets and templates are not necessarily an attempt to deliver mainstream content to professionals or to un-diversify workflows, rather they can have actual benefits in the beginning of commercial projects. However, if we set dedicated professionals and artists aside, even if customization and scripting mechanisms are provided, most users will not depart from the default (settings) because both good technical understanding and special intentions are preconditions for this. The software medium makes media (and software) literacy more necessary than ever.

Chapter 5

Tools and Operations

As pointed out, software development as well as the animation production are per se evolutionary and innovative processes. Software constantly needs to be updated, enhanced and extended. Even after careful writing, planning and design the work on an animated movie requires frequent evaluation and subtle and often enough radical corrections.

Computer Graphics in particular are still evolving rapidly and it is therefore logical that the tools created for them allow for extreme adaptability. In general, if innovation is a goal, not only should the software tools incorporate innovative technology and research but they also should be flexible enough to **encourage** innovation themselves.

5.1 Analog vs. Digital

When new technologies are developed there is always the question of whether or not it is feasible or reasonable to change the whole existing and functional system over to using that new thing. As outlined in Chapter 6, usually the change happens gradually because a certain level of sophistication and maturity must first be reached in order to make a new technology economically attractive. The beginning of a new technology is often marked by a handful of people seeing great potential in certain mechanisms, and this is exactly what happened in computer graphics as well. When computers first got the capability to work with graphics about 1960[1] traditional animation was already very much matured and a whole industry, led by the Walt Disney Company, had been formed. But from the point where Ed Catmull created his *Computeranimated Hand* in 1972, which could easily and with much less effort have been produced with traditional methods, it took only about 25 years to the creation of the first fully computer animated feature: *Toy Story*.

[1] The first graphical computer screens were used with the TX-0 in 1951 at MIT [23, p. 38f.]. In 1962 Ivan Sutherland developed the first graphical drawing program *Sketchpad* [23, p. 41f.].

To the early CG-pioneers it may not have been entirely obvious that computers would afford qualities impractical for analog animation. Apart from the different ("digital") look of computer-generated images which was not at all polished at the beginning of the CG success history[2], the real potential became clear in 1982 with first complex effects and in 1985 with the first believable integration of a near photo-realistically rendered being into live-action footage [12, p. 9].[3]

5.1.1 Detail

In the 1900s the first animated films were created by J. Stuart Blackton and Émile Cohl and in the 1910s the techniques were further refined with an already sophisticated depiction of character in Winsor McCay's *Gertie the Dinosaur*. What is striking about all animation of that time and most animated films for the rest of the century, is the cartoony style and simplicity. Although painted backgrounds were introduced in the 1930s or even earlier, characters and moving parts are being drastically simplified in hand-drawn animation to the day.

It proved impossible to use the intricate painting techniques developed over hundreds of years, with the goal of realism, on thousands of frames. Furthermore, the effect of the story and overall emotional impact go unharmed when abstracted, "cartoony" styles are chosen. As opposed to the shape and motion of characters, the detail of the imagery has no critical impact on how the story or the characters are perceived; otherwise powerful studios would have gone the extra mile of creating more detail instead of staying with the established style.

5.1.2 Parameterization

Computer animation has a different mode of creation than frame-by-frame animation techniques. Animation software is designed in a way to exploit the computer's innate ability to calculate fast and efficiently, simply lets the animator define what is most important to them and interpolates everything in between. Instead of having to draw every frame individually, the animator sets up the key poses (posing), sets them apart from each other in the timeline (timing) and adjusts the interpolation curves in order to create sensible arcs. A traditional key animator is essentially doing the same but more people are involved in creating the full animation and changing any aspect usually means that everything must be redone. In computer terms,

[2]Take Pixar's first animation short *The Adventures of André and Wally B.* (1984) for example.

[3]In *Star Trek 2–The Wrath of Khan*(1982) fractal geometry generated by Loren Carpenter and particle systems by William Reeves were used to create a planet. For *Young Sherlock Holmes*(1985) a glass knight jumping out of a window was animated, again by ILM [12, p. 9]

the animation in digital is just the product of a limited number of parameters which the animator has to define in order to receive the desired result.

5.1.3 Complexity and Realism

The history of complexity as one of the primary incentives to move over to digital animation begins with the early fractal experiments by Loren Carpenter in the 1970s and leads up to whole cities generated procedurally. Fractals are a key technology for the creation of realistic images because, as Benoît Mandelbrot argues, many shapes and structures in nature are fractal, having repeating and often nesting patterns that is. For example, mountains, trees and textures are frequently generated through fractal algorithms [13, p. 6ff, 25f.].

Particle systems simulate values on a great number of individual data points. For computer animation, these points are usually located in 3D space and forces like gravity, friction and wind are exerted on them. While the primary attributes are usually location and speed, special effects like fire and water can be created by adding simulation of temperature and branching systems where a particle can itself produce more particles. Additional logic can usually be attached to particles, which enables complex behavior like flocking.

Complex phenomena like water and fire, but also whole cities and trees, are very difficult to reproduce in analog animation because they need to be made for each frame and give a continuous impression throughout the sequence. Procedural structures can be generated easily by the computer once the according algorithms have been implemented and even for big simulations of water the complexity achievable through acceptable calculation times is far greater and the results more flexible than with traditional techniques.

5.1.4 Amount and Mass Production

Apart from qualitative (complexity and realism) and productive advantages (duplication, proceduralism, references) the computer also provides benefits when the amounts and distribution of content are concerned. The ease with which digital data can be transferred from artist to studio or from producer to the consumer is also one of the reasons hand-drawn animation is nowadays done on the computer as well (see A.2). The traditional process required masses of paper, expensive animation cels and hours of scanning in the drawings in order to digitize them. With a high-resolution graphic tablet and a few days of practice, most artists are able to reproduce their drawing style directly in the computer. The resulting data is cleaner and easier to compress and use in the further production. Instead of ink and paint, vector lines are used and the areas are filled with color automatically. Disney's CAPS

(Computer Animation Production System) revolutionized this process for their large productions in the 1980s. CAPS maximized the work-load a single animator could do by simplifying processes and automating the redundant ones [57].

As with most technological developments, with every breakthrough in animation technology, the amount of work and the budgets of movies did not decrease despite gains in productivity and efficiency. In contrast to *techniques* which enable the efficient creation of something and are limited in the degree of acceleration because of human limitations, new technologies generate even more goals and optimization to be made. The result is that an even greater amount of content can be produced with the same means. Since creativity cannot (yet) be industrialized and mass produced, particular effort must be made to uphold the quality of storytelling and visual diversity [75].

5.1.5 Accessibility and Distribution

The already mentioned speed of data transfer has more advantages than just to make collaboration of artists around the globe easier. Today, most animated short films are released on online movie platforms like *Vimeo* or *YouTube*. All in all, every movie (short or feature) is available online at some point and therefore accessible to an increasing amount of people all over the world. Student and artist films are commonly put online after their initial reception through animation festivals and thus receive much more audience than they would without distribution on the internet. Since most animated short films are not made with the intention of making profit (which is very difficult), their free availability does not conflict with the makers' interests. The animators are often trying to get into the industry and make use of the public channels which bring their work closer to potential employers.

5.2 Procedural vs. Handmade

What the above discussion points at is that part of the question of analog or digital is actually not about whether a computer is involved but whether the art is hand-made or procedurally generated. While frame-by-frame animation—either with a rig or a brush—is possible on the computer just as painting every texture by hand, the computer offers the power to generate content automatically as well. The properties and strengths of software and computers have been discussed in Chapter 2. They appear again in the context of animation software because they have essential impact on how people work to create animation.

5.2.1 The Thought Process

The approaches taken to produce visual art by hand in analog or digital or procedurally differ significantly as two completely different thought processes are involved. For creating art by hand, artists think in terms of shape, color, composition. Artists need to envision and think about the overall structure of the image from the first drawn line on because every additional line will just add something and correction is often difficult. Especially in analog, one usually plans the composition and certain aspects of the image in advance but once the actual drawing or painting has started, one is working on the artwork itself.

When working in digital, like *Painter*[4], *ArtRage*[5], *Photoshop*, *Krita*[6] (open-source) or comparable painting software, established mechanisms like layers and non-destructive effects give the artist more flexibility to make changes to any part or aspect of the image at any time. The underlying paradigm is to encourage a mode of working in which big structures are put first, because they have the biggest influence on the image, and finer details are added gradually, layer by layer. That way one can change the appearance of foreground elements without damaging the background behind, which is impossible to achieve in analog without a layering technique, for example with animation cels. The digital medium makes the painter more flexible and efficient [9, p. 29].

Procedural generation of an image requires a different way of thinking about nature and the object one wishes to show. Instead of focusing on appearance, the structure and rules behind the objects and phenomena must be analyzed. Since procedural generation is happening in the context of the computer, precise instructions must be defined which the machine can use to create the representation itself. These instructions can then be used to produce a near infinite number of instances and if the procedure is powerful enough, randomness in parameter values can lead to different results for every instance of the generated object. In this context, objects or "instances" can be textures, meshes, complex ecosystems or other digital content; images, text, animation or even speech are possible as well [22, p. 95–113].

While the artist should be aware of the desired end-result in analog construction, this is not necessary for procedural generation. One reason is that rules and procedures can be changed at any time because the computer allows it. The other is that the computer makes it easy to change perspectives on the content and, in 3D software for example, enables the artist to move the camera and view the scene from any perspective. The image composition and many other components of the final image can be decided and easily adjusted later on. These properties are a result of software **variability**.

[4] http://www.painterartist.com/
[5] https://www.artrage.com/
[6] https://krita.org/

5.2.2 The Efficiency of Proceduralism

The efforts and steps necessary to create procedural or hand-made animation are very different. Building a complex system of objects, materials, lights, shaders and more takes a lot of time. Therefore, it is often easier and faster to use traditional techniques to create a single image, because thought must only be given to visual aspects and the artist works on the final image. On the other hand, painting the same image over and over again for every frame, with little variation, takes a lot more time. If a logical structure (like a scenery) is used for more than one frame, it might be faster and less tiresome to build the structure once and to let the computer do the repetitive work. Once the definition of the animation and visuals is established, a whole animated film can be rendered without someone ever having to paint a single frame.

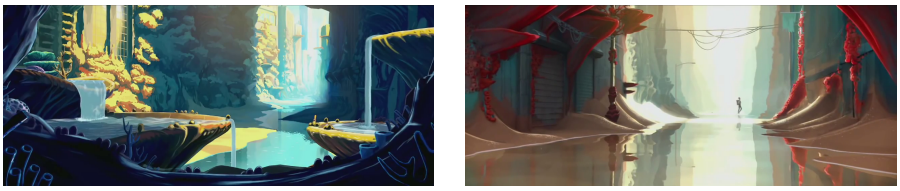
5.2.3 Faking It

Animated films in the style of painting often use painting techniques on media less permanent than actual oil on canvas, like *The Safe House* by Lee Whitmore, for which she painted with oil on a glass pane, with colored paper as a background. But even with this reduced style only between two and forty frames could be produced on a day at certain times, depending on the activities on screen [58, p. 7f.].

Visual Effects: Matte Painting

Since the beginning of the 20th century a technique called "matte painting" had been used, where parts of the image the movie camera recorded were covered with black tape and a painting on glass was created to replace them. Thanks to this technique places could be shown in movies that were either impossible or too expensive to go to or to build as a set. The principle this originally analog technique used was adapted, brought to the digital domain and is now a standard procedure in post-production. Digital matte paintings expand the possibilities of the matte painting by drawing from "illusionist" compositing techniques developed much earlier by filmmakers like Georges Méliès. The workflows involved often combine multiple technologies and are built upon a layering strategy where visual elements from different sources are used to compose a seamless fitting background, sometimes even extending into the foreground. As is accordant with Lev Manovich's theories about the roots and properties of hybrid media, the imagery of which these elements are composed can come from a variety of different production methods [14, p. 281].

The usually cheapest and easiest way to generate imagery for this purpose are stills and video with removed background, e.g., by means of keying or rotoscoping; if shot and prepared in the right manner they can be used



(a) (b)

Figure 5.1: The team of *Contre Temps* used a lot of matte paintings to speed up and simplify the process of creating the elaborate backgrounds they had planned [41].

(a) (b)

Figure 5.2: Cyriak's animated films are based on de-construction and re-reconstruction. He separates visual elements (film, photos or animation) from their context and recombines them to form creatures and even new spaces (a) *Cirrus* (2013)[39] (b) *Cycles* (2010)[42]. The result may have an analog look to it, due to the nature of the used material, and its resemblance to traditional collage, but the complexity of motion and recombination points to the computer's power of abstraction and automation.

to replace a sky, parts of buildings, to insert people or even crowds in a scene. With the same techniques the British After Effects artist *Cyriak* creates animation of a radical fascination and complexity, using the effects of repetition, combination and transformation.[7]

The second main way of producing material to enhance the footage shot on set with is, of course, computer animation. "Set extensions" are commonly used to replace sometimes ridiculously small set structures with computer-generated environments. For that, the set must be measured precisely in terms of scale and lighting and in combination with the data acquired by camera tracking the footage, the physical elements can be perfectly overlaid with computer rendered images of the desired location, designed and built entirely in the computer. Unnecessary to mention that the whole range of animation characteristics—transformations, characters and effects—can be inserted in these ways.

[7]Cyriak's work can be found on http://cyriak.co.uk/index.html.

5.2.4 Realism

Before moving to the criteria of realism, we should discuss whether a 3D scene consisting of hand-made models, textures, lighting and shading should be considered hand-made. This is clearly not the case because the rendered image, which is calculated by the computer through algorithms and simulations, is the end-product, not the scene itself.

A couple of factors are essential in the examination of what makes an image realistic. Questions about the meaning and differentiation of terms like "realism", "photo-realism" and "naturalism" are ignored here and the aspect discussed shall be the quality of animated imagery to appear "real" or indistinguishable from reality. The art-genres called "Photorealism" and "Hyperrealism" are the upper limits of the degree of realism possible with traditional techniques. They have the professed goal of creating art (e.g. images or sculptures) that looks exactly like the "real" originals which it is trying to imitate. Painting a photo-realistic image can take weeks or months and actual references must exist to be copied. It is evident that it would take far too long to create photo-realistic animation with thousands of frames by hand [19, p. 396].

Although an exceeding amount of intricate work is involved in creating a photo-realistic 3D rendered scene and the rendering takes very long because of the physically based lighting algorithms used, this is still a more practical solution than creating every frame by hand. In order to integrate animated imagery with live-action footage, like for most of today's movies where set extensions, characters or whole environments are done in 3D, the perspective and motion of the two components have to match as well. This is almost impossible to do in traditional animation. The movie *Who Framed Roger Rabbit* (1988)[46] mixes live-action with animation, and frequently cameras are not static but moving, which entailed an incredible amount of work. Today, many movies use shaky hand-camera or at least freely moving cameras throughout or in action sequences, and despite these camera movements being a horror for match-moving and rotoscoping, studios doing visual effects seem to cope [37].

5.2.5 Coherence and Abstraction Mismatch

In order for imagery to be visually appealing, it must possess a certain degree of coherence. This is the reason why collage-like animated films where live-action footage is combined with abstract computer generated material, cartoon animation or graphic elements often fall apart visually or have a strange feel to them. The structure of textures used in an image should be roughly similar in terms of detail, organic-ness and color.

While traditional techniques mostly provide a natural feeling to images on their own—due to imperfections in the materials and the artist's

technique—in digital one is constantly fighting against the precision and sterility enforced by the computer. Analog artistic methods usually lead to a good level of coherence by design because the materials involved are the same over the whole piece of work. Still, consistent technique and conscious balancing are necessary [27, p. 325f.].

For computer animation, objects and characters are created one-by-one from scratch and thus, coherence must be actively imposed and checked, especially when several people are working on them. Painters usually develop individual styles early on and learn to imitate other styles properly if trained formally. Good 3D-artists may have a formal training in design and in the technologies of computer animation but their work often contains a technical component that can draw too much focus to itself, leading to a blindness to the relevant qualities of what one is creating.

When we look at Ed Catmull's *A Computer Animated Hand* (1972)[34] we can observe this kind of clean, unnatural feel to the images which persists in amateur animations to the present day. In Pixar's *The Adventures of André and Wally B.* however, this abstract quality has already been overcome to some degree, not least because of John Lasseter's excellent character animation and the effort put in the background details.

5.3 3D Animation Suites: The Technology in the Back

Complexity is one of the severest problems 3D software has nowadays. When 3D animation was first explored in the 1970s and 1980s one had to be a proficient programmer if not a genius in computer science in order to work on and complete a 3D-animated film.[8] Although multiple approaches were taken to make the software more user and artist friendly, in the prevalent 3D applications like *Maya*, *3dsMax*, *Cinema4D*, *Modo* or *Blender* we have huge and very complex programs and user interfaces. Moreover, it is often not enough to have some superficial knowledge of how to perform a certain task because issues will arise that can only be handled with deep knowledge of the system [31]. Some of the properties (and troubles) of 3D applications, resulting from their technology, are outlined in the following:

5.3.1 Operations

While in 2D space most common operations can be performed with a limited number of (old and tested) algorithms and implementations following best

[8] In 1985 four Canadian students (programmers) created the first computer animation exhibiting emotional facial expressions on a person: *Tony de Peltrie* (1985). Considering the means they had at their disposal (mainframes running a self-written animation software called *TAARNA*) and the state of the technologies, their work was awe-inspiring [23, p. 67ff.].

practices, in 3D space the number and complexity of algorithms increases considerably. For this, again, there are many reasons. One of them is that screens and print media are two-dimensional and therefore cannot show a three-dimensional model directly but it must be projected onto a plane (and rendered) in order to be visualized. Another reason is simply that with every dimension a system becomes more complicated because there are more possibilities and degrees of freedom. So, *Quaternions* had to be developed as a representation of rotations, *ray-tracing* is used in many areas from lighting and rendering to physics engines, *texture mapping* is used to project a two-dimensional texture onto a 3D surface. Other concepts that were developed for representation and manipulation in 3D space actually have 2D variants but become a lot more complicated and calculation intensive when used in 3D space, like NURBS, Bézier splines, polygonal meshes or Boolean operations. Many techniques have been invented since the 1970s and many have been rigorously tested and improved, but still, accord might not exist in the industry about which implementations are the fastest and most stable. And often, different techniques are used for similar things but neither is ideal for all use cases, which makes it more difficult for amateurs to choose the right one.

5.3.2 Data Structures

Apart from a plethora of techniques and their implementations there are also different complex data types they have to work on. Three-dimensional shapes are represented in either mathematical form as the result of formulas (NURBS) or as a collection of points connected in a particular order (polygon mesh). Polygonal meshes are complex data structures because they can consist of millions of points (vertices) and many problematic situations can occur that make it very difficult for algorithms to operate on them successfully. Examples of such problems are faces with more than three (or four) vertices, non-planar faces, faces without an area, holes in the geometry, wrong-ordered vertices. During the course of the work on a model many of these can appear; often they are difficult to avoid, if at all, and they can only be resolved with the proper knowledge. Naturally, when one has to deal with such intricate problems, the implementations of operations working with this data contain many mistakes as well because they have to account for a lot of situations and issues, which makes them more vulnerable. Furthermore, these different data and file types are difficult to convert between and for many concepts no standardized formats exist yet. Applications have to support a number of file formats in order to be compatible with each other in one or the other way. Many of these formats are legacy and not up to date with the most recent technologies.

5.3.3 Renderers

Responsible for creating the visual representation of the 3D data (objects) on the screen, renderers are usually integrated into 3D suites as a plug-in and must be compatible with the internal data formats of the software. They add another level of complexity as well because data might have to be prepared for a specific renderer and for many of them, particular arrangements must be made in order for them to work properly. Some demand completely new ways of working and thinking as compared to the renderers built-in or included with the 3D packages. Most of them are focused on a particular style and consequently employ a specific, optimized set of technologies. For example, one might choose V-Ray or Arnold for photo-realistic rendering, Renderman for a "Pixar-Look" or a Vector-renderer for a cartoon look. Currently, it seems like renderers are at times having deeper impact on workflow and visual results than the 3D applications themselves.

Next Limit *Maxwell Render* is built for applications where photorealism is required. As a product for a very specific purpose, its professed goal is to provide short render times as well as maximum usability for the task. Therefore, it is designed exclusively for people with a background or solid knowledge in photography, and not rendering and lighting technology. In order to "speak their language" it has controls based on observable and intuitive parameters. The website also promises predictability and reliability, as one would expect from a realistic renderer [59].

Photorealism is certainly a fortunate case where usability can be enormously improved through technological progress because the expectations of what the result should be are homogeneous throughout developers and artists. Set aside the consequent loss of diversity when it is so easy to create high-quality imagery, this is a positive step, enabling artists to focus on results instead of algorithms. This simplification of the production of the romanticized photorealistic images might even lead to a decline of their importance and, if corresponding tools follow that bring similar usability to software for more liberated creation (of a less specific nature), other visual styles might regain popularity.

5.4 Animation Software: Logic and User Interface

With many available (and necessary) operations a great number of commands must be accessible through the interface. These operations have different meanings, purposes and targets and therefore they should be grouped so the human user can more easily recall where they are. Working with different data types means that even more options must be available to *create* the data, *manipulate* and *view* it. The process of viewing the data should be possible in real-time in order to provide instant feedback about an operation and information about the state of the data and the software system. This

can be challenging because large amounts of data and complex scenes can easily put too much strain on the computer's hardware and make the speeds necessary for professional work impossible.

Most important, however, is constant feedback about the state and current doings of the system. Since operations in 3D-applications can take a very long time it is extremely frustrating for the user when they don't know what the system is processing and why it is taking so long. Sources of the prolongation should be pointed out right away, defective data sets be identified and solutions offered.

5.4.1 Software as a Medium for Creating Animation

For every medium fitting instruments are required to comfortably and efficiently create and manipulate content for it. Most media were invented before the computer and hence devices had already been constructed for them when the computer became a *meta-medium*. These devices, be they video editing systems, darkroom equipment or the multiplane camera, are tailored to the needs of the operator and the properties of the physical medium that is to be manipulated by them. To reach a point where a satisfying experience could be offered took a long time of experimentation but in the end the limitations of the physical devices and media themselves were the main reason basically all media were transferred into the digital domain [14, p. 58f.].

Remainders of these physical devices are still lingering in the GUIs of our media applications, though. Graphical interfaces are usually modular so their elements can be arranged as favored by the user and to provide visual structure. Each of these arrangeable elements (panels) has a specific purpose and controls certain aspects and properties of the content or medium. For example, the Maya *Attribute Editor* controls attributes of objects, the Photoshop *Layer Palette* controls the order and properties of Layers, the ZBrush *Material Palette* controls the properties of materials.

5.4.2 Timeline

Timelines are GUI elements that are used to navigate inside a temporal structure and to indicate the occurrence of certain events over its course. They can be used to represent data or operations that are active or valid only at a particular point in time and to show processes that take place over a period of time. While keyframes, which are used to describe the state of properties at one time, sit on a single point on the timeline (e.g. a particular frame), the activity of layers[9] in *After Effects* is depicted as a bar spanning between a start and an end point on the timeline.

[9]The layer bars are controlling whether the layer is shown and evaluated or not [28].

In most 3D animation programs there are options to control how keyframes are shown on the timeline, to adjust their color and which keyframes should be shown, but there are no ways of creating a more meaningful view telling about properties like visibility, the state of IK/FK switches or color gradients directly in the timeline. It can also be important that the user be able to give input through the timeline. The position and length of transitions from one footage item or animation clip to another can be indicated or a time-range be marked that a particular effect should be operating on.

The appearance and functionality of the timeline depends highly on the kind of data or material that it is supposed to control. Multi-lane timelines like in audio editing software (digital audio workstations, short *DAW*[10]), video editing software (non-linear editing systems, short *NLE* system[11]) or After Effects support and enforce a "directing" kind of workflow because they allow to have an overview of the properties (position, length, etc.) of several lanes (or layers) and to manipulate them in context. So far, animation applications usually do not provide such a kind of overview because they are not designed as a directing tool but just for implementing a preconceived idea in the software medium. It is therefore difficult to work in a mentally more abstract mode where one can think about the synchronization of many objects at the same time.[12]

5.4.3 Layers

Layering is the process of stacking items on top of each other. This is used to combine items that should be kept separate for practial reasons. Whereas with analog media, like photographs, one can simply put one image over the other without one replacing the other completely (physically), when it comes to data the mechanisms involved are not so straightforward. Because data is numerical and exists only virtually as a physically transitory configuration, two data items require both a context and an operation to combine them together and constitute a "layered" state. Specifically, data items usually possess additional information about their state in the context, like position, rotation and size. An algorithm then receives the items as an input and generates the same result for the same input every time. Technically the two data items are used to generate a third item that represents the combination of the two. The original data items are used only to manipulate the result in a way more flexible than if the items were combined irrevocably (the original data be deleted).

Although layering is used in a variety of areas, the easiest to understand use case is images. In the *Photoshop* layer palette, layers carry the data

[10] http://en.wikipedia.org/wiki/Digital_audio_workstation

[11] http://en.wikipedia.org/wiki/Non-linear_editing_system

[12] Compare this to my notions about abstraction in software programming outlined in Sections 2.3.1 and 2.3.2.

while the **layer mode** defines the calculation method by which a layer is combined with the result from the layers below. As color is a rather complex data type with several representations, the number of layer modes available in *Photoshop* is far greater than what is usual for other applications like **animation layers** where basically only one combination (addition) makes sense.

In some cases, like 3D applications where 3D primitives are added to a scene, the combination of items takes place not on the actual data level but happens in a final rendering step, like the projection of 3D objects onto a 2D plane, in this case. Still, scene editors (in Maya the *Outliner*) allow additive, non-destructive, layering workflows.

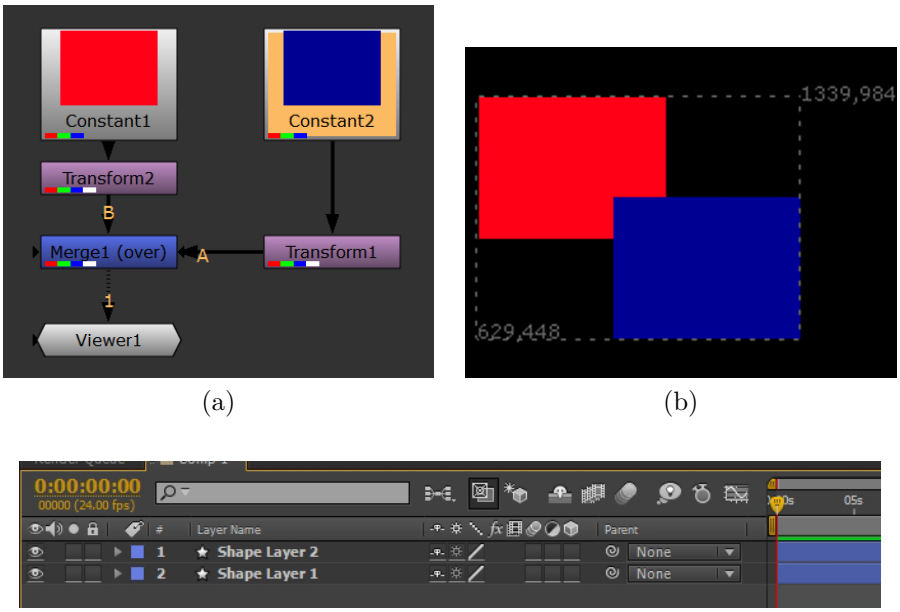
The effect layering workflows have on the user is that they don't see the working area not as a continuous surface anymore to which only permanent changes can be made, but a collection of separate objects that can be set in relation to each other. This is especially relevant when it comes to animation because it makes possible a true decoupling of elements: independent objects can be moved independently without influencing other objects. As Lev Manovich describes, this act of abstraction is a result of the thinking of data and code as modular [15, p. 30f.].

The layer representation, however, has a severe limitation that is its focus on the **combination** of data instead of its **transformation**. This means that common layer GUIs often have inconsistent ways of representing operations because the only operations planned for in this concept are those combining two data sets together. Photoshop, for example, shows filters like *Blur* or *Stylize* as components of individual layers[13] while it shows color transformations as *adjustment layers* which are similar to data layers in all respects except that they cannot hold pixel data. Both *Smart Filter*-components and adjustment layers can be deactivated, activated and have their properties and parameters changed, but they possess different capabilities and limitations. For example, a filter cannot be applied to the result of multiple layers at once and adjustment layers can be applied only to all layers below it but not to an arbitrary set of layers in the palette [33].

5.4.4 Nodes

Nodes are the next step of abstraction following the layer representation. They are individual items that can either hold data or represent an operation that it performs to data it receives. The program holds the data and behaviors in these nodes which have inputs and outputs and can be connected in a usually separate editor. We are here talking about "node networks" as opposed to simple stacks and this implies how much more

[13]This is only the case if they are set to be "smart", meaning they get the ability to be manipulated in a non-destructive manner on the object basis (as opposed to the layer basis).



(a) (b)

(c)

Figure 5.3: The network shown in the *Nuke* node panel (a) and the layers in the *After Effects* timeline (c) both produce the result in (b).

complex and confusing they can be.

Each node receives the data from the preceding node through the input channels, processes it and passes the result on through the output channels. This concept is a lot more flexible because it treats data items in the same way as operations, but as can be seen from an example from Nuke, the functionality of two simple pixel layers in After Effects requires a more verbose representation in Nuke with nodes: five nodes and a Viewer are needed instead of one layer (see Figure 5.3) [32].

There are also differences in how the data contained in the nodes is processed to contribute to the actual media result (the image or 3D scene). In Nuke or VVVV, nodes need to be connected to a *Viewer* or *Renderer* node in order to be processed and added to the image, whereas in Maya it is not always clear which nodes produce output and which don't. This may be partly because these applications put different focus on the node graph. Nuke and VVVV use the node graph as their main working area; so in order to add anything to the project, be it a data item or an operation, a node must be created and connected in the node editor. Maya, on the other hand, due to different established workflows in 3D applications, offers the usage of the node editor as an addition to the other editors of which the *Outliner* is most important for scene organization. Both solutions can be confusing in different ways or make it difficult to manage data and data-flow in the

project.

5.5 Introducing: Maya 2016

The following discussion of the **Autodesk Maya 2016** 3D animation software will be based on several videos provided by Autodesk to give an overview of the new features in this 2016 release.[14]

In each of these videos Daryl Obert, then Technical Marketing Manager for Maya at Autodesk, presents features of Maya 2016 together with another person from the management or development teams of Maya. The changes addressed in [73] are all about how they tried to make Maya faster and easier to use in general but also specifically for beginners with no experience with the software. Some of these features seem to come a little late into an application that should really be all about usability and artist empowerment, especially considering the trends and advancements in user-centered development of the past years. For many years Maya lacked a general direction in the design of the software and features added were mostly technical. However, due to its universal node architecture and the integration of deep interoperability mechanisms like the Maya API and the MEL and Python scripting languages, unlike simpler architectures might have, Maya did not hit the ceiling of its technical capacities. With the 2016 release, it seems, Maya finally arrived in the twenty-first century and includes features like *Editable Motion Trails* which allow editing animation paths in the viewport [73, T=00:44:35], the *In-View Attribute Editor* for faster access to tool parameters [73, T=00:18:25], a *Look Development Environment* inside the Hypershade where material properties can now be edited and effects observed directly [73, T=00:06:20], a new *Pivot Workflow* inspired by *SoftImage* [73, T=00:21:50]. But efforts were also made to amend the outdated *Scene Evaluation* logic which now supports multi-core systems and graphics processors, improving animation playback speeds significantly [79]. For riggers and technical versed people a profiler was added [79, T=00:03:35].[15] Very thoughtful changes made to the Maya user interface are the task-based menu sets that group tools based on what part of the general workflow they are used in [73, T=00:03:10], DPI-dependent scaling of interface elements so they appear the same size even on high-resolution displays [73, T=00:01:20], and the grouping of tools in the shelves by color which makes the identification of shelf buttons and their purpose easier [73, T=00:02:35].

Setting these changes in relation with the factors of interactive systems already discussed in Chapter 3 about user interfaces suggests more immediate feedback, faster and more direct input and better visual organization of

[14]The videos used are the following: Supercharged Animation Performance in Maya 2016, Making Maya Easier To Use.

[15]The game engine *Unity* has had a performance profiler feature for years now.

the user interface. Albeit, even more shortcuts and "quick–" features have been added which still, as the videos indicate, appear to be badly documented or hard to find.[16]

5.5.1 Software and Best Practices

Talking about the efficiency of deformers and supposed "better practices" inside Maya, at [73, T=00:40:24] Maya Animation & Rigging Development Lead Martin De Lasa makes an interesting point: "rather than continuing to encourage people to maybe abuse Maya in ways that, you know, were kind of not consistent with the original design we're trying to encourage people to move towards other kinds of deformers". This expresses two ideas, one being that users indeed **abuse** an application when not working in the ways originally intended by the developers, and also that the developers are actively thinking about how to constrain people in how they can work with the software, or at least about how to make them use predictable workflows.

We should not go as far as to accuse Autodesk or other software manufacturers of deliberately trying to prevent people from using their software in unintended ways (and to make them think in specific ways), but rather it is the essential problem of all interactive software applications so far—applications without intelligence and high-level communication abilities, that is—that certain paths through the system must be secured and recommended. After all, the developer cannot actually foresee what the user is going to do, but they can try and steer them in particular directions.

The concept in question here is that of **best practices**. In the software context best practices are usually sets of instructions and guidelines accumulated by one or many entities that yield better results than other methods [66]. Especially when development and management are concerned, best practices sometimes conflict with common sense or the solution reckoned best at first view. When technology is concerned, like in the example of *deformers* above, results are commonly measured numerically[17] and the most efficient method identified in that way. On the one hand techniques are essential when very specific tasks must be performed with a minimum of time or resources, but on the other hand techniques inhibit the artist from experimenting with less-than-perfect approaches and block their view from encountering new creative options.

The problems outlined above remind of those that persisted for a decade with the **JavaScript** programming language[18], whose concept and design

[16] See [73, T=00:14:48] and [73, T=00:23:25].

[17] Depending on the case the frame-rate, execution time or quality of the result, measured by an algorithms can be used. For complex problems best practices are often based just on statistical analysis.

[18] JavaScript is the first and only major language used on websites and interpreted by web browsers. Due to this availability it is perhaps one of the most used languages in terms of how many people engage with products using it every day.

was badly misunderstood, which was used in amateurish ways and, moreover, implemented differently and sloppily throughout browsers and therefore really hard to use in a consistent and reliable manner by developers at all. From about 2005 on, frameworks were developed that exhibited and enforced a best practice style in JavaScript and with the ever-growing number of powerful tools (which are free and easily available) the Web 2.0 and 3.0 flourished and expanded rapidly in the numbers of users and developers alike. It was only with these new, simpler frameworks and other tools like Yeoman[19], which enable streamlined mechanisms to set up and manage projects, that more people got interested and more creative work could be done on the web platform [61]. In the liberated context of the world wide web it proved a necessity to bring order to the massive number of different software involved by installing and promoting standards that create a basis for more advanced technologies and more democratic access. These standards, however, were established not by a single player but through collaborative processes.

5.5.2 Open-Source

Partly as a result of the open-source trend—catalyzed by the web's induced networking, democratization and globalization trends—by now a number of universal data exchange formats has been developed. Unlike many of the web standards, 3D-related file formats and conventions are mostly originating from corporations like Autodesk or ILM. The FBX file format, which has become one of the most widely used formats for mesh and animation data exchange because of the availability of FBX import and export options in all major 3D software, is still undisclosed and can only be used through a SDK provided by Autodesk [62, 70].

Alembic is an open-source file format that enables the exchange of mesh data between 3D applications as well as renderers and compositing applications. It is not built around an abstract representation of shapes, scene hierarchies or rigs but the common denominator of all 3D software, that is vertices and meshes. Alembic can be used to transfer baked animation from one software to the other. Although it offers a safe, efficient and consistent way of doing this, which is very helpful for visual effects because of varying compatibility with other "high-level" file types, it cannot replace FBX due to its limited scope [64], [8, p. 8].

OpenEXR is an extremely powerful and versatile open-source image format for high-dynamic-range images, developed by ILM. It supports a spectrum of features essential to computer animation and VFX: high precision (each color channel can be saved in a 32-bit floating point format), a number of compression algorithms, Deep Data, multiple channels. Being able to

[19] Yeoman is based on Node.js, a cross-platform runtime environment running JavaScript code [60].

save many channels into a single image file makes sense for rendering and compositing workflows because thousands of frames might accumulate for an animated film and if all render passes were separate, each in its own file, the numbers would multiply. With OpenEXR all connected data stays together in one place [76], [8, p. 8].

Conventions like the data formats above are very valuable to the community and stimulate cooperation and exchange. Every artist and professional may have their own preferred tool to create, edit and distribute content but only if common technologies afford easy **interoperability** can more creative collaboration and constructive cooperation take place on a larger scale.

5.5.3 Learning Curve

There are currently maybe over a dozen professionally usable general-purpose 3D animation applications on the market. If we attempted to oversimplify the situation, we might say that there are essentially two types of applications: the quick and easy type and the powerful and difficult type. Now, in this particular context these terms are actually misleading because they suggest there is software to create good animation with easily at all. But in my opinion there is no software that makes animation really easy and provides an agreeable set of advanced features nonetheless.[20] Rather some applications are only as complicated to learn and use as necessary for basic workflows and others are built for more complexity and flexibility.

On the upper end of this scale, Maya has a very complex architecture and it exposes a great deal of its sophistication on the user interface. The result is that it has a steep learning curve and requires a lot of dedication to learn well but proves extremely powerful and extensible. This becomes clear already from the way Maya manages the modularity and variability of objects in the 3D scene: with nodes.

As discussed, in order to build flexible structures, components which build upon each other are necessary. The logical and simplest representation of such a structure is a layer stack. In 3ds Max so-called modifiers are layered on a stack, each taking the input from the modifier directly below (see Figure 5.5). Just like in Photoshop this has the advantage that it is easy to understand and operate. Maya on the other hand uses the node concept to represent scene objects and connects them by itself when operations are called. Initially before digging deeper into the software, one has no idea of these concepts and might see changes as permanent but as one goes on to encounter errors, tutorials or the manual, these more technical aspects of the application can be engaged with and used to achieve different results.

[20] Coming developments may prove this statement wrong. The French company *Nukeygara* is working on an animation application **Akeytsu** which is very light-weight and built exclusively around the requirements of animators. It also has basic rigging functionalities [29].

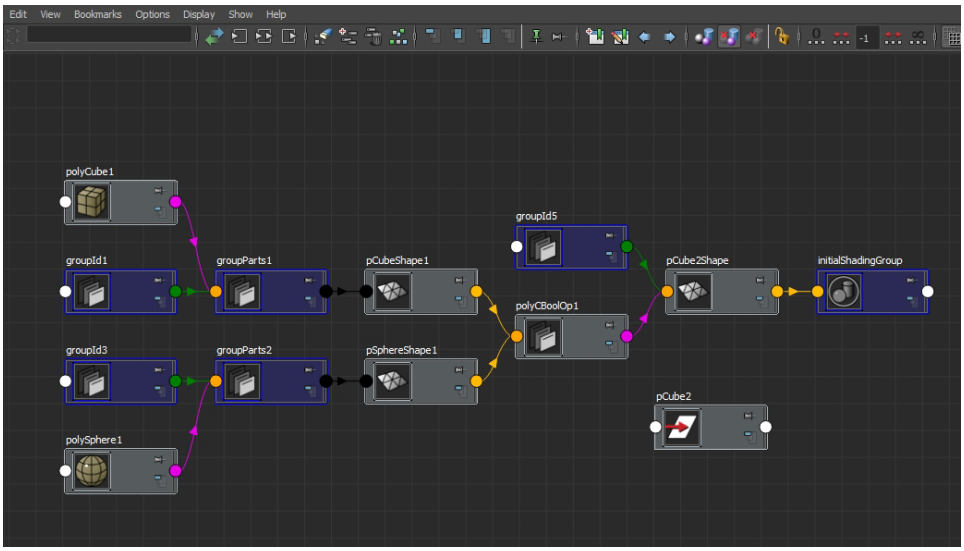


Figure 5.4: The Node Editor panel in *Maya*.

It is important to note that neither representation is better than the other, nodes are simply more powerful and precise but less expressive (in programming jargon) and layers are much easier to understand but have certain limitations in what can be built with them. What all of the current applications for animation (be it 3D or 2D, hand-drawn or vector-based) have in common is the lack of a mechanism to switch through different levels of abstraction quickly and intuitively.[21] Each offers a different kind of representation and tries to stay faithful to their strategy but all have certain obvious limitations that can be got used to only through lengthy training and comprehensive knowledge of the application's features.

[21] *Layers* are here referred to as one and *Node Networks* as the other layer of abstraction.

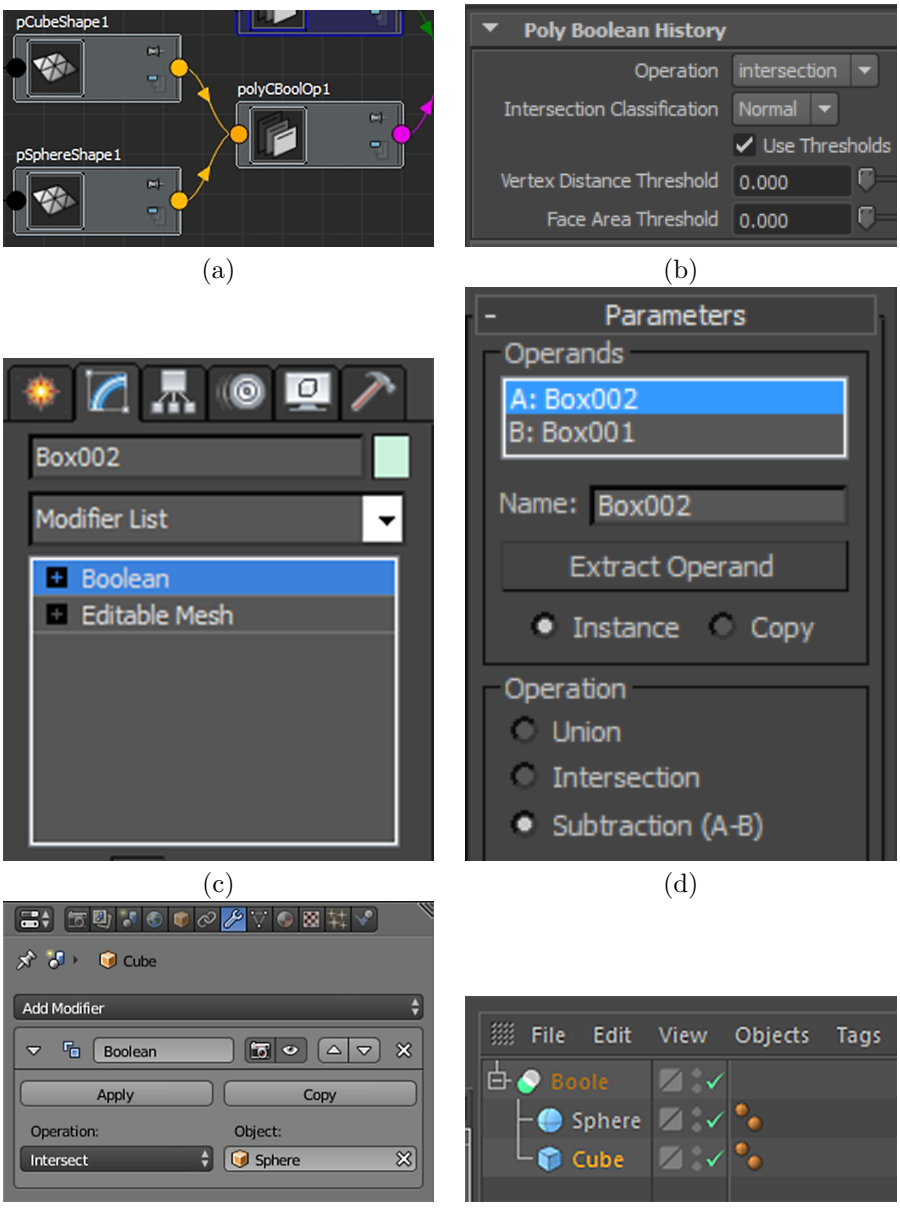


Figure 5.5: These panels show how the components controlling a Boolean operation are represented in various programs; the *polyCBoolOp* node (a) in Maya and its attributes (b); the *Modifier List* with the *Boolean Modifier* in 3ds Max (c) and its parameters (d). In Blender's modifier list (e) each modifier's parameters are integrated into the list item as an expandable form. In Cinema4D the target shapes of the operation are put underneath the *Boole* object (f).

Chapter 6

The Animator

6.1 Digital Workflows

Before asking questions about the potential or actual problems with digital or software-based production, one should also think about its immediate advantages. There are two main properties of software that make it especially efficient and convenient to use as a tool, in comparison to analog media: the universality and compactness of data. With compactness I mean that data is extremely portable and can be transferred all over the world with unrivaled speed. Universality means that data can be very compatible because simple programs or exporters can be written that bring data structures into specific formats, where all, the information about the format, the exporter and the data itself can be easily exchanged. Also, the internet makes it easier to negotiate a common data format.

Quicker and easier *collaboration* is certainly a plus of software, as tools exist to facilitate the organization and *exchange* of production data. Media like movies, texts or music can be *distributed* cheaper and more efficiently through the internet and common data formats enable that everyone can consume them without any further requirements than a computer or smartphone. Digital media are usually *cheaper* to produce because no physical tools and materials are needed. However, the increase in performance and the amount of research required for big projects generally bring the budget up to level, as in software one can always push further if the means are available. Another important convenience is that a plethora of tools, algorithms and *possibilities* of data manipulation (in the positive sense) exist and that many people spend time developing new ones, due to the ease of access to software as opposed to hardware tools.

Apart from these advantages associated with the general properties of software there is also the arguable aspect that it is commonly regarded faster and easier to learn to do something with software than with traditional media. Automation is hard to do for analog media but is one of the core virtues

of software, so it is no wonder that with presets, auto-features and image filters even the beginner can quickly put together something agreeable and sometimes even commercially successful. Unfortunately, automation seduces us to let the computer do the work, and to slide into a mode of combination, repetition and adaptation where the necessity for a vision is dropped and the software predetermines the outcome with its powerful, convenient but standardized settings.

Data Flow

In the context of or when set in relation to software, the concept of "workflow" receives a whole new connotation than what was common before the information age. Data in computers essentially has the ability to flow freely between different hardware systems and software programs. If not obstructed by hardware or software mechanisms designed for this, due to its universal binary code, data has no limits as to where or how fast it can move around, except the speed and bandwidth of the infrastructure. Hence, unlike the painting which takes a long time to be transported from the artist to the consumer, the digital image can be transferred almost instantly. Although it seems, digitality has solved a whole set of problems, like the resistance of content to be moved, these problems have simply been shifted to another point in the process. Unlike the painting, which as a physical object has a truly universal structure to all humans, the image data is not accessible to human senses and requires translation by a human-interactable computing device. A painting may have several levels of messages that can be interpreted from it, and just as a human requires certain knowledge and states of mind to extract them, the computer requires certain information in order to extract the universal image data from the image code.

Data Formats

Like cultural conventions, of visual, social or whatever nature, data formats are merely conventions, based on some sort of optimized structure, to facilitate understanding and communication. Cultural conventions, although sometimes seeming arbitrary when one looks at the broader multi-cultural picture, are usually not only agreements constituting the cultural context, but include certain optimizations like practicality and unambiguousness. The structure of data formats, however, is optimized for different aspects, one of which usually being file size. Admittedly, there never is a single, obvious way of encoding a complex structure like a program or a database, but the necessity of saving space is the cause of many frustrations due to one person or the other company feeling obliged to think of an especially shrewd way of compressing data into an even smaller file.

6.2 Technology: From Geeks to Mainstream

The simple distinction between professionals and consumers made in Section 2.1 can be further refined in regards to technology. The following "three phases of adoption" have been developed by David Liddle [18, p. 245ff.]:

- Enthusiast: When a technology is first invented and introduced it will be used only by enthusiasts who do not necessarily care about its usability. They will tinker with it and try to make it work for them [18, p. 246].
- Professional: Once the technology has been developed into a product—by the developer or by the tinkerers themselves in the case of open-source projects—people are going to try to make money with it. If a similar product already exists the existing professionals will be hesitant at first to try out or move to the new one. Otherwise, some of the former enthusiasts will try to found a new industry. Now, the technology must not only produce nice results but it has to perform well in order to enable professionals to make a living and to keep them from moving on to a competing product. Still, professionals will make an effort to make it work for them since their know-how with technology is one of the factors that distinguishes them from others and that is what they are paid for [18, p. 246].
- Consumer: In the final stage the technology must be so easy to use and understand that anyone without massive technical knowledge can use it. For the developers this is the most difficult step because at this point people without a clue about the underlying complexity will challenge every aspect of the implementation and design and must be led to a satisfying experience with the product [18, p. 246].

6.3 Animation: Schools, Studios and Artists

This above scheme is, of course, valid also for animation software. Most certainly animation programs have moved through the enthusiast phase with pioneers like John Whitney from the 1950s to the 1970s [23, p. 26ff.] making early digital animations, to early commercial software in the 1980s and the first industrialized animation productions in studios like *Pixar* and *Industrial Light & Magic*. There definitely are trends indicating that soon it will be possible for animation to be taken up by an even greater number and variety of people.[1] Looking at these structures in society installed to have animation as their focus, a different categorization could be made:

- Learning: Students; they know they have to learn it to get a job.

[1] Consider e.g., *Adobe*'s recent product **Character Animator**: https://helpx.adobe.com/after-effects/how-to/adobe-character-animator.html retrieved October 14, 2015, 13:00.

- Commercial: Production studios and advertising agencies; they just want to get it to work.
- Independent: Artists and amateurs; they have little willingness to cope with technologies or concepts they don't understand—content is king.

These three groups differ a lot from each other as far as their usage of animation software is concerned. The main factors distinguishing them are to what kinds of software they have access and what their motivation is to use it.

6.3.1 Mentalities

One might be tempted to believe that these categories reflect the actual path people go from being students through a career as a professional animator to being an artist, and in some cases this is true because studios are always looking for talented and creative people who, at some point, wish to be independent. This certainly serves them well in their understanding of and working with the technologies. But with the contemporary notion of the term "art" goes along its parting with the associated obligation of mastering a craft (like painting, sculpting or animating). Feeling that bothering with technology and craft are just obstacles to self-expression, more artists now refuse formal education and focus on their convictions, learning bits and pieces of their tools on the way, mostly auto-didactic.

6.3.2 Students

In the 1970s and 1980s, schools with a focus on animation and little later computer animation began to pop up. Since animation studios had always had troubles finding talented animators, these schools specialized in training people to work in the animation industry. The demands on young animators have certainly always been high and to be able to work in a field as commercialized and economically driven as the entertainment industry, students must specialize in a particular area soon after learning the essentials. Practical experience and particularly a strong "show reel" are critical to one's entry into the industry, but with ten thousand applicants on one hundred internship spots, studios like *Pixar* can choose freely among the available talent [1, p. 143]. Moreover, the fiercer the competition gets, the more specialized and the better trained young professionals will need to be.

6.3.3 Training: To Be an Artist or an Animator

Training, however is not the same as education plus experience. Focus on only a few aspects of the production and the resulting animation, its message and the experience it offers, leads to disconnection from the outcome and a decrease of the number of people who influence it. Commercial entertainment

does not have to be experimental—that is the arena of artists and small independent entities—however, since everything distributed and marketed through public media channels targets a substantial amount of people (unlike most art, unfortunately), these media objects need to be examined closely. Frequently, it seems, political and corporate agendas are in one way or the other communicated through moving images, and it is not in the interest of the producers to have employees question their motives.

6.3.4 The Artist and the Animator

The animation artist David O'Reilly approaches software in a very optimistic way and uses its limitations in order to create unique and new animation styles [63]:

> Software has inherent limitations. When your imagination comes into contact with limitations it produces ideas.

As an artist nowadays mostly working with traditional media, Angie Jones—formerly an animator and now teaching Maya at university—has a very different perspective on software and tools in general. For her, just like the oil pigments she is using for her elaborate painting process, software applications are merely a tool for realizing her ideas. It is the story and the feeling that really matter and after having worked in the CG animation industry for a considerable time she does not feel constrained by software in any way but rather enjoys the flexibility of her tools (see Section A.1).

It seems, many experienced artists—especially when coming from traditional media, which require long training and perseverance—tend to focus on content and its quality and therefore see technology as only a minor obstacle on the way. With 3D-animation software becoming publicly available and making the use of the timeline overly easy with functions like automatic "easing", the barrier to produce any kind of animation is lowered considerably. Drawing skills are difficult to acquire but anyone can pose a well-rigged character in software, without much training. To introduce transitions between poses and to make a character perform certain actions is a matter of clicks and a couple hours tinkering. But to learn the principles of animation—posing, timing and spacing—and to be able to produce high-quality and natural motion requires far more time, knowledge and skills. While software makes it easy to start out with something, by replacing complex skills (drawing a whole character in every detail) with simple processes (moving the characters leg where you want it), it does not make the underlying skill sets—observation of people and their behavior, judgment of color, proportion and shape—any easier. If anything, one will be less inclined to learn something difficult after learning something easy because the effort doesn't seem worth it anymore when everything else is easy.

Indeed, through standardized software, the computer allows not only to create animation in industrial standard quality but, as Paul Wells argues "at a industrially determined aesthetic level" [26, p. 50], leading to the situation where "everyone's movies sort of start to look and feel the same" (Don Hertzfeldt).[2]

6.4 User Motivation

Psychologically, there is a variety of reasons to make animated movies, or practice any other kind of art or craft, for that matter. There is also, however, the very obvious construct surrounding the professional industry. Professional animators make movies to earn money and students make movies to learn it, to go on making movies and money. Since the skills required to make animation are very advanced and achieved only through long and hard work most people in the animation industry, just like in games but possibly even more so[3], are enthusiastic about animation itself and therefore highly motivated and dedicated to their work. The high level of abilities reflects in the choice of new employees which puts animation students under pressure to present show reels and portfolios exhibiting state-of-the-art technologies and skills as well. Consequently, the student's focus, whether they want to get into the industry or the art scene, will decide whether they concentrate in their preparation on experimentation and storytelling or on visual complexity and technology.

In the art scene of animation there is a lot of pressure as well, however. Due to either insufficient funding programs or too many artists applying for them, depending on the country, funding for a project is difficult to get and it is therefore difficult for animation artists to make a living off their work. The long time it takes to learn animation software and subsequently also the fundamentals of animation which require practice and live study to master, makes the situation even more severe. The complexity of the software alone can make ambitious future animators shy away and even once one has acquired the necessary skills a single animation project can take a long time to finish and many frustrating mistakes can be made along the way.

[2] Quoted in [26, p. 61].

[3] See the famous example of Pixar: http://thenextweb.com/media/2012/05/21/how-pixars-toy-story-2-was-deleted-twice-once-by-technology-and-again-for-its-own-good/, retrieved October 17, 2015, 15:00.

Chapter 7

Animation and Software Artifacts

The history of computer animation is mainly a history of technological breakthroughs.[1] For motion pictures the invention of the movie camera was already the biggest step forward and further developments were heavily concerned with visual language, editing and acting techniques. The technology of the camera is not a primary factor for the overall effect of the movie. With digital production, computer generated visual effects and set extensions this changed dramatically and full-CG animated films rely on certain technologies to deliver a coherent and compelling impression.

7.1 Animation and the Impossible

In the introductory Section 1.5 about animation it was already pointed at the notion that the domain of animation is the impossible to see. Despite all the definitions of animation, pinning it to techniques, technologies, genre, historical meanings and developments, animation is eventually simply defined through what it is used for and what it can be used for—what its purpose is. If we watch a live-action movie with VFX, what we see is an actor, who is obviously not animated, and a monster, which is animated. Whether the set extension, replacing the movie set with a view on cities and mountains, is animation is probably dependent on the technique used and the type of camera movement. But even if a still image or an inanimate 3D model may not traditionally be seen as animation, I argue they are subject to the same effects and principles.

I am proposing the following list of categories of things that animation is mostly used for (or why it is used in the production of a story). It should not be seen as comprehensive but rather as the starting point of a discussion

[1] https://en.wikipedia.org/wiki/History_of_computer_animation

about the suitability of software for these purposes:

- Impossible structures (e.g. spaceships, alien planets, aliens),
- Invisible structures (e.g. atoms, the inside of the body),
- Impossible transformations (e.g. that of one person in another, of a thing in another, or even the continuous changing of shapes as demonstrated in Foldès' *La Faim* (1974)) and
- Possible structures/transformations for cost (e.g. set extensions)

7.2 Modes of the Digital

Now, by coincidence (or design) software, with the blank page it creates for the building of new tools, has at its core a similar purpose—the liberation of the human from their physical form—and thus is a platform capable of visualizing just about anything imaginable [26, p. 96]. Naturally, software dominates where it comes to realizing the impossible and brings some of its own logic into animation techniques, processes and in the end, style.

7.2.1 Transformation, aka Blending

As mentioned above, transformation is one of the key modes of animation because it usually goes beyond what is possible in reality. But transformation is along with variability a core concept resulting from parameterization, that is, using variables instead of constants to control shapes makes it possible to change the shape of something over time. Several mechanisms based on the principle of blending are in place in software today.

The simplest example would be the **blend modes** in image editing or compositing applications like Photoshop and Nuke. Blend modes define a formula which takes two images, usually one over the other, as an input and gives a single image as a result, blends them together. By putting different weight on the two input images or varying the transparency of the images, a blending effect can be achieved—a transformation of one image in another.

The usual, basic, approach to animation, if we disregard complex and overlapping movements, is to see motion as a gradual change of an objects position from one place to another. While in discrete (e.g. traditional hand-drawn or frame-by-frame) animation motion is seen as a change from one frame to the next, computer animation abstracts the process and lets us define motion by a start and an end point. The computer, then, is able to calculate the object's positions in-between the two, to **interpolate** them. Since this information alone is often not enough to design a motion pleasingly, additional tools like tangents and curves are used to tweak timing and motion path. The great advantage of this approach is that very little definition (in the sense of data, not know-how, of which a considerable amount is required) by the animator is actually needed to create a decent animation.

Also, start and end points are potentially variable, so that the computer can "decide" itself, what paths the objects should follow, which is exceedingly helpful for procedural animation.

Blendshapes are a technique used to accomplish the animation of complex organic (or not organic) objects like faces, which cannot easily be rigged part by part because they consist of many parts enclosed by a single hull. They let the user define several instances of geometry with the same topology (e.g. several duplicated human heads), but different shape (e.g. normal face, smiling face), and blend additively between them. In practice, this means that one can add certain features of the blendshapes to the "normal" state (e.g. the lifted eyebrow from one and the squinted eye from the other blendshape) and thereby not only avoids modeling every single distinct expression (like it is done in stop-motion), but can also interpolate between them continuously. This technique is based on the blending of the vertices between the base mesh and the changed positions on the deformed meshes. Each blendshape has its own weight, controlling the impact it has on the resulting shape. As sometimes hundreds of blendshapes are used on a single face, to enable sophisticated, natural expressions, additional tools are devised to facilitate control and make the usage more intuitive.

7.2.2 Resolution

Numerical values, the ones that are used to interpolate between states, are infinitesimal, at least as far as the user is concerned—the limited precision of decimal numbers in the computer is rarely an issue. The limitations of resolution, however, are frequently disconcerting. As parameterization is limited by the capacities of the human brain (too many parameters destroy the usability), resolution is normally limited by the speed of the computer. Parameters are intended to be an interface for the user to control a machine, whereas the computer itself manipulates a far greater number of parameters (be they pixels or vertices), according to algorithms and their exposed parameters. The term "resolution" means that something consists of smaller parts and implies that we are not interested in what these parts are, as is true for pixels and vertices. It becomes indeed difficult when we are forced to think about these smallest parts nonetheless. Natural shapes and colors are grasped intuitively, our perception cannot be measured, and when we have to deal with their mathematical representations, work can get frustrating.

Actually, the frustrating part seems not to be the development of a virtual representation (like meshes, NURBS, skinning), but the work at the border of technology and art. Technologists are used to solving technical problems and are absorbed in their matter; artists are used to working with intuitive tools to express their thoughts. Of course, this view is a simplistic generalization—indeed both groups need to be really into their work and have a purpose for it at the same time—but still there is a border area be-

tween art and technology where it is difficult to act, as one is not sufficiently immersed in both domains. Nevertheless, it is exactly this border area which proves most inspiring and fruitful because drawing from concepts of both areas leads to unexpected, innovative ideas and solutions [24, p. 79].

7.3 Visual Style

Although the basic materials involved in the creation of analog animation are the same as they have been in traditional arts for hundreds of years, the animation culture enriched the overall stylistic diversity of visual arts dramatically by adding previously considered childish or simplistic styles to the new canon. That this was possible might be due to various reasons, one of which being that animation was not subjected to professional criticism for a long time because of its experimental nature and the fact that it was long not recognized as an art form but seen more as entertainment, which enabled animation artists to work more freely within their own modes of expression and style. Another aspect could be that the images were now put to more explicit use as they told stories that everyone could understand and relate to, as opposed to still images which require more knowledge and time of reflection in order to lead to a satisfying interpretation. They can also be made more entertaining as the attention of the audience—which naturally follows action—can be massaged by the skillful lowering and heightening of tension. It is therefore easier to accept a simplistic style in an animation than in a painting because the latter has to rely on the quality of its static visual appearance only, whereas animation can distract from its appearance with other aspects. In fact, according to Scott McCloud visual abstraction can help the identification of the audience with characters in a story [16, p. 30f.].

The way animated films are made, the styles and approaches used, highly (but not exclusively) depend on which of the groups mentioned in Chapter 6 the makers belong to.

7.3.1 Perfection

The entertainment industry is a particularly competitive one and animation studios must stand up to large production companies with enormous resources. Animated feature movies are generally very successful with the young audience and the number of adults taking interest in animation is growing constantly. Especially if the narrative abides by high standards and offers depth and a critical component the chance to reach a larger audience is very high as compared to entertainment-only comedies or children's series. But still, the main selling points of animated movies seem to be comedy and entertainment as well as visual impressiveness and complexity. This is not to say that realism is the ultimate goal of the so-called "animated feature"

Figure 7.1: In *Big Hero 6* (a, b)[38] and *The Lego Movie* (c, d)[45] huge cities are seen (a and c), often only for a couple of seconds. (b) and (d) show plastic materials, as did *Toy Story* because they could be rendered more believable than organic materials at the time. But now, the lighting and rendering technologies make the translucent surfaces look realistic in ways impossible only years ago [74].

(this is the case for visual effects) but the audience recognizes incoherent levels of abstraction in the image.[2]

Since the image has such an impact on the audience, studios are always struggling to stay on top and to incorporate the most recent technologies into their films. Many of them are actively researching and developing tools and some are even contributing their developments to the public.[3] For a while now, crowds of people, morphing, hair, fur, liquids, trees, particle and rigid-body simulations and cloth have been standard for big productions and they are (or becoming) available to amateurs as well. More recently it has become possible to create large procedurally generated cities, realistic snow, even more realistic lighting models and to control immense masses of individual parts in very complex ways (see the effects in *The Lego Movie* and *Big Hero 6* and Figure 7.1). The reason why these cannot be used by amateurs and small studios for now is that they require enormous computing power which is (despite Moore's law) yet available only to large commercial facilities.

[2]For example, if the behavior of clothes and fabrics is physically accurate it might look strange if the character's floating hair were reduced to a rigid mass; imagine Rapunzel's hair in *Tangled* being a mesh rigged with a simple joint chain.

[3]See OpenSource projects like *Alembic*, *OpenSubdiv*, *OpenEXR*.

7.3.2 Abstraction

Hand-drawn animation has traditionally stimulated creative character design. Not only in artist animated films but also in series and feature films, characters have been given expressive and strongly abstracted shapes. The drawing medium allows that shape per se (as opposed to detail) has no impact on how much work it is to bring a character to life.

In 3D computer animation, on the other side, more restrictions exist regarding the flexibility of characters and their deformations because less tricks can be used to make them look correct. Although shapes are not limited as such, their motion must be more accurate because the camera is positioned in 3D space and often cannot be prevented from seeing mistakes. Incorrect deformation and interpenetration can become an issue if a rendering style is chosen that accentuates mistakes instead of hiding them.

As a result of the procedural design of 3D animation software (parametric parts are defined instead of continuous object-states) more thought has to be given to the design and technical structure of assets, characters and scenes overall.

7.3.3 Faces and Expressiveness

The expressions of the face are an essential part of human communication and they can convey feelings or even thoughts. Since animating details and believable facial expressions is laborious and difficult in traditional animation, in many stop-motion or hand-drawn animations faces are deliberately left abstract and generic. Commercial productions always incorporated more or less detailed facial animation but on the other hand, means were developed to transport emotion on different levels. The body language exhibited in the Italian animated TV series *La Linea* is a great example of how facial animation can be substituted or complemented effectively [40] (see Figure 7.2).

With technologies like *blendshapes*, *muscle rigging* and *facial motion capture*, facial animation has become a standard in 3D animation and it is not regarded a specialty any more. At the same time, the techniques originally designed to make up for the missing face expressions—not only intense body language but also techniques in editing and visual design—are used less and less. Despite the immense possibilities of the computer, mainstream 3D animation is not any more "hybrid" than the traditional Disney cartoon animation.

7.3.4 Transformation and Smear-Frames

Despite the supposed "nativeness" of transformation to software, transformation is not a very simple thing to do in computer animation. Although it is fairly easy to do "tweening", that is to animate a single value over time,

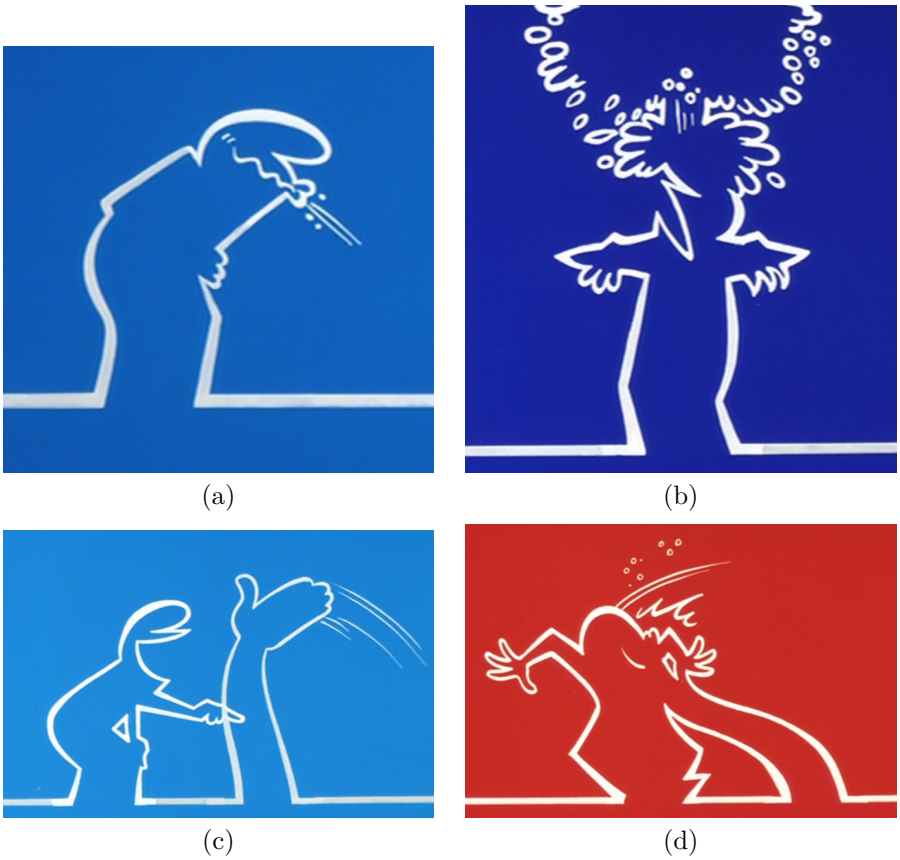


Figure 7.2: With extremely reduced means *La Linea* (1972)[40] manages not only to tell stories but also to evoke empathy.

it is usually difficult to transform complex objects like people or creatures. Non-linear deformers like *squash and stretch* and *bend* give good results for primitive deformations adding to expressiveness but they do not allow what we mean by transformation as in *Hulk* (2003) or even *Transformers* (2007). For effects like these, many, possibly nested, transformations of individual parts must be put together to form a coherent whole. Such effects were mostly simplified or left out in traditional animation, and rarely used in full-CG animation until recently; they are, however, an essential asset for Visual Effects movies.

So-called **smear-frames** are used to make motion appear more fluid and they are based on a simulation or faking of camera artifacts in animation, where motion-blur does not occur. Rather easy to create in hand-drawn animation, they are distinctly hard to create in the computer because creative and controllable solutions are usually more desired than simple simulations of motion-blur. Several categories of smear-frames are used in traditional an-

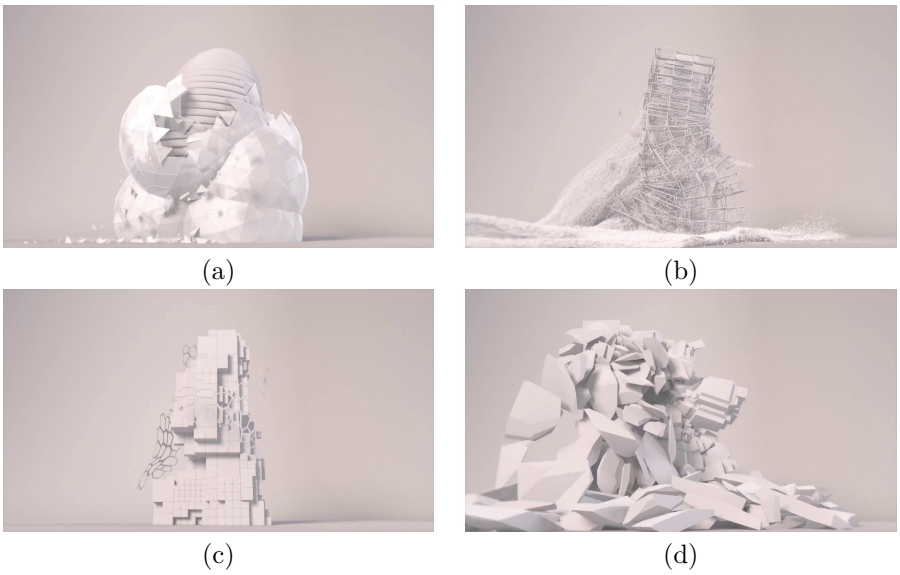


Figure 7.3: *WalkingCity* shows the ever same walking animation in a loop. But while the base motion stays unchanged, the geometry following it transforms constantly. While frequently the old version is substituted by the new one and organically reduces itself (c), sometimes when the walking character is changing, the old geometry is subjected to simulation and left behind like trash (d). The look of this film is dominated by global illumination and fluid (a) and particle effects (b) [43].

imation and some are much more difficult to make in computer animation than others. In general, considering the fact that smear-frames are individual frames which must be crafted basically one-by-one, they are far greater effort to use in digital than in analog animation and they have therefore been mostly ignored, so far.

7.4 Art and Animation After the Digital

Before digital tools were invented to create media and art, the artist stood in direct and usually tactile contact with their medium, be it pencil and paper, chisel and stone. With software comes a medium that allows for the creation of other media that do not stand in direct relation to what the artist uses in order to shape them. After a digital image has been printed on canvas or a movie transferred on film it cannot be connected back to its technical source, were it not for specific characteristics of digitally produced media, for example those pointed out in Section 7.5. The medium for the artist to carve their message into, now, is not necessarily the medium the recipient interacts with to perceive it. With physical media the possibilities

of reshaping a specific medium and its content to fit another one, or even to allow mechanical reproduction, are very limited. Digital data on the other hand is a universal carrier of content and can therefore be remodeled (say, converted) rather freely. Thus, the message the audience receives is dependent mainly on the properties and capabilities of the media decoding it, nowadays mostly software applications like video players, web browsers or games [14, p. 149f.].

7.5 Arising From Constraint

Constraints and thereby challenges are known to produce particular ambition and creativity in us to solve the problems and find ways to realize ideas and visions. Certain ideas and solutions would not have come to being at all without the proper challenge presenting itself. It is the problems and limitations that drive and inspire us.

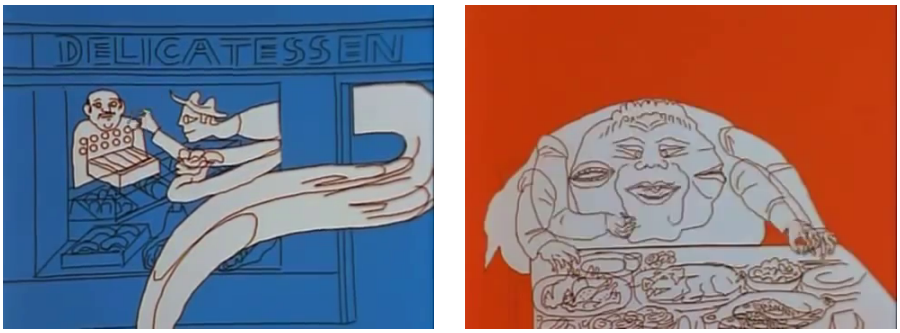
The following are only a few of countless aesthetic styles that have made their way into art and culture, being the result of embraced technological limitations:

Abstraction

Abstraction can be applied to the appearance of beings, motion, behavior, etc., and represent them to a certain degree and under certain circumstances. But abstraction can also be exaggerated and then the image might not be a representation as such but an indication of some thing's existence and acting (as in the case of Memo Akten and Quayola's *Forms*; see Figure 7.7) or the image might trigger a mere association with one or a variety of phenomena. As the "abstraction and automation machine", the computer is the ideal environment for the production of complex representations and motions and lends a whole new mode of working and a new scope to already existing and proven techniques.

In *The Lego Movie* explosions, water and other effects, normally created with millions of tiny particles, are represented with *Lego* bricks and therefore, like the whole movie, have a very abstract appearance. "Analog" effects[4] like this—and the movie has indeed a sort of analog look to it, due to its photorealistic shading (see Figure 7.1)—would not be possible with analog techniques but are a development based both on traditional techniques and the possibilities of the computer.

[4]The term is used here to emphasize that the look of these effects is derived from traditional techniques with a lot of limitations regarding complexity, as opposed to the standard, highly complex digital effects.



(a) (b)

Figure 7.4: *Hunger* (1974) was one of the earliest computer-animated films at all and features objects and characters constantly transforming into each other [44].

Vector Graphics

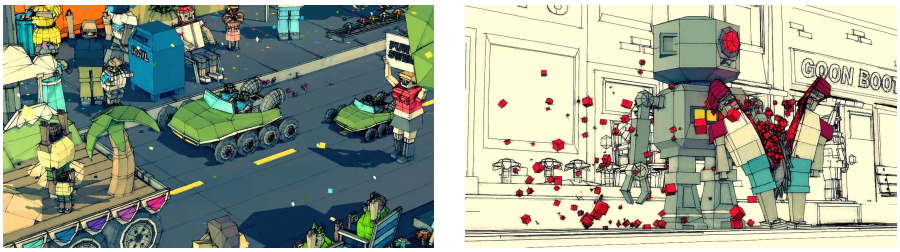
The characteristics of *vector graphics* are their smoothly flowing edges and organically shaped color regions. Vector-based images contrast to hand-drawn images by their perfect line quality which is a result of the (Bézier) splines' continuous interpolation. The probably first vector-animated films were made by Peter Foldés in the 1970s 7.4. These techniques were then used in motion graphics and for advertising and information purposes—advertisements and infomercials.

Low-Poly

The previous limitations of *polygon modeling* (exemplified by the "box-modeling" technique) lead to the popular low-poly look where complex shapes are represented by abstract polygonal objects. A whole culture has been built around this style which is associated with, or even a result of, the restrictions imposed on artists working for game productions, because game engines are very limited in the amount of data they can handle.

One of the consequences of this connection is that the low-poly style is sometimes used to distract from how brutal or macabre the animated scenes are (see Figure 7.5), or to give them a naive undertone emphasizing the story's irony. This could support the hypothesis that abstract visual style also supports more abstract thinking and perception or, as McCloud says: "amplification through simplification" [16, p. 30].

The goal not to create realistic looking characters and objects can be inspiring and also lead to interesting designs and solutions regarding how characters are made to express emotion [47].



(a) (b)

Figure 7.5: The animated short *Flying Lotus—Kill Your Co-Workers* by Mike Winkelmann is made in a minimalist low-poly technique. It receives part of its charm from the analog looking rendering with lots of imperfections and artifacts added on purpose. Particle effects augment the piece through their complexity and intensify its game-like character [47].

Ambient Occlusion

Ambient Occlusion is a technique that is supposed to fake light behavior in locations of a scene that are "less accessible" to light. This is done by measuring how much geometry is surrounding a certain point and darkening the area according to how many directions there are from which light might be coming. The effect is that corners and grooves appear darker, which gives surfaces nice gradients and makes them more natural than if their shading was constant. However, this technique is not an adequate simulation of light, like path tracing, but merely a quick approximation based on observations.

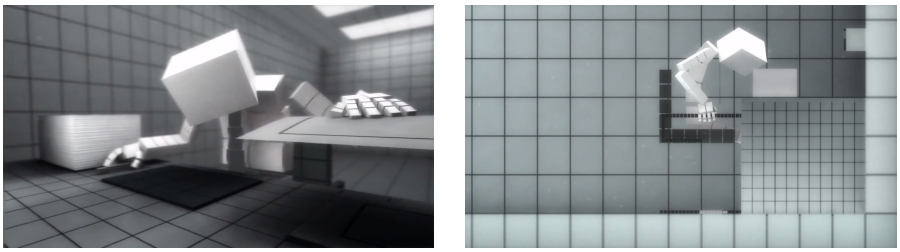
Actual simulations are often difficult to control in artistic ways because they are not based on a design whose purpose is to give control to the user, but built around an understanding of a process. While with simulation the definitive focus lies on getting the most *accurate* result in reasonable time[5], with faked solutions the original intention is to reproduce an effect without knowing or respecting the logic of the physical phenomenon.

Ambient occlusion may be an outdated technique but as a means of creating volume and space without realistic shading, it is still an interesting tool for animators who are not after photo-realistic imagery (see Figure 7.6).

Particle Systems

Early examples of the usage of *particle systems* demonstrate the wish to make effects without having to painstakingly draw them frame by frame, but make evident that having the right mechanism alone is not enough. The imitation of natural effects using large particle systems often requires using

[5]This means the developer is concerned more about the steps of the calculation themselves, and that they are correct and optimized, rather than great flexibility. Scientifically, there is only one true solution to a simulation problem.



(a) (b)

Figure 7.6: Although surfaces in Or Bar-El's *Beat* have almost no texture and are differentiated mostly by subtle variations of color value, light and darkness create a strong feeling of space. The strong exaggeration of shadows in corners and at the bounds of surfaces suggest the usage of non-physically-based lighting models used to generate a digital, almost clinical look [36].

millions of particles and immense computing power.

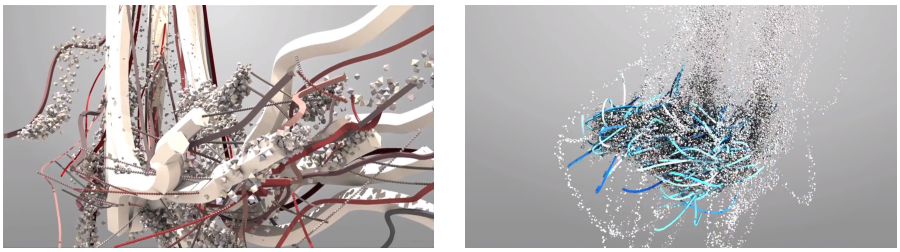
Today, the typical old look of particle effects—the unnatural behavior and artificial texture—can only be found in cheap games because with optimizations and fast computers, quite realistic looking water and fire effects can be simulated in real-time. Up-to-date liquid simulation technologies require much more calculation power and render time but don't seem to have any limitations as far as realism is concerned.

For the artist, particle systems have great potential because they are a way of abstracting the individual object and thinking of a complex system in a more general way, instead of having to plan the path of every single object. Particles add texture by contributing random elements, choreographed behavior or even near-physical simulation. In *Forms* the artists Quayola and Memo Akten use particles and procedurally generated geometry to visualize forces and motions occurring when world-class athletes perform their routines (see Figure 7.7).

The "Pixar look"

As pioneers of computer animation, the *Pixar Animation Studios* were the first to tackle and succeed creating a feature length computer animated film. The style exhibited in *Toy Story* was refined and increased in quality according to new technologies developed or improved especially for particular movies, like hair for *Monsters, Inc.* or underwater shading for *Finding Nemo*. In its essence, this style has been adopted by all other animation studios and a great number of animated films, both of long or short form, build on it.

In the tradition of Disney's traditional animation, it has a very cartoony visual language but combines it with strong physical lighting models like global illumination and subsurface scattering. Detailed textures and models add complexity just like fur, hair, particles, fluids and physics simula-



(a) (b)

Figure 7.7: The data on which the imagery is based was gathered entirely from video footage of the athletic performances. The result is an abstract animation that still suggests a certain connection with its source but operates in a digital, sterile space. It receives its meaning only through the association with the real-world events behind it [35].

tion. Until recently shading and lighting were the major culprits in creating images that clearly pointed to the computer as their source. But with developments in this area, 3D computer animation seems to develop a style independent of its technological limitations. Examples for this are movies like *Monster's University*, Dreamworks' *Home* and Disney's *Big Hero 6*, which are both, realistic and pleasing on their surface, and fantastic and meaningful in their core. Once the race for unprecedented realism and technological break-throughs in CG is coming to an end (and as it seems this is already happening), maybe animation will become more liberated and diverse again.[6]

[6]In painting, prevalent styles became much less prominent as soon as all secrets of photo-realism had been discovered in the 20th century. Today there still are many different directions in painting, most in the tradition of or based on previous styles, but there is no single one at the top.

Chapter 8

Conclusion

This paper may have touched upon only a small fraction of the actually observable effects of software on art and animation but I believe in conjunction with theoretical work and research from other fields and the opinions of artists representing the practical side, it has become clear that software influences its users and especially artists and professionals in the creative industry significantly. Still, psychological research about the effects of tools and especially software may be highly necessary.

One of the outcomes of the ELIZA project, conducted from 1964 to 1966 by Joseph Weizenbaum, seemed to be that people are easily tricked into believing a computer system to be intelligent. Humans are programmed to recognize other people based on a minimum of information and this principle has been used to simulate human vision, by making computers see faces everywhere, or to make software impersonate a therapist [25, p. 368ff.].

In opposite to this effect stands the objectification of computers and software. Since the processes taking place in the computer—the services, applications and algorithms—are invisible, the workings of the computer overall are transparent. This means that what the user sees of the ongoing processes is controlled by the computer itself or, in the end, the developers of the software. Algorithms and programming are not impartial and abstract, as corporations like Google or Autodesk want to make us believe, but expressions of the judgment and thinking of their creators, developers and executives. Much of this thinking might be directed to design and logic, but information about the actual agendas are not available to the consumer [67].

There are two fundamental differences between traditional tools and software, particularly in the art context: While most traditional (artist) tools are simple and easy to understand, question and manipulate, software is extremely complex and difficult to have an influence on. Also, software is mostly a black box whose content and principles of operation are completely inaccessible. For these reasons it seems naive and careless to believe that

software can be a tool just as subordinate and unbiased as a paintbrush—without access to the source code at least.

More specifically relating to art and animation software is the question of their missing flexibility in regards to the current requirements of the user. As opposed to programming languages, which are really more a problem-solving environment where one does the creative work in beforehand and then goes in to implement the solution in code, the purpose of animation software is not only to provide the fastest way of creating what the user imagines. Of course, a first step might be just that, to enable any user to quickly visualize what they have in mind, but on top of that animation software should also be the platform for innocent and undirected creative experimentation. It should inspire and challenge the mind of the user through its immanent logic when this is desired but be available as a well-meaning servant when absolute efficiency and obedience are required.

Overall, digital animation seems to go towards more interactivity, realism and hence less handmade and more code-generated content. It is obviously possible to proceduralize all existing visual styles—even the pre-digital ones—and thereby broaden the spectrum of available languages, but the logic of proceduralism has in fact been established firmly and is starting to become ordinary. Of course, new-media art can still be impressive and breath-taking, just like a great pre-digital James Bond movie, a Mozart Symphony or a fantasy novel. But they are not impressive mostly because of the novelty of their design, structure, materiality or logic but because of their universality, for which the rules have developed over the entire history of humanity and which probably are part of our nature, by design or culture. After all, despite being pre-digital relicts, the formations and the nature of this earth can still be very impressive as well. They may not be strictly speaking **impossible** or **invisible** like alien planets but they are still very much “animating”.

Appendix A

Interviews

A.1 Angie Jones

An article on CGW [1] *indicates that you have found your way to traditional animation after having worked with computers for a long time. I conclude that you must have considerable insight into the influences of animation tools and media on the artist.*

This statement is inaccurate. I started in traditional and moved in computers once traditional animation died as a medium in main stream animation. I am now working in oil painting and no longer work in animation at all as the industry has changed drastically and doesn't support creativity anymore.

What is your occupational background? (analog/digital, 2D/3D, creative/technical, etc.)

I am now a Assistant Professor in Practice of Animation at the John C. Hench Division of Animation & Digital Arts, School of Cinematic Arts, for the University of Southern California. I teach using Maya animation software but only as a tool for my students to tell their stories. I mostly teach how to tell a story visually no matter if you are doing it with paint, sand, clay or the computer. I also am a fine arts oil painter. I have not worked in CG animation production since 2011 when I last worked on the Smurfs movie. Mainstream mass produced animation (game, tv, feature or whatever) is about the bottom line for the board of directors, now and holds no interest for me any longer.

Why are you working digitally as opposed to working with traditional (analog) media?

I am not. I haven't worked digitally in over four years.

If you are working with analog as well as digital media, what do you see as their strengths and weaknesses?

[1] http://www.cgw.com/Publications/CGW/2006/Volume-29-Issue-11-Nov-2006-/Bridging-the-2D-and-CG-Gap.aspx

Both are tools. Simple as that.

What media software applications do you use for your work? Why are you working with these as opposed to other products?

Work: Teaching = Maya – Storytelling

Fine Art: Painting = Oil Pigment – Storytelling

In what ways do you feel restricted in your artistic freedom when working with these applications?

I don't feel restricted by either.

How would you describe the (visible) effects your software has on your (art)work, style and workflow?

The fragmented surface of my oil paintings reflect the wire frame models I have looked at in animation production for over twenty years.

What makes animation look digital or analog? The computer is a tool like the quill, but is it more constraining or liberating? How does the thinking change when moving from analog to digital media?

Do you think GOOD animation is defined by it's medium? The computer is just another tool. What you do with that tool using your own innate creativity is what is important. I find my creative thought process is the same no matter what tool I use. I am only chasing what Tom Waits refers to is the intangible spark that comes when you are stuck in traffic and cannot do anything about it... I am talking about "the zone" where the chatter in your mind goes quiet and it doesn't f-ing matter what tool you are using.

That you do not feel restricted by software applications (like Maya, which I consider bloated and buggy) is puzzling to me but all the more interesting. This I attribute to your two decades of experience in visual effects, animation and computer animation, which, on the other hand, makes me think that the computer and software can indeed become a part of oneself. Like pencil and brush.

Bloated and buggy? You can make ANYTHING with Maya!!! ANYTHING! I find the economy of low polygonal art refreshing and NOT ridiculous... like abstract art.

To me, low-poly art is not ridiculous (nor is abstract art, in some cases). But when I go outside I do not see wireframes and vertices on objects. Objects have volumes and surfaces, materials. They are tangible.

Everything I see breaks down into planes in my mind because that is how an artists thinks to delineate light color and form. I challenge any artists who does not see the world in that way.

A.2 Alvaro Gaivoto

What is your occupational background? (analog / digital, 2D / 3D, creative / technical, etc.)

I have been working in analog traditional 2D animation since 1976.

Why are you working digitally as opposed to working with traditional (analog) media?

I have recently changed to digital animation software at the request of various studios to make it easier to transfer files over internet to them, and cut down costs in the production.

If you are working with analog as well as digital media, what do you see as their strengths and weaknesses?

I find pencil and paper has a different "feel" to that of working on tablets. I can lightly sketch what I am thinking before committing to a stronger line. The tablet, pressure sensitive pens and modern software although are coming close to this ability. I think it is a matter of getting used to the new method. The great advantage of Digital is the delivery method. No more scanning of paper drawings and sending heavy files over the internet.

What media software applications do you use for your work? Why are you working with these as opposed to other products?

I am using for drawing and painting, Sketchbook pro, and art rage. There is an old obsolete animation line test called CTP (crater software) that I use often for animation linetests because it is very basic and easy to operate. I also use Toonboom studio and storyboard pro. As an animator I am basically interested in programs that I can do line tests with and not in how many colors it has or how many pencils. I normally don't color in-between or clean up my animation. A great advantage of digital is the ability to cut or copy and paste the whole or parts of a drawing to help in the next frame.

In what ways do you feel restricted in your artistic freedom when working with these applications?

This is an interesting question which I have talked about with other animators. If you do a nice pose on a piece of drawing paper, most people will comment on how nice YOUR drawing is, if however you show them a drawing on a tablet, they will comment on what a great tablet you have, or ask what software you use. Non artistic people will always appreciate paper drawings or analog painting much more that digital. They don't understand the digital process and think that somehow the software made the drawing look good and not the artist.

How would you describe the (visible) effects your software has on your (art) work, style and workflow?

I am still coming to terms wit digital drawing and for relaxation I will still go back to the old paper and pencil technique. The great advantage of the digital method is the increase in productivity and ease of delivery over an internet connection.

Further notes and comments

I still remember (yes I am that old) waiting for a DHL truck to show up at my door in London to take my scenes to Gatwick airport to be shipped to a studio in Munich. These would then be scanned and painted at the studio by other artists. Now I click on the send button and the files are there in

under 5 minutes. There two further points I would like to make about the advent of the digital process. First, most studios now expect the animator to clean up and paint his or her own animation without adding the extra funds or time allowance. Second, the client or agency have become aware of the ease of production and do not understand the creative process involved in animating. They think that you just press a button on the software and out comes a rabbit dancing, therefore the amount of time allowed for good animation is impossibly short, and the art suffers. As Richard Williams (Roger Rabbit director) once said to me in London "The monkeys have taken over the zoo"

Do the strengths of digital (speed, automation, composability) really contribute to the art and quality of animation?

The digital process does not necessarily contribute to the art and quality of animation, but it makes the artist more marketable. (Unfortunately this is reality). As I said in Hagenberg, a tablet for an animator is just a very expensive pencil. The real art and quality comes from the animator and not the tool he or she uses.

Do you think animating in 3D has an impact on how animators think or on how and what they animate?

A 3D animator is just as important as a 2D animator, but he or she is more of a puppeteer than an artist (in the drawing sense). One of the advantages 2D animators could have over others was the ability to draw. With the advent of 3D this necessity to be able to draw well became obsolete. In 2D we only have to worry how the drawing looks in a flat plane. In 3D the animator has to worry how the character is setup in all views. I really don't think an animator is impacted by the tools they use. The quality of the animation comes from the person and not from a pencil or a tablet.

Do you think it's easier to start animating (well) in 2D or in 3D? Assuming one has no prior knowledge of the basic principles.

I might be biased here, but I think it might be easier to animate in 3D, if you don't have the drawing skills. I didn't say more fun, I said easier. Once you have learned the secret of TIMING, POSING AND SPACING, it does not matter really what you use to animate with.

Apart from the visual and technical differences, do you think the question of 2D- or 3D-animation is merely a matter of preference, maybe? Of how one likes to think — visually or logically? Or has one a fundamental advantage?

I think 3D animation is an evolution of the animation genre. We might think back to the good old days, but the fact remains that producers follow the public taste and today apart from some old farts like, me most people want to watch 3D films. I of course think that the quality of story and direction has suffered to ease of scene manipulation. It used to be very hard to set up a pan and track in a 2D animated film. (check out Richard William's *The Thief and the Cobbler*), but with 3D it has become so easy that it is being used just to show off the software and not because the story

needs it. The question whether we will ever go back to 2D, I find no reason to think so. All of the 2D Animation films here in Europe that I have been asked to help with would not pay a living wage.

A.3 Mike Winkelmann

Why do you work with Cinema4D (and not with say ... Maya) [30], [31]?

I heard it was a lot easier to learn and have stuck with it. don't need anything high end like maya.

What do you like about its features that you think you wouldn't get from another software?

I've heard the mograph stuff is a lot better. to be honest i've heard people mostly complain about maya.

Do you work "against" the software / "abuse" it / use it in unintended ways?

Ummm, not sure. not sure how it's intended to be used. not sure that's even possible. good software is really open and can't be "abused"

Can you think of any visual characteristics that you would think are "totally Cinema4D"?

Ummm, i think people copy certain styles but i think that's mostly out of laziness or a lack of ideas. has nothing to do with the software, this happens in every medium.

Appendix B

CD-ROM Content

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

B.1 Thesis

Pfad: /

Svetitsch_Klemens_2015.pdf Master's Thesis

B.2 Online Literatue

Pfad: /OnlineLiterature

*.pdf . . . . . Archived online literature

B.3 Images

Pfad: /Images

. . . . . . Images included in the Master's Thesis PDF

References

Literature

[1] Ed Catmull with Amy Wallace. *Creativity, Inc.: Overcoming the Unseen Forces That Stand in the Way of True Inspiration*. New York: Random House Publishing Group, 2014 (cit. on p. 64).

[2] Daniel Chandler. *Semiotics: the basics*. Second edition. London and New York: Routledge, 2007 (cit. on p. 4).

[3] Ben Cole. "An Overview of a Film Production". In: *Production Pipeline Fundamentals for Film and Game*. Ed. by Renee Dunlop. Burlington, Massachusetts: Focal Press, 2014 (cit. on p. 15).

[4] Florian Cramer. "Language". In: *Software Studies: A Lexicon*. Ed. by Matthew Fuller. Leonardo Books. Cambridge, Massachusetts: The MIT Press, 2008, pp. 168–174 (cit. on pp. 19, 20).

[5] Florian Cramer and Matthew Fuller. "Interface". In: *Software Studies: A Lexicon*. Ed. by Matthew Fuller. Leonardo Books. Cambridge, Massachusetts: The MIT Press, 2008, pp. 149–153 (cit. on p. 21).

[6] Mihaly Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. New York: Harper and Row, 1990 (cit. on p. 21).

[7] Sigmund Freud. *Das Unbehagen in der Kultur*. Wien: Internationaler Psychoanalytischer Verlag, 1930 (cit. on p. 4).

[8] Gottfried Hofmann. "Open Source von den großen VFX-Studios". In: *Digital Production* 2015.05 (August 2015), pp. 8–11 (cit. on pp. 57, 58).

[9] Elésiane Huve. "Beat the Brush". In: *Digital Production* 2015.05 (August 2015), pp. 28–31 (cit. on p. 44).

[10] Steven Johnson. *Interface Culture: Wie neue Technologien Kreativität und Kommunikation verändern*. Stuttgart: Klett-Cotta, 1997 (cit. on pp. 25, 26).

[11] Christoph Keese. *Silicon Valley: Was aus dem mächtigsten Tal der Welt auf uns zukommt*. München: Albrecht Knaus Verlag, 2014 (cit. on p. 35).

[12] Keywan Mahintorabi. “Das digitale Zeitalter der Spezialeffekte”. In: *Digital Production* 2015.01 (Januar/Februar 2015), pp. 8–12 (cit. on p. 41).

[13] Benoît B. Mandelbrot. *The Fractal Geometry of Nature*. Rev. ed. of: Fractals, 1977. New York: W.H. Freeman and Company, 1982 (cit. on p. 42).

[14] Lev Manovich. *Software Takes Command*. International Texts in Critical Media Aesthetics. New York: Bloomsbury Academic, 2013 (cit. on pp. vii, 14, 32–34, 45, 51, 75).

[15] Lev Manovich. *The Language of New Media*. Cambridge, Massachusetts: The MIT Press, 2001 (cit. on pp. 15, 53).

[16] Scott McCloud. *Understanding Comics: The Invisible Art*. New York: Harper Perennial, 1994 (cit. on pp. 4, 70, 76).

[17] Marshall McLuhan. *Understanding Media: The Extensions of Man*. Originally published in 1964. Berkeley, California: Ginkgo Press, 2003 (cit. on p. 4).

[18] Bill Moggridge. *Designing Interactions*. Cambridge, Massachusetts: The MIT Press, 2007 (cit. on p. 63).

[19] Winfried Nerdinger. *Perspektiven der Kunst: von der Karolingerzeit bis zur Gegenwart*. München: Martin Lurz, 1994 (cit. on p. 47).

[20] Don Norman. *The Design of Everyday Things*. New York: Basic Books, 2013 (cit. on pp. 22–26, 31).

[21] Søren Pold. “Button”. In: *Software Studies: A Lexicon*. Ed. by Matthew Fuller. Leonardo Books. Cambridge, Massachusetts: The MIT Press, 2008, pp. 31–36 (cit. on p. 26).

[22] Casey Reas, Chandler McWilliams, and Jeroen Barendse. *Form+code in design, art, and architecture*. New York: Princeton Architectural Press, 2010 (cit. on pp. 6, 14, 44).

[23] Tom Sito. *Moving Innovation: A History of Computer Animation*. Cambridge, Massachusetts: The MIT Press, 2013 (cit. on pp. 6, 7, 40, 48, 63).

[24] Kostas Terzidis. “Tool-Makers vs Tool-Users (or both)?” In: *Digital Pedagogies. form-Z Joint Studies Program Report* (Dec. 2006), pp. 77–79 (cit. on pp. 11, 13, 30, 70).

[25] Joseph Weizenbaum. “From Computer Power and Human Reason-From Judgment to Calculation”. In: *The New Media Reader*. Ed. by Noah Wardrip-Fruin and Nick Montfort. Cambridge, Massachusetts; London, England: The MIT Press, 2003, pp. 368–375 (cit. on p. 80).

[26] Paul Wells. *Re-Imagining Animation - The changing face of the moving image*. Lausanne, Switzerland: AVA Publishing SA, 2008 (cit. on pp. 66, 68).

[27] Aylish Wood. "Behind the Scenes: A Study of Autodesk Maya". In: *Animation: An Interdisciplinary Journal* 9.3 (November 2014), pp. 317–332 (cit. on p. 48).

Software

[28] *After Effects*. URL: http://www.adobe.com/products/aftereffects.html (visited on 10/16/2015) (cit. on p. 51).

[29] *Akeytsu*. 2015. URL: https://www.nukeygara.com/akeytsu (visited on 10/09/2015) (cit. on p. 58).

[30] *Cinema4D*. URL: http://www.maxon.net/en/products/cinema-4d-studio/ (visited on 10/18/2015) (cit. on p. 86).

[31] *Maya*. URL: http://www.autodesk.com/products/maya/overview (visited on 10/09/2015) (cit. on pp. 48, 86).

[32] *Nuke*. URL: https://www.thefoundry.co.uk/products/nuke/ (visited on 11/10/2015) (cit. on p. 54).

[33] *Photoshop*. URL: http://www.photoshop.com/ (visited on 10/14/2015) (cit. on p. 53).

Films and audio-visual media

[34] *A Computer Animated Hand*. By Edwin Catmull and Fred Parke. 1972 (cit. on p. 48).

[36] *BEAT*. Story, Direction and Animation: Or Bar-El. 2011. URL: https://vimeo.com/31423544 (visited on 11/15/2015) (cit. on p. 78).

[37] *Behind the Ears: The True Story of Roger Rabbit*. DVD. Who Framed Roger Rabbit (Vista Series). Bonus Features, Disc Two. Published by Disney Home Video. 2003 (cit. on p. 47).

[38] *Big Hero 6*. DVD. Production: Walt Disney Animation Studios, Directors: Don Hall and Chris Williams. 2014 (cit. on p. 71).

[39] *Bonobo - 'Cirrus'*. By Cyriak. 2013. URL: https://vimeo.com/58115286 (visited on 11/15/2015) (cit. on p. 46).

[40] Osvaldo Cavandoli. *La Linea 3*. DVD. Produced by Wagner-Hallig-Film. 2004 (cit. on pp. 72, 73).

[41] *Contre Temps*. By Jérémi Boutelet, Thibaud Clergue, Tristan Ménard, Camille Perrin, Gaël Megherbi, Lucas Veber. 2012. URL: https://vimeo.com/71695621 (visited on 11/15/2015) (cit. on p. 46).

[42] *Cycles*. By Cyriak. 2010. URL: https://www.youtube.com/watch?v=-0Xa4bHcJu8 (visited on 11/15/2015) (cit. on p. 46).

[47] *Flying Lotus - Kill Your Co-Workers*. Animation: Mike Winkelmann. 2010. URL: https://vimeo.com/15572863 (visited on 10/17/2015) (cit. on pp. 76, 77).

[35] *Forms*. By Memo Akten and Quayola. 2012. URL: http://www.memo.tv/forms/ (cit. on p. 79).

[44] *La Faim*. Animation. Production: National Film Board of Canada, Director: René Jodoin, Animation: Peter Foldès. 1974. URL: http://www.nfb.ca/film/la_faim (cit. on p. 76).

[45] *The Lego Movie*. DVD. Production: Warner Bros., Director: Phil Lord and Christopher Miller. 2014 (cit. on p. 71).

[43] *Walking City*. Production: Universal Everything, Creative Director: Matt Pyke, Animation: Chris Perry. 2014. URL: https://vimeo.com/85596568 (visited on 11/15/2015) (cit. on p. 74).

[46] *Who Framed Roger Rabbit*. DVD. Production: Touchstone Pictures, Directors: Robert Zemeckis. 1988 (cit. on p. 47).

Online sources

[48] URL: http://en.wikipedia.org/wiki/Technology (visited on 04/29/2015) (cit. on p. 1).

[49] URL: http://en.wikipedia.org/wiki/Tool (visited on 05/16/2015) (cit. on p. 2).

[50] URL: https://en.wikipedia.org/wiki/Semantics (visited on 10/11/2015) (cit. on p. 4).

[51] URL: http://en.wikipedia.org/wiki/Pictogram (visited on 05/24/2015) (cit. on p. 5).

[52] URL: http://en.wikipedia.org/wiki/Turing_completeness (visited on 05/18/2015) (cit. on p. 19).

[53] URL: http://en.wikipedia.org/wiki/Programming_language (visited on 05/21/2015) (cit. on p. 20).

[54] URL: http://en.wikipedia.org/wiki/Programming_paradigm (visited on 05/21/2015) (cit. on p. 20).

[55] URL: http://wiki.blender.org/index.php/Doc:2.4/Manual/3D_interaction/Transform_Control/Pivot_Point/3D_Cursor (visited on 07/26/2015) (cit. on p. 24).

[56] URL: http://en.wikipedia.org/wiki/Interactivity (visited on 05/17/2015) (cit. on p. 34).

[57] URL: https://en.wikipedia.org/wiki/Computer_Animation_Production_System (visited on 08/25/2015) (cit. on p. 43).

[58] URL: http://sa-staging.com/programs/attachments/thesafehouse_presskit.pdf (visited on 08/31/2015) (cit. on p. 45).

[59] URL: http://www.maxwellrender.com/index.php/products/maxwell-render-suite/why-choose-it (visited on 10/27/2015) (cit. on p. 50).

[60] URL: https://en.wikipedia.org/wiki/Node.js (visited on 10/09/2015) (cit. on p. 57).

[61] URL: https://en.wikipedia.org/wiki/JavaScript (visited on 10/08/2015) (cit. on p. 57).

[62] URL: http://download.autodesk.com/us/fbx/20112/FBX_SDK_HELP/index.html?url=WS1a9193826455f5ff-150b16da11960d83164-6c6f.htm,topicNumber=d0e294 (visited on 10/07/2015) (cit. on p. 57).

[63] URL: http://brighttyger.com/post/312089462/david-oreilly-golden-bear-winner-interview (visited on 08/24/2015) (cit. on p. 65).

[64] *Alembic*. URL: http://www.alembic.io/ (visited on 10/14/2015) (cit. on p. 57).

[65] Giannalberto Bendazzi. *Defining Animation - A Proposal*. 2004. URL: http://giannalbertobendazzi.com/wp-content/uploads/2013/08/Defining_Animation-Giannalberto_Bendazzi2004.pdf (visited on 05/03/2015) (cit. on p. 5).

[66] *Best Practice*. URL: https://en.wikipedia.org/wiki/Best_practice (visited on 10/09/2015) (cit. on p. 56).

[67] Nicholas Carr. *The Manipulators: Facebook's Social Engineering Project*. Sept. 14, 2014. URL: https://lareviewofbooks.org/essay/manipulators-facebooks-social-engineering-project (visited on 10/12/2015) (cit. on p. 80).

[68] Florian Cramer. *What is 'Post-digital'?* URL: http://www.aprja.net/?p=1318 (cit. on pp. 29, 38).

[69] Nichola Dobson. *Taking for granted the digital world*. 2013. URL: http://blog.animationstudies.org/?p=390 (visited on 10/07/2015) (cit. on p. 6).

[70] *FBX binary file format specification*. Aug. 10, 2013. URL: https://code.blender.org/2013/08/fbx-binary-file-format-specification/ (visited on 10/07/2015) (cit. on p. 57).

[71] Wesley Fenlon. *2D Animation in the Digital Era: Interview with Japanese Director Makoto Shinkai*. Sept. 20, 2012. URL: http://www.tested.com/art/movies/442545-2d-animation-digital-era-interview-japanese-director-makoto-shinkai/ (visited on 10/28/2015) (cit. on pp. 17, 36).

[72] Jeffery Harrell. *What went wrong with Final Cut Pro X*. June 23, 2011. URL: http://jefferyharrell.tumblr.com/post/6830049685/what-went-wrong-with-final-cut-pro-x (visited on 10/28/2015) (cit. on p. 17).

[73] *Making Maya Easier To Use*. URL: https://www.youtube.com/watch?v=zR5ZsEIc9u8 (visited on 10/09/2015) (cit. on pp. 55, 56).

[74] David Mitchell. *The Future of the Cartoon Feature Film: An Overview of CGI Animation*. 2002. URL: http://www.zenoshrdlu.com/zenocgi.htm#oview (visited on 10/28/2015) (cit. on p. 71).

[75] David Mitchell. *The Future of the Cartoon Feature Film: Appendix 2*. 2002. URL: http://www.zenoshrdlu.com/zenocgi.htm#app2 (visited on 10/28/2015) (cit. on p. 43).

[76] *OpenEXR*. URL: http://www.openexr.com/ (visited on 10/14/2015) (cit. on p. 58).

[77] *Plug-in (computing)*. URL: https://en.wikipedia.org/wiki/Plug-in_(computing) (visited on 10/12/2015) (cit. on p. 16).

[78] *Software*. URL: https://en.wikipedia.org/wiki/Software (visited on 10/16/2015) (cit. on p. 33).

[79] *Supercharged Animation Performance in Maya 2016*. URL: https://www.youtube.com/watch?v=KKC7A9bbUuk (visited on 10/09/2015) (cit. on p. 55).

[80] *Transistor*. URL: https://en.wikipedia.org/wiki/Transistor (visited on 10/10/2015) (cit. on p. 29).

[81] Steve Wright. *The Importance of Invisible Effects*. 2008. URL: https://library.creativecow.net/articles/wright_steve/Creative_Cow_Magazine_VFX_Invisible_Effects.php (visited on 10/07/2015) (cit. on p. 6).