VU VRIJE UNIVERSITEIT AMSTERDAM

Faculty of Science
Computer Sciences

Exploring hypermedia in 2018 through the lens of XIMPEL

Implications on education, parallel media and frustration detection

Master's thesis in Master Computer Science (Multimedia specialization)

Melvin Richy Roest

Vrije Universiteit Amsterdam
Faculty of Sciences
Amsterdam, The Netherlands 2018

Master's thesis 2018

Exploring hypermedia in 2018 through the lens of XIMPEL

Implications on education, parallel media and frustration detection

Melvin Richy Roest

Department of Computer Sciences
Section: Business Web & Media
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands 2018

Exploring hypermedia in 2018 through the lens of XIMPEL
Implications on education, parallel media and frustration detection
Melvin Richy Roest

Supervisor and 1st reader: Prof. Dr. Anton Eliëns
2nd reader: Dr. ing. Sander C.J. Bakkes
Technical Supervision: Msc. Winoe Bhikharie

Master's Thesis 2018
Department of Computer Sciences
Section: Business Web & Media
Vrije Universiteit Amsterdam
De Boelelaan 1105, 1081 HV Amsterdam
Telephone +31 20 598 9898

Cover: A visual metaphor for exploring the intersection between choice and media. The camera is cropped with the pen tool in Sketch 3. It comes from a beautiful photograph made by Fabrizio Verrecchia. The image in the camera is cropped from an epic background figure shot by Tom Barrett. Unsplash.com is the website where these beautiful gems have been found. The camera icons (record icon, time icon and battery icon) are made by yours truly with the marvelous pen tool.

Typeset in LuaLaTeX
Printed by Melvin's printer in his home
Amsterdam, The Netherlands 2018

Exploring hypermedia in 2018 through the lens of XIMPEL
Implications on education, parallel media and frustration detection
Melvin Roest
student number: 1914588
Department of Computer Sciences
Section: Business Web & Media
Vrije Universiteit Amsterdam

Abstract

Hypermedia is a research topic that has gone into obscurity. However, it is useful for many different use cases such as education and interactive storytelling. The web front-end framework XIMPEL (eXtensible Interactive Media Player for Entertainment and Learning) is a cross section between hypermedia, storytelling and gameplay. It is able to provide for these use cases, as well as the unique use case of for a (subset of) digital games creation such as simple shooters, point & click adventures and playthroughs of simple existing games such as Flappy Bird.
This thesis is an exploration on the topic of hypermedia that was seeded with the question of: how does XIMPEL need to be extended in order to facilitate development for a bigger variety of educational applications? This led to other questions which were then looked into as well while keeping the following question in mind: what research topics are relevant for hypermedia in order to advance the field? The approach of asking a question giving an answer, which may lead to another question is called an exploration. If a new question arose, then that question was pursued if an adequate answer or approach to an adequate answer was found with regards to the old question. Six explorations have been done and produced the following outcomes:
Exploration 1: XIMPEL could not offer interactive programming lessons. It has been extended with a new terminal tag in order to allow interactive command-line lessons.
Exploration 2: XIMPEL could not offer parallel media playback. It has been extended with a parallel tag for parallel media playback.
Exploration 3: XIMPEL and ReactJS seemed to have many similarities, which could justify a port of XIMPEL to ReactJS in order to develop and maintain XIMPEL more effectively in the future. To explore this link, XIMPEL has been reimplemented in ReactJS and results are mostly positive.
Exploration 4: frustration and engagement are important topics for education and have not been explored in hypermedia research. A nuanced characterization of frustration is motivated through literature. Moreover, XIMPEL has been extended with detecting it in users through facial expression classification and capturing: mouse movements, mouse clicks and the user history.
Exploration 5: design questions of time scrubbing and parallel media has been explored. All permutations on the design questions asked have been illustrated by figures.
Exploration 6: mechanisms for media playback that needs to play after an overlaying rectangle shaped link is clicked are presented.
In the process of these explorations, XIMPEL has been extended for better massive open online course creation by being extended with a microservice architecture. It has been found that parallel media playback impacts design and implementation more than expected such as the design questions regarding time scrubbing or media items surviving subjects switches (exploration 6). Exploration 6 would be impossible without parallel media playback. The most potent future areas of research are: user testing regarding time scrubbing and hypermedia, XIMPEL and the intersection of augmented and virtual reality, hypermedia and gamification, remixing the web through hypermedia, hypermedia and its potential for procedural rhetoric and porting XIMPEL to React Native in order to make gamified hypermedia possible on smart phones and tablets.

Keywords: hypermedia, gameplay, frustration, engagement, interactive narratives, choice, education, software development, exploratory research.

0.1 Preface

Hello dear reader,
welcome to the preface of my computer science thesis. Before we start I would like to point out that computer science is not about computers, nor is it a science. It is not about computers since the computer is a tool. It is mainly a tool for simple calculations in a step by step fashion (e.g. adding, subtracting or multiplication, remembering old results). Producing a series of these steps is what makes these calculations complicated or complex. In itself this is not really useful for the lay person. It calculates what? How do we see this calculation? In 2018, we see things through a computer screen. The combination of a computer and a computer screen (and speakers) allows us to consume multimedia content like audio, images or videos. 50 years ago, this idea was groundbreaking. Douglas Engelbart demonstrated in 1968 what a computer combined with a screen, speakers, keyboard and a mouse is really capable of[48]. When people talk about computers they imply that a computer screen, speakers, keyboard and mouse comes attached with it. However, since my grandparents might read this preface, I will not do the same.
So computers (including computer screens and speakers) are about the ability to: consume multimedia, perform calculations and control these two things in a very fine-grained or coarse-grained fashion. The big problem with this model is that in order to use a computer, one needs to have detailed knowledge about the inner workings of a computer in order to perform calculations or consume (or create) multimedia. The few people who had this knowledge decided to make it easier for others by instructing the computer to load up images of a metaphorical desktop with files on it. Most people have some experience with an actual desktop and actual files, and it helped. All computers do this automatically nowadays. These combined computer instructions are able to – in this example – load up images of a desktop with files. Together, these instructions are called a computer program, or better known as software. This particular piece of software is called an operating system. The few people with expert knowledge decided to build more instructions with even more metaphors that would allow people to understand how to perform calculations on a computer and create or consume media. The spreadsheet application was one of the first pieces of software that made the computers ability to calculate tangible for the lay person.
The study of computer science is not about computers. Computer science courses are not about the computer itself but the software running on it. Perhaps software science would be a better term then? It even has an alliteration! Most courses I followed were about the creation of software. More specifically, it is about software in general and the creation of software in order to make software creation easier[1]. In other words, it is not about computers. The courses were about about creating software and creating software that allows one to create software more easily. Perhaps it could be called meta-software creation. This reflective quality sure makes it have some deeper layers, which feel right at home at a university.
It is also arguably not about science. Science can be defined in many ways. I recommend the reader to follow an introductory course in the philosophy of science if they have not done so. I also would like to note that some researchers do not necessarily know about the philosophy of science. Nevertheless, in the philosophy of science it becomes quite clear that there are multiple views on what science is. So there is probably a way of arguing that it is a science. I have spoken to physicists who believe psychology is not a science and spoken to angry psychologists who believe it is. The claim that something is not a science could become quite a messy and emotionally heated discussion.
However, unlike psychologists, the computer science community internally disagrees on the question

[1]Computer graphics (computer graphics engine creation), Multimedia authoring (general software creation), Distributed Multimedia Systems (fundamentals of multimedia streams and codecs), Distributed Systems (one piece of software on multiple computers and its implications), Knowledge and Media (information organization), The Social Web (web app creation with a social twist), Experimental design and data analysis (statistics for programmers), Concurrency and Multithreading (writing software for multiple processor cores), Software Configuration Management (tracking changes in software), Operating Systems (creating operating systems and understanding how operating systems work), Systems Security (how to defend against people using software in ways it was not intended, mostly for personal enrichment purposes), Binary and Malware Analysis (how to analyze software created for criminal purposes)

whether it is a science or not [24]. At its most optimistic one could argue that computer scientists are unknowingly practicing science. A cynic would say that there are simply two types of computer scientists: those who practice science and those who do not. However, it begs the question how can one practice science unknowingly? And how can papers be published as science by non-science practicing computer scientists? The answer is: any argument that I come up with to give a proper justification is far-fetched and hairy.

What shocks me the most is the little reflection I have experienced on the Vrije Universiteit Amsterdam regarding whether it is a science. I do not think this only pertains to the Vrije Universiteit Amsterdam. When I was at the Foundation of Digital Games conference, there was a research paper on all the research of game studies clustered in graphs [62] and if I remember correctly, it proved a point that scientific philosopher Kuhn or Lakatos made. I remember that I told them that this point was made by one of the two philosophers of science and they never heard about the philosophy of science as a field! Yet, their methodology was a demonstration on how science could be seen as a social construct and has loosely defined principles at best. It gives me some faith in the academic community that they do not need philosophers of science to showcase similar thoughts, but at the same time Google Scholar is one or two clicks away.

The most notable example at the Vrije Universiteit Amsterdam was when I asked in a computer security class "is computer science a science?" I got laughed at. Then, the security researcher looked at me with a pause and said "it is not science. It is engineering." [44] I did not ask the class why they laughed at me, but a friend and classmate laughed at me because he had a full conviction that it is a science – he was really surprised to hear the lecturer say otherwise. I believe that many people in that class had the same reaction for more or less the same reasons. It is a shame that my university was not the place that these in-class discussions happened more often and when they do, you get laughed at. Another example: just before publishing this thesis, I found a blog post of an AI researcher that I like who also claims his work is "not science, barely engineering and absolutely not mathematics." Nevertheless, it is "valuable and interesting research." [95] I agree, his research his awesome.

I cannot blame the average reflection of the computer science student regarding this question. As you may have read in a previous footnote, there is no course about such reflectiveness! Contrast this with my psychology bachelor program and it is visible that one third of the courses are directly about this question or connected to the question: is psychology a science? Psychologists would say yes and have compelling arguments. Though they are terrible practitioners of it as a group, but they even acknowledge it! [17] That is scientific integrity, especially considering the forces that work against academia nowadays. Apparently, these problems exist in many disciplines, but I do not know enough about them to comment on it.

Now, dear reader, there are a few problems. The first one is that this thesis is for a degree of a master in science. I feel like a charlatan, because I believe software creation to be more close to engineering than science. The saving grace is that engineers are useful to scientists since they are more capable for developing and commercializing tools that allow the advancement of science[2]. Computer scientists create software of a certain complexity and with a certain reflectiveness in mind, but most of the time it is not science. Moreover, when it is arguably science it may pertain to another discipline. For example, human-computer interaction is psychology. The theory of computation is called theoretical computer science but to me it seems like mathematics. Is mathematics even a science? Forget I asked, it is at least very helpful to science. I find something being helpful to science a much more important criterion on whether something is academic than if a certain pursuit is scientific.

The second problem is that this master program gave me very little formal scientific training. What I know about science comes from: my own interest, my bachelor in psychology and my master in game studies. So I happen to know a thing or two about science and demonstrated it in the respective programs. But what is a thesis supposed to be for computer science? It is confusing to say the least. For example, a friend of mine created an app that should be able to foster trust in people as a

[2]For example, I wonder how many computer scientists or similar researchers develop tools for other research departments. I would be a prime candidate to do this in the future. And I have done this in the past for the psychology and education department at my university.

master thesis of computer science. One could say it is human-computer Interaction. I would say it is psychology. The app is the intervention in order to create a certain behavioral outcome. I would also say it is useful and beneficial for society to have an app that helps fostering trust in a day and age where it sometimes seems to be lacking a bit.

Because of this confusion, I decided upon an I am going to please everybody approach. Not only because it seems like my best bet to get a degree in computer science, but also because it seems to be really fun and reflective on the process of science itself! Parts of this thesis are clearly scientific in nature (e.g. chapter 5 where I try to track frustrated facial expressions among other things). Parts of this thesis are engineering (e.g. chapter 2, 3, 4 and 7). Perhaps computer scientists consider it academic, I would not know since the training on academic writing in computer science was very low at the Vrije Universiteit Amsterdam. While I did enjoy a great amount of academic writing skills in my psychology program and game studies program[3], they are different fields and their quirks get in the way of understanding the science in computer science. And to cement the academic nature of my thesis I wrote a chapter that is very similar in methodology but uniquely different to someone else his master thesis, who got a degree in computer science because of it. If people believe my thesis to be non-academic, then I can point to his work and claim my work is comparable and therefore worthy of a degree (see chapter 4).

So I covered three aspects in which my thesis is worthy for receiving a computer science degree. It is: scientific, engineering and up to the standards of other people who wrote theses in a similar style, pertaining to that specific chapter. Another reason why I wrote this thesis is because this project would set me up for industry. It involved a lot of web development in technologies employers are searching for. It is my personal goal to be employable since most of my family have blue collar jobs and they did not want me to suffer the same almost back breaking fate. The final reason is because it is fun! Science is fun. Exploring a topic is fun. And the theme of transformation in computer science (e.g. parsing) is fun! One chapter shows me having pure fun deep diving a topic, which is chapter 6. The chapter is not: scientific, engineering or a blueprint in order to get your degree. However, it does ask important questions needed to advance the field of research that I am exploring: hypermedia.

A third problem is that academia in general is quite rooted in tradition and conventions. One is not being able to write "I" lest it hampers the objectivity of one's statements. However, how do these conventions make sense when computer scientists do not even agree on basic questions regarding its scientific status? I believe, that people follow conventions because they either have to or are not even thinking about it. The reason I believe this is because in every academic writing course or exercise teachers told me a variation of: we follow conventions because other people do. In high school I learned this is a fallacy. Therefore, I have decided to relax any conventions as much as possible since I believe academics are following them blindly. Sometimes I do use those conventions in order to give fellow students and academics a sense of familiarity.

Unfortunately, I introduce problems by taking this approach since it will be tougher to understand me. Researchers and research articles normally follow a blueprint. This thesis relaxes this blueprint. I will do my best to foreshadow at any point what is about to come, especially if it is something more unconventional. Moreover, I encourage the reader to take out the good bits of this thesis and leave the bad bits (no error correcting pun intended). Doing a thesis this way is an exploration and almost a small experiment in itself – except it is not controlled enough to be called an experiment. Innovation means taking on risk and I ask my supervisors to allow me to take this risk even if it fails. My other theses show that I know how to do experiments or write academically, this format – including this preface – is hopefully the start of a discussion on how to improve the academic writing and publishing process.

So what do I do differently? Ha! I am glad you asked! In this thesis I explore. It has six explorations[4]. Some of these have a theoretical background, others do not, which is on purpose. In these explorations

[3]I should probably note that my game studies program was followed at the Universiteit van Amsterdam. My psychology bachelor was at the Vrije Universiteit Amsterdam.

[4]Edit: 7 now, I keep on going, I am pretty sure my supervisors would have been happy with only the first 4 explorations. The seventh exploration is in appendix A. The thesis deadline is too close to rewrite everything.

I will tell the story of how I came to that particular exploration. I believe it is important to tell the context of an exploration, because it may help others to be inspired to form intuitions to advance their academic or scientific thinking. This part is always overlooked in published papers, which makes no sense to me. I thought the observation part of the empirical cycle is a necessary step, then why is it underreported? The process of how one does science should be part of academia, even if it sounds silly.

0.2 A small guide on the reference list and footnotes

I am a big fan of showing where I got my information from. However, I found it too restrictive to only cite academic sources. Therefore, I did not. Unfortunately, a citation seems rather authorative [42, 13, 7].
In this case I just wrote random numbers. The first number is the answer to the ultimate question of life, the universe, and everything. The other number is rather unlucky according to superstitious folks. The final number is lucky in a gambling context or biblical in a religious one. More importantly, if these three numbers gave you a small feeling that I knew what I was talking about, then I have successfully fooled you. Sources should not induce such a feeling, since the source always needs to be verified, not glossed over! Alas, I know that most academics do not have the time. Because of this, I organized my bibliography slightly differently than what I have been taught.
First of all, my bibliography has section headers. The sections *Articles* and *Conferences Papers* are peer-reviewed. The references of the section *Master Theses* are reviewed by the respective supervisor and second reader. The section *Other Online Sources, Books, Book Chapters, Chapters in a Collection, Misc* and *Reports* are academic in nature but not peer-reviewed. *Github repositories* speak for themselves. The final section *Hip Blog Posts From The Web* and *Hypermedia Companies* are not peer-reviewed or academic. Why do I use these as references? To show where I know my truth from. I fortunately use references of the *Hip Blog Posts* section rarely and for very tangential related ideas. These references are not strictly needed for this thesis (the articles are a fun read though!).
Then, there are also footnotes. In some cases these footnotes have URLs. These URLs did not make the reference list for the simple reason that in most cases are also very tangentially related, and I have one or two sentences on how to use the resource in the context of this thesis. These footnotes are as important the *Hip Blog Posts* section. Though in some cases they are rather serious material (e.g. see the footnote in exploration 4 about PCA and related statistical methods).

0.3 Acknowledgements

I have many people to thank for writing my most daring academic thesis and text. First and foremost, I have to thank my supervisor Anton Eliëns who has the audacity to be controversial if his first principle styled or associative thinking leads him there. I am grateful for that he is giving me the space to do my utmost best, no matter how unconventional it looks. Also, the one on one discussions I had with him are always thought provoking and amazing. I have to thank my second reader Sander Bakkes for being the biggest contributor in showing that I matter to the academic community. This is especially true after feeling like I did not matter during psychology degree, or during my bachelor thesis of information science where I believed to have a publishable result (it probably still is, anyone?). I also would like to thank Winoe Bhikharie for showing how to create magic with the frontend framework I wrote a whole thesis about. I also want to give a warm thanks to Hugo Huurdeman. He made me believe the frontend framework of which the name will be mentioned a lot (!) later on has a future. This thesis project started out in a completely different way on a completely different subject. Because of this, I have met people at the play space company called Terragon. I would like to thank everyone I met there, especially Minne Belger. If one person is larger than life, it is you. When it comes to people at the Vrije Universiteit Amsterdam in general I have to give Otto Schrofer a heartfelt thanks. I know

I am not the easiest student (probably measurable by the number of words this preface is), but you guided me above and beyond. The same goes for Frank Nack at the University of Amsterdam who helped me a great deal for having issues regarding academia.
I would also like to thank people I am close to. A few of them stand out. Timothy Tjen A Looi is the first person to thank. He helped me obtaining a JavaScript/NodeJS teaching position at the New York Code + Design Academy and then pulled me into Fleks, a startup that needed help with their ReactJS-driven application. Becoming a teacher and freelancer while writing a thesis forced me to professionalize, which helped my programming and my workflow for my thesis as well as my professional life. The conversations we still have about programming are invaluable. The same thanks about awesome programming conversations goes to Gábor Kozár. Creating a virtual environment so that you can have ring 0 privileges, play kernel and prevent dangling pointers during memory allocation is no easy task. But you did it and I am proud of you! On another note, I have to express my gratitude to Marvel Maljers. I believed I was reflective on the question of what science is, until you claimed that all science is gendered and prompted me into asking myself what that means. It is statements like those that advance my thinking and allow me to do seemingly crazy things all in the hope to advance the positive parts of humanity. Your steps that took you into learning C, Python and JavaScript directly made it into this thesis through means of a picture with a rubber duck. It is a picture from the course of Introduction to Computer Science by Harvard, also known as CS50. CS50 is an amazing inspirational source on how to give online computer science education. Moreover, I just have to thank you in general: you support me, help me in developing to become a better person (by the way, I meditated today!) and keep me curious. Somehow I really felt that going to opera's and piano trios helped me to appreciate media a lot more in a much broader context. I also thoroughly enjoyed the conversations I had with your mother about my thesis. It was the first time in my life that I connected with someone on the struggle of writing a thesis, because it can be a struggle! So Chantal, thank you. Moreover, Chantal your husband will be acknowledged as well because the general brainstorms I had with him, or talks about algorithms or algorithmic art were all topics that played a role in writing this little non-fictional novel that I call my thesis. I also have to thank Jeffrey Bruijntjes who gave me advice on domain specific languages, which this thesis will partially be about. But Jeffrey, I mostly have to thank you for joining me on a journey that we walked together! It was crazy, it was fun and man you are fast! You won the race, and I miss you. I hope working life at ING is treating you well.
Since my acknowledgements section has been written in a short span of time – a snapshot if you will – it may be the case that I have forgotten some people who really helped me with it. For everyone who felt they have invested in me working on my thesis, I would like to extent my gratitude to you. I am sorry that you were not directly included, but since my thesis will be hosted on my website, I can always update it and add you in order to resolve this.
Finally, I would like to thank my grandparents. I know you have no clue what I am doing, and you may not even read this. But I feel your support every day. Moreover, the relentless focus on being able to do mental arithmetic and touch typing as a kid undeniably helped me to learn computer programming faster and better than I otherwise would have.
There is also a postface! Apparently, it is a preface put after the main body of work. There were some topics that I believe are more interesting to discuss after you have read the thesis like my own evaluation, for example.

1 Introduction

In a lot of cases groundbreaking scientific discoveries are made by accident. Without these serendipitous discoveries we would not have: penicillin [98], the discovery of cosmic background radiation [15], the microwave [98], X-rays [98], the pacemaker [98], matches [98], gunpowder [98], nuclear fission [98] and the chocolate chip cookie [100]. So it begs the question should we have a plan at all? Just explore! This is a risky proposition indeed but not one without merit.

In his book *Against Method* Feyerabend argued for the idea that scientists should have a dose of theoretical anarchism [34]. He showed that groundbreaking discoveries did not come about by adhering to a strict scientific methodology. Quite the contrary, in the case studies that he outlined all methodological principles were violated at some point.

This is not to say that researchers are able to do whatever they want, but if exploration is a part of research, then according to Feyerabend "anything goes." [34] And in this thesis, exploration will be the biggest theme. For I (Melvin Roest) explore. For I fiddle and break things. For I acknowledge my subjective perspective and try to advance the realm of scientific knowledge.

"What is the exploration you are on?" you might ask. In its most simplest form I am exploring the intersection of a certain framework (XIMPEL) and its relationship to education, user profiling and hypermedia (the biggest theme). This is choice is not entirely objective. In most cases it cannot be. I study at the Vrije Universiteit Amsterdam and there is a research group developing this framework. If I would have done my Master thesis at the CWI (Institute of Informatics in The Netherlands), then I would have most likely done similar explorations with its competitor SMIL. Moreover, this choice has a component of necessity. If hypermedia and user profiling in hypermedia (and by extension the web) had been more well-documented this exploration had been more focused on (computer science) education. In order to serve education, exploring these topics are needed.

Fortunately, XIMPEL has one saving grace in which I could make it sound objective, which is: for the last 10 years XIMPEL always attempted to be at the intersection of educational games and hypermedia. It furthermore never had a logging extension for user profiling [1]. It is most likely the only framework to have had this intersection. Therefore, asking the question of: "what is the relationship between XIMPEL and education?" does not sound out of the ordinary. Neither does "how does XIMPEL need to be improved for it to better serve education?" These were the questions I started with.

In my first explorations I extended XIMPEL to create a command-line tutorial. Would it be beneficial to make such a thing with XIMPEL compared to creating it from scratch (spoiler alert: yes)? In other explorations I simply built on future work from Stefan Bruins since it was (1) future work and (2) it was a key pillar in creating a command-line tutorial. There also were explorations in which I had a failed result relatively quickly. Yet another exploration could be seen more as traditionally academic. My penultimate exploration is entirely conceptual. And my final exploration is a showcase of hacking for future XIMPEL media type developers. Though, I have to warn you dear reader, there is a secret final exploration. After reading this thesis, would you know which one it is? Knowing it gives a hint

[1] Though when my logging extension was complete, I discovered that another logging extension had been independently made for XIMPEL in Norway by Dan Michael O. Heggø and Huggo Huurdeman. Fortunately, the focus of both logging extensions are different enough in aim and design to be separate useful contributions.

of the thesis writing process. For the process of scientific discovery itself should also be a science[2].
On my explorations I got into more questions. Such as: "what does interactivity mean for media and hypermedia?" or "is there a faster way to develop XIMPEL?" and as a last teaser "how could we improve the user experience of a XIMPEL presentation and make it less frustrating? Should it be less frustrating in the first place?" These questions will be more formally introduced in each relevant chapter.

1.1 What is XIMPEL?

Talking about a framework without explaining it does not sound like it is XIMPEL at all. It needs an explanation! More importantly, it needs a showcase. Multimedia frameworks and the like are best to be shown and explained afterwards. For the readers who never saw a XIMPEL presentation, Go to http://www.ximpel.net/showcase. The Abel Prize presentation and the tour of the Zaansce Schans presentation show various capabilities of what XIMPEL is able to do.
There is a reason that I display a link directly in the text and not in a footnote or as a reference. It would make both our lives much more simple if you would click on that link and see what XIMPEL is about. The paper model in academia is a bit archaic. Because of this I encourage you to be bold, click the link and *experience* what XIMPEL is about!
Now dear reader you have a choice. To read on, or to go to the link.
What will you do?
Who said that textual media has to be linear? You can jump to a video on the web and then read on. Click on http://www.ximpel.net/showcase if you want to see the showcase, or do nothing if you want to stay here.
If you do nothing you are missing out dear reader. Our visual cortices process information so much better than the little areas we have dedicated to purely reading.
I advice you to go to http://www.ximpel.net/showcase.
You have made your choice. We will move on[3].
The short explanation is that XIMPEL is a hypermedia player (linked media through overlays which are overlaying shapes such as rectangles) or an interactive video player (linked video through overlays), depending on whom is asked. The core feature of XIMPEL is that through XML, it is easy for non-programmers to play media and link it to other media by a mechanism called overlays. For example, one video is playing and one transparent square (an overlay) is in the upper right corner. A user clicks on it and another video with an image appears and both start playback from the beginning. That is the essence of XIMPEL.
When it comes to the XIMPEL framework the following stakeholders take a role in the production or consumption of a presentation. (1) XIMPEL core developers who maintain and expand the core of the framework. (2) XIMPEL media type developers who create plugins for the XIMPEL system. It is analogous to Wordpress plugin developers. (3) The author of a XIMPEL presentation. And finally, (4) users experiencing a XIMPEL presentation.
XIMPEL its main focus is to make the lives of XIMPEL presentation creators – also known as XIMPEL authors – easy and simple while nudging them to become a bit more literate in programming. In today's world it is "program or be programmed" [19]. The way XIMPEL achieves this is by having an own XML specification for creating a XIMPEL presentation, known as a XIMPEL playlist. XIMPEL authors only need to know how to write this form of XML and when they do, they are able to make a presentation. It is useful in the same way that finite state machines (FSMs) are useful as opposed to a computer: the application of FSMs are limited but within that limit, it is easier to create what one wants to create.

[2]Addendum: exploration 7 is now the final exploration! This exploration has been unplanned, out of the scope of this thesis and is therefore found in appendix A. The question about the secret final exploration pertains to the first six explorations.

[3]There is a mirror on `http://melvinroest.com/ximpel/showcase`

1.2 Thesis structure

Before we move on I will introduce you to a short history of hypermedia. This knowledge situates the context of what XIMPEL is, what it is not and which ideas from hypermedia are fundamental to XIMPEL. After that introduction we will move on to the explorations. For each exploration: we will first start with a short story on how I came about the idea to explore a certain topic and why it is relevant. Then I will continue to explore the topic. Each exploration will have its conclusion. After all the explorations have been presented a general discussion section concludes the thesis by summarizing it and by giving future work recommendations. As a last note, all the code is on my personal Github account (view the links at [87, 86, 84, 88, 85]).

Reflection - Standard Academic Style

In the penultimate thesis meeting, both of the assessing readers claimed that while the research questions regarding this thesis exist, they are a bit too unclear when reading the thesis. This is a similar comment to my assessment in the postface. Both readers understood the contents of this thesis. However, they are afraid that no one else will due to its free spirited writing style. It is not that this thesis is impossible to read. The problem is that this thesis is impossible to read within a reasonable amount of time. For this reason, they both suggested that reflection boxes are written per exploration and in the introduction. A reflection box looks like this, and are the most important pieces of text if the reader values a standard academic writing style or quick reading experience.

They would also call them irony boxes, since they broadly agreed with the assessment, made in this thesis, about the status of scientific texts. However, they learned that conformism has a necessary role to play. Conformism allows for survival upon closer scrutiny, and – according to one reader – probably makes some contributions of this thesis a bit more visible.

Consider the aggregation of these boxes a thorough academic summary. It is bigger than the abstract, but still readable within a reasonable amount of time. As for this particular reflection in the introduction, it will be stated: what the research questions are, the approach for answering them, the expected results and how this is academically relevant. Moreover, standard academic conventions will apply regarding the next reflection boxes.

Research Questions

Massive Online Open Courses are possible through a mixture of HTML, CSS and JavaScript. Certain types of non-linearity are not well-captured within HTML documents such as the notion of a playlist. For example, it does not seem possible to have multiple HTML documents within one file and then select which one to render. The ideas in the (perhaps) forgotten sibling of hypertext, which is hypermedia, does have such a notion and therefore seems to add useful mechanisms for delivering online education.

Perhaps the biggest reason hypermedia frameworks did not come to fruition has been because of a lack of funding and comparatively little effort, as opposed to the industry-driven HTML specification (e.g. Google, Mozilla and Microsoft). This is not only regarding hypermedia frameworks, this is even regarding HTML and the W3C. The browser vendors are collaborating and are in control of the HTML standard [49]. Nevertheless, a hypermedia framework that has survived in good condition during the competition with the current hypertext and HTML standard is XIMPEL.

Extending XIMPEL may give insights into the uniqueness regarding what hypermedia may have to offer. However, an extension for all domains would be out of the scope for this particular project and XIMPEL has already shown promise in education by being used in courses at the Vrije Universiteit Amsterdam since 2007. Therefore, the most important research question is: (1) how does XIMPEL need to be extended to contribute to online education?

During the explorations it became more apparent that having an implemented parallel playback feature trickled down into all kinds of manners in the XIMPEL framework. While this has been unintentional, a clear question arose with it. This question is relevant in most explorations. The second research question is: (2) what are the (technical and design) implications of parallel media playback?

Another theme that became apparent is that XIMPEL is not a pure hypermedia framework. A good example of a pure hypermedia framework is SMIL. However, XIMPEL has been created with non-linear storytelling and gameplay in mind. While storytelling seems to be aligned with hypermedia, gaming does not seem to be aligned with it. Therefore, in some explorations the third research question has been asked, which is: (3) what areas of research could XIMPEL benefit from, and how?

Question 1 will be present in all explorations. Question 2 will be present in most explorations. Question 3 will be present in some explorations. The research questions have been chosen on their orthogonality, meaning that the idea is that they go in relatively unassociated directions.

1.3 A brief Amsterdam-centric history of hypermedia

The research area that the XIMPEL framework connects with or situates itself in are: education, game-design, narrative structures, interactive video and hypermedia. There might be more areas, but previous research regarding XIMPEL has been around these areas [31, 4, 30, 8]. A chapter published in 2016 about XIMPEL is called: "XIMPEL for Education – inspiring creativity through storytelling and gameplay." Furthermore, XIMPEL presentations themselves demonstrate that they are a form of interactive video. A user is able to click on overlays and then jumps to another video which could be part of a narrative. Interactive video is a part of hypermedia and since more media items are able to be linked in XIMPEL one could argue that XIMPEL itself is a hypermedia framework or at the very least heavily inspired by ideals of hypermedia.

Because of this it is of utmost importance to describe this field. Hypermedia is the research field where fundamental questions regarding XIMPEL lie. Without the ideas of hypermedia, and by extension ideas surrounding the world wide web, there would be no XIMPEL. By portraying the historical roots, any reader is able to have an idea of how this research field has developed. On another note, to a

certain extent, the research field of interactive video is implicitly described a little bit as well since it is part of hypermedia.
Education, game-design and narrative structures are research fields of their own that are associated with XIMPEL but not the research context of XIMPEL. The connection of education, game-design and narrative structures to XIMPEL are that it is possible to create presentations or applications with XIMPEL with possible help of these research fields. These applications may show unexplored topics of areas of research within these research fields. Since not all XIMPEL presentations or applications are subsumed under these research fields, they will not be described from a historical perspective. Where it is needed, research from these areas will be used as reference.

1.3.1 In the beginning

In his essay As We May Think, Vannevar Bush envisioned what we would now call the information age. He wrote: "Wholly new forms of encyclopedias will appear, ready-made with a mesh of associative trails running through them, ready to be dropped into the memex and there amplified." [10] The memex was a hypothetical machine that would store all sorts of media through film. The storage mechanism resembles technology of the year the essay was written in (1945), such as microfilm. I would not be able to describe that mechanism well since my knowledge of mechanical technologies is very limited. Fortunately, that is not important.
What is important is that this hypothetical device was seen as a proto-hypertext system at the time. At the time, such a system was wholly hypothetical, just like any modern algorithm that has yet to be executed for the first time. Features that stand out are: the ability the connect sequences of media (they called it associative linking), being able to annotate media and retrieval upon cues rather than using an index. Bush focused on the idea of supplementing creative thinking, which he believed a non-linear media display and retrieval system would be able to accomplish.
As We May Think has been a highly influential essay. For example, it influenced Douglas Engelbart, for whom it helped to conceive: hypertext, the computer mouse and the computer display [32, 33]. During that time things were called differently. It was not a mouse but a pointer[4]. It was not called a computer display but a computer-driven cathode-ray-tube display. Apparently, seminal visionary essays about information technology of the future do write about how such technology should be implemented.
Independently, Ted Nelson, who coined the terms hypertext and hypermedia, has been influenced by the essay. He designed the first hypertext system called Xanadu. This first hypertext system, while old, still has ideas that compete with the dominant hypertext system today, the world wide web. From this perspective it is important to understand that: while the world has adopted a hypertext system, skepticism about the system still needs to be applied, debates about the benefits of it still need to happen. The Xanadu hypertext system implemented two way linking (e.g. no 404 errors) and version control management. It may still be useful for communities of people who are in dire need of these features. One of the reasons Xanadu did not get adopted is because the ideas have only relatively recently been fully implemented[5] – in 2014. Possible causes are: (1) being too early and (2) lack of funding.

1.3.2 What is hypermedia

Hypermedia is to media what hypertext is to text. Moreover, text is a form of media. The idea of hypertext itself is confusing since it can be seen as (1) text with links or (2) the HTML specification. HTML is a markup language for hypertext, but there are also images and since HTML5 also video and audio. With this recent addition, it could be argued that HTML5 has hypermedia elements, much more so than the earlier versions of HTML.

[4] Why did we move away from this term? A pointer makes sense!

[5] An easily digestable demonstration of the system can be seen in this YouTube video: `https://www.youtube.com/watch?v=KIOuRuvQ1Oc`

The conceptualization of hypermedia started in the early 90's, and during that time HTML was purely meant for (hyper)text [3]. Hence, the first definition of hypertext seems to be the intended definition by the authors of the early 90's. So a hypermedia system can be viewed as a system that is able to display all forms of media and all forms of media can be linked to each other in the way that users are nowadays familiar with the web (e.g. by tapping or clicking on areas that link to some other form of media).
Regarding the implementation of a hypermedia system it needs to have 3 components: (1) a description of its media components, (2) a way of defining relationships between media components such that they are connected and (3) a way of presenting these components on the computer screen. Media components could be blocks of text, animations, audio, video and user-defined media components. Links describe the relationships between media component. The idea of linking could be defined in a number of ways. The presentation can be determined by the hypermedia system itself through a rendering system.
Specifically for XIMPEL these 3 components have been realized by using web technologies. (1) the description of media components are described in XML within a document called the playlist, (2) overlays (hovering rectangles or other shapes that are clickable) can be laid over the media item playing within XIMPEL and when clicked, it stops the current media item and plays the media item that the overlay links to. (3) The Flash player used to be the rendering engine of XIMPEL. Nowadays, it is the browser for which HTML5, CSS and JavaScript are needed regarding the rendering process.
During the early 90's, research in hypermedia has been conceived by research on hypertext systems. Most notably, the Dexter Hypertext Reference Model has been influential in the development regarding popular hypermedia concepts [42]. It has been competing against the hypertext model of Tim Berners-Lee who created the world wide web as we know it today. The model itself came into existence after two small workshops on hypertext, which had representative participants for most major hypertext systems made during that time. The researchers behind the model, Halasz, Schwartz, Grønbæk, Kaj and Randall remarked that the model "is an attempt to capture, both formally and informally, the important abstractions found in a wide range of existing and future hypertext systems" [41].
The Dexter Hypertext Reference Model (hereafter Dexter Model) has three layers: the run-time layer (user interaction mechanisms), the storage layer (a network of nodes and links to those nodes) and the within-component layer (content and structures within hypertext) [41]. A big difference compared to the world wide web today is that links needed to resolve both ways [35]. If this was not the case, then if one hypertext component was deleted, every hypertext component linking to it would need to delete its link.

1.3.3 The Amsterdam Hypermedia Model

The Dexter Model gave rise to the Amsterdam Hypermedia Model. It is the predecessor of what was going to be the most popular hypermedia framework for about twenty years. The most popular hypermedia framework SMIL eventually became a W3C recommendation [71]. The Amsterdam Hypermedia Model began by addressing the limitations of the Dexter Model for describing hypermedia. Hypertext and hypermedia are not the same so it seems obvious that these limitations should exist.
Researchers on the Amsterdam Hypermedia Model, Hardman and Bulterman, stated that The Dexter Model "has no notion of time beyond the within-component layer," [42] and further stated it has "no way of specifying higher-level presentation information, and no notion of context for an anchor" [42]. Context here means: does the whole presentation change or just a part of it? At the time their own media model, the CWI Multimedia Interchange Format (CMIF), had the drawback that it did not specify links but did address the other issues that the Dexter Model had. The solution was taking inspiration from both and it could be argued that The Amsterdam Hypermedia Model in its shortest (and not fully accurate) description is the combination of CMIF with linking capabilities inspired by the Dexter Model.
The Amsterdam Hypermedia Model (AHM) is about combining multimedia in such a way that it forms a presentation in which there is the possibility of (non-linear) choice at certain moments. For example, one could watch a penguin video, see a header text penguins and then be presented with the

choice to let them dance or to let them walk a bit more. One could then interact with the video by stating the choice and then a subsequent video with new header text would be displayed. We will now look on what the elements of the AHM are that allow us to create such a multimedia presentation.

1.3.3.1 Components

The AHM has multimedia resources which can be an image, text, video or music. These resources are called atomic components. Another type of component is a composite component which can either be a component that plays atomic components in parallel or a component that allows the user to select one atomic component to be played. These are two respective types (parallel or choice) of the parallel component. This design feature is ~~stolen~~ direct inspiration regarding parallel media play in XIMPEL (see chapter 3).

1.3.3.2 Time

The AHM had a way of defining time relationships between siblings and children in a composite component in the form of temporal relations. These relations are necessary, otherwise no one knows when something ought to be played. For example, each child in a composite component has a start time offset and a duration. An example of this is when a video plays, then when subtitles are played, they need to start 5 seconds later than the video. The AHM also has the capability to define time relationships between different (non-family) components via synchronization arcs, which are optional. A synchronization arc allows non-siblings to have a timing relationship to each other. For example, it can be determined when they should play one after another, what the variable length of this playback may be, to what extent the system should enforce these rules. For more information see paragraph 3.1 of [9], another good resource is figures 5, 6 and from [42].

1.3.3.3 Channels

How is certain media presented? Is there space for a menu? How does that look like when a video is played? Can a video play full screen when there is a menu? These questions are all about the general styling of a multimedia presentation. To answer these questions, the AHM introduced the idea of channels. Channels are “output devices for playing multimedia items” [42]. Channels play an atomic component, for example: one channel could be playing videos, whereas another channel would display text. It is not possible for a channel playing videos to also play text. Also, a channel can only play one atomic component at a time. Examples of channel definition are: the default font size, size of the window, z-index layers, and position of the multimedia element (atomic component). One could already imagine that generic layouts are a good use case for channels since channels are reusable.

1.3.3.4 Links

Since the Dexter Model is different than the current model of the world wide web, the idea of links here are slightly different as well. A link in the AHM can have a context. A link context allows for specification for which part of the document needs to change. Specifying a link context has the idea that when a presentation changes, it only changes what it needs to change, allowing for reusability of the components in the AHM. Links ought to be clicked, and therefore links introduce choice. For this reason, the choice component is an obvious but necessary requirement.

1.3.4 SMIL

“Most constructs in CMIF and SMIL are explicitly modeled by the Amsterdam hypermedia model” [71]. Therefore, SMIL could be seen as the successor of the proposed AHM. SMIL has been the standard on synchronized multimedia until 2013. While SMIL has been supported by a lot of well-known companies, it has not been supported by all of them. Unfortunately, the SMIL working group disbanded and SMIL is not updated anymore [96, 97]. Moreover, future ideas of SMIL seem not to be included as a standard, such as Time Style Sheets [56]. It seems that HTML5 bares the burden of bringing hypermedia to the masses [97], albeit much less feature rich. An example of a SMIL presentation can be found here [20] and a lecture about SMIL 3.0 and how the XML tags look like can be found here [50]. Current SMIL presentations need an external player or browser plugin.

1.3.5 XIMPEL

In 2007 a new hypermedia framework started. It is called the eXtensible Interactive Media Player for Entertainment and Learning [4, 31], or in other words: XIMPEL. Regarding to the origins of the acronym: simpel is the Dutch word for simple. It was created in order to design a game about climate change. The goal was to lead the debate away from pathos (emotional arguments) and towards logos (logical arguments) [30, 31]. Before XIMPEL was conceived, the idea was to create an immersive game with the Half-Life SDK about climate change. However, it seemed infeasible because of too few resources. Therefore, the team created the XIMPEL framework and decided to created a game purely based on videos still allowing for what they called a *poor man's immersion*.
Borrowing ideas from hypermedia for game development was perhaps overlooked at the time since hypermedia formats – including its platforms – such as SMIL were focused on a pure structure of hypermedia. Where probably most (if not all) hypermedia platforms focused on the formalities of a hypermedia system. The XIMPEL platform focused on exploring possible applications. XIMPEL was created out of necessity for an educational game that would otherwise take a long time to make. From this perspective, XIMPEL should be compared to game development engines and it is one of the most unconventional game development engines in existence.
Over the years the framework has evolved from its Flex SDK/Flash roots to a HTML5/JS/CSS version. One creator (Anton Eliëns) of XIMPEL taught every year how to use it to first year information science students[6]. These students created at least one presentation with it in a group of 2 or 3. In some cases, students ran into technical problems and either extended XIMPEL themselves or through the help of Winoe Bhikharie. By doing this Anton and Winoe unwittingly explored the question: what can hypermedia mean for serious games? Moreover, what hypermedia presentations are relevant and interesting? While XIMPEL is a hypermedia system, the questions explored with it are about serious games and non-linear narratives.
Current future work regarding XIMPEL is expanding the framework for usage of big tablet installations at museums. One example of that is an app made for the Abel Prize, which is a prestigious prize for mathematicians [46]. Workshops are given in Norway to librarians for whom the XML format is easy to work with.
So XIMPEL is useful for:

- education,
- prototyping digital games,
- video narratives/interactive storytelling,
- storyboarding,
- big tablet installations

1.3.6 How XIMPEL works

XIMPEL has subjects. These subjects can be seen as states that the XIMPEL presentation or XIMPEL application is in. Each subject is able to be linked to and has its own unique identifier. The goal of these subjects is to play one (1) media item at a time. This concept is called sequential media playback. A media item is an instance of a media type. Examples of media types are: audio, video, images, text or a custom-defined media type (which can be anything). The links in XIMPEL are called overlays. The default shape of an overlay is a white rectangle, this shape and color can be altered. When a user clicks on an overlay, then the XIMPEL framework destroys whatever is displayed in the current

[6] I was a student in one of these years

subjects and starts to play all media items that are associated with the subject id that the overlay links to.
So the important actors in a XIMPEL presentation are: subjects, instantiated media types known as media items and overlays which act as a linking mechanism to other subject when a user clicks on it. In this thesis the core will be extended for subjects to play multiple media items. This extension is presented in exploration 2 and is called parallel media playback.
Advanced features regarding XIMPEL are: (1) the ability to create quizzes, (2) associate any score to any key when a user switches to another subject or clicks on an overlay and (3) the ability to have conditional subject rendering by having the possibility to create simple if-statements when a user clicks on an overlay or arrives at a subject. The advanced features deviate from the idea of hypermedia and allows XIMPEL to be between hypermedia and gaming.
The official documentation is to be found at `http://www.ximpel.net/documentation/`. The up to date documentation (for now) regarding the extended XIMPEL of this thesis is to be found at `http://melvinroest.com/ximpel/documentation/`. In the programmer tradition of not repeating myself, see these two resources if you want to know more technical details.

1.3.7 Now

Hypermedia is still here. One of the creators of SMIL (Dick Bulterman) asked himself how the ideas he knows from SMIL could be useful for CSS. He proposed additional CSS attributes to the W3C [56]. Hypermedia and hypervideo systems have been ported to HTML5/JS/CSS and are still used today [61, 12, 11].
All systems have their own unique spin on it. Some emphasize videos [61] or mix it with it with other ideas of software engineering (e.g. learning objects) [11]. Others focus on hypermedia systems [12]. To clear out confusion, this thesis is not about hypermedia APIs. A concept in API-design that allows for REST-based APIs that are more flexible. For example, it is easier to avoid backwards compatible breaking changes compared to standard REST API-design.
With the advent of HTML5 the role of hypermedia will be different. Considering an HTML document merely as a document and not as a possible software application would be short-sighted. Thanks to JavaScript it is possible to create: Linux in the browser [2], create a programmable computer through the Game of Life [76] or create anything else that one could imagine. In short, it used to be very difficult to create any application – requiring technologies like Flash. Now this is not the case anymore.
However, there is a role for hypermedia frameworks. While everything is possible with HTML5/CSS/JS, this does not mean it is the right level of abstraction. A hypermedia framework forces any developer to think about non-linear narratives. Any developer framework that forces the developer to create a subset of all computer applications could by extension be called a creative constraint framework with regards to software creation. This is seen in industry as some companies implement hypermedia ideas, particularly pertaining to non-linear video. Companies such as: Zentrick[7] [103], Wirewax [99] and Eko [28] use proprietary frameworks and it shows the importance of creating open source hypermedia frameworks.

1.4 Explorations

Now that we have some idea of where the philosophical forefathers of XIMPEL came from it is time to explore! The conscious choice of having no methodology has been informed by the philosopher Feyerabend. There are two starting questions. These are: what is the relationship between XIMPEL and education? And how does XIMPEL need to be improved for it to better serve education?
The idea of the explorations are that once an exploration is done, it may give rise to other questions that need to be explored. One exploration question came up by daydreaming about XIMPEL, this is exploration 4. I took a course in user experience design and realized how important a good user experience

[7]They have an old application that they made for Unilever `http://watch.zentrick.com/m5tMjr/`

is. During my time as a student I noticed that a lot of students experienced frustrating moments while experiencing a XIMPEL presentation. It begged the question: is it possible to (semi)automatically detect user frustration within XIMPEL presentations? One assumption is that this type of detection may help the user experience for XIMPEL presentations. The chain of how one exploration caused another exploration happened as follows: exploration 1 (from the starting questions), exploration 2 (from exploration 1 and from the starting questions), exploration 3 (from exploration 1), exploration 5 (from exploration 2) and exploration 6 (from exploration 2). For some preservation of the creative context: what went through my mind will be a short story per exploration as the first few paragraphs. Among other things, explorations have one key difference with experiments in that they are unfortunately not reproducible. This is a blessing and a curse. It is a blessing because I have encountered zero papers (!) that reported the reproduction of a result of another computer scientist. It is a curse, because I am admitting that a certain form of objectivity is lost. To be fair, many papers in computer science seem not to be reproducible because not enough information is presented in order to recreate a proposed system. And the nature of the field gives the impression that engineering replaces science. One could argue that working technology is a certain truth on its own, which is what science is about. Fortunately, there are other ways to contribute towards the academic discussion for any given topic. A few examples are: creating tools, new relevant examples of application regarding a particular system or simply asking questions that have not been asked but which are relevant. In the following paragraphs the specific contributions per exploration are stated.

In exploration 1 The implementation regarding the creation of a command-line tutorial in XIMPEL is presented. The contribution of this exploration is showing the boundaries of XIMPEL when a developer discards the hypermedia ideal and solely focuses on education. The implementation of the command-line tutorial itself is a contribution to the XIMPEL framework, including its programmed server. Moreover, there is an attempt to answer the question: when is something hypermedia and when is it not?

In exploration 2 the future work that was presented in the thesis of Stefan Bruins [8] which is creating a parallel player for XIMPEL. Bruins focused on the main question whether the XIMPEL framework could be implemented with HTML5, CSS and JavaScript. In this exploration I focus on the question on how to architect and implement a parallel media player within XIMPEL. The contribution is an implementation and architecture overview of a parallel media player. Furthermore, the parallel media player in XIMPEL creates new questions such as: what if a media item survives a subject switch (called media item subject switch survival, i.e. MISSS)? Or: how does one do timescrubbing in XIMPEL? These questions are explored in a new exploration.

In exploration 3 I assess whether XIMPEL can be recreated with ReactJS (a front-end JavaScript library created by Facebook) in an effective manner. In this exploration I successfully and unsuccessfully recreate XIMPEL with ReactJS as opposed to plain JavaScript and jQuery. The contribution of this is most of XIMPEL recreated in ReactJS and a relative full description of the creative process of getting there. Another contribution is that this could be seen as a case study of how of observing architectural patterns in the XIMPEL specification leads to finding a library that has more or less the same patterns in its developer philosophy. It furthermore, implements ideas from almost all explorations and has ended as a recreation of XIMPEL and also as my vision for XIMPEL, which is more web-based than the current version[8]. For the attentive reader this is secretly both exploration 3 and the final exploration because of the messy timeline. For naming purposes, it is called exploration 3 since it started as the third exploration and was the last to finish.

In exploration 4 I explore how to (semi)automatically detect frustration of users within XIMPEL presentations. The contribution of this exploration is an implemented logging extension for XIMPEL. The logging extension captures: facial expressions, mouse clicks, mouse moves and the history of what the user already has seen. There is furthermore a conceptual description on how to detect frustration specifically for the XIMPEL framework.

[8]While not described in this thesis, the positioning of media items could be done entirely with CSS. In the architectural design I kept an eye out for CSS-support, but I was not fully aware of it that I did.

In exploration 5 I wanted to implement a time-scrubbing mechanism, but slowely and surely I realized that there are many twists and turns when it comes to time-scrubbing in hypermedia applications! It suffices to say that architecting and implementing time-scrubbing for single videos is much more straightforward than multiple videos (and other forms of media) that share a certain relationship with the other media elements. The contribution of this exploration is an analysis of which design choices one could possibly take in order to create time-scrubbing in XIMPEL. Analyzing these design choices was equally rooted in: literature, rationality and imagination.
In exploration 6 I write about how I extend XIMPEL with one extra parameter in a method definition and show how this extension can lead to some serious media type hacking for media type developers. The whole chapter is written in story form, in order to not tire the reader out. To showcase this, I implemented a YouTube media type that does not immediately detach from the DOM when a new subject is clicked on. The contribution of this exploration is: (1) the introduction of a new concept – partial subject refreshes known as media item subject switch survival (MISSS), (2) a showcase on extending media types so that this functionality becomes available and (3) an implemented YouTube media type that supports partial refreshes.
In closing, the structure of any exploration is as follows:

- short story what went through my mind,
- introduction of exploration,
- the exploration itself and
- conclusion and future work.

2

Exploration 1: creating a command-line tutorial in XIMPEL

Looking at the structure of many MOOCs, it could be observed that they all have a simple video playlist and then follow up with some multiple choice questions or a coding assignment within an online code editor. XIMPEL has the ability to create videos through means of a playlist and is able to create multiple choice questions. It does not have a code editor. The question arose: is it possible to recreate a command-line tutorial with XIMPEL? Because if it is, then it will be a lot quicker to create such an application compared to creating it from scratch!
On a related matter, with the implementation of a command-line tutorial, XIMPEL could compete with computer science courses on Coursera. It could not compete on design but on pragmaticism, not on flexibility but on programming simplicity, not on comprehensiveness but on development speed. With this extension, XIMPEL occupies a unique niche of hypermedia and education.
By recreating a command-line tutorial in XIMPEL I stumbled upon a question. A command-line interface is not a form of media. A command-line interface is an application. So by recreating this in XIMPEL I am mixing a computer application with hypermedia. What does this mean? Am I now leaving hypermedia? Are we mixing hypermedia elements with a computer application? Could XIMPEL still be considered a hypermedia framework? Should XIMPEL be a hypermedia framework in the first place? While it is clear that the advanced features of XIMPEL leave the realm of hypermedia, it does so for a specific use case, which is gaming. Creating a command-line interface is not about digital games, it is an application. Therefore, this is a philosophical design change to the framework, and investigating it seems important to XIMPEL, but also to have a better understanding of hypermedia in general.
The questions that I will formally focus on in this exploration are: how would XIMPEL need to be extended in order to create a command-line tutorial? And what are the implications of mixing a hypermedia application with another (non hypermedia) application with regards to hypermedia as a field?

Research Questions and Contribution

All research questions (1, 2 and 3) are relevant for this exploration. Most notably, question 1 and 2 are relevant regarding this exploration.
The first research question is: (1) how does XIMPEL need to be extended to contribute to online education? A specific contribution has been made regarding online computer science education. Creating a terminal media type helps computer science education and is a basic skill set that computer science students need to learn. Furthermore, it is possible to learn many more skills that computer science students could benefit from via the command-line such as learning other programming languages. Finally, educators have the ability to set up very specific programming environments that students do not need to install. By detailing the architecture and implementation it helps for more developers to create a similar tutorial-like applications since no clear guide seems to exist.
The second research question is: (2) what are the (technical and design) implications of parallel media playback? In this particular case, it means that video is able to play alongside a command-line tutorial. Furthermore, such an implementation is trivially simple to implement for XIMPEL presentation (or application) authors.
Finally, (3) what areas of research could XIMPEL benefit from, and how? In this particular case, the advantages and disadvantages are discussed of going beyond hypermedia. While not discussed in this exploration, going beyond hypermedia positions XIMPEL as a framework to remix the web through iframes (see the future work section of the discussion). This is especially the case compared to other hypermedia frameworks. Custom media types such as a command-line interface, contribute to this remixing of the web by allowing for more specialized content.

2.1 Extending XIMPEL to create a command-line tutorial

Did you know that the XIMPEL is an acronym for eXtensible Interactive Media Player for Entertainment and Learning? Neither did I at one point. The extensibility of this media player will be demonstrated here. Together with the core changes I made in the framework, it is even more extensible!
So is it possible to recreate a command-line tutorial with XIMPEL? The answer is yes. Our research question has been answered. I suppose we are done.
Which command-line tutorial are we recreating you might ask? From CodeCademy.com of course! It is one of the leading organizations for online programming education. To give a feel for how CodeCademy looks like I recommend you to try the command-line tutorial yourself. For readers who are reading this on paper, figure F.1 will show a screenshot from a part of the command-line tutorial.
When I first started recreating this command-line tutorial I realized two things. (1) It would be interesting to see how such a tutorial would be with video besides it instead of text and (2) I did not have time to recreate the whole tutorial so by using my artistic freedom I recreated the first two lessons which are about how to use the `pwd` and the `ls` command. By doing this the framework is going towards online education. Media itself is in most cases already a form of education, but by creating a command-line tutorial the XIMPEL framework moves more towards the interactive part of education.
The command-line tutorial requires two things. XIMPEL needs to be extended in order to play multiple media types. It could be possible to create a media type that plays video and has a terminal emulator which is too tightly coupled. This requirement necessitated exploration 2. The other requirement is that XIMPEL needs to be able to have a terminal of some sort. In this exploration I will focus on how I created a terminal application within XIMPEL, assuming parallel playback is possible.

2.2 Architecture of the command-line media type

In order to create a proof of concept I started to reverse engineer how other command-line web applications were created. I took a look at: R-fiddle and CodeCademy. I also took a look at JSfiddle since I entertained the idea of executing JavaScript as a code tutorial. I quickly stopped entertaining that idea when I realized I had to use the `eval` function and would create a whole host of additional security hazards within XIMPEL.

R-fiddle has the following architecture: it has a client, a server and it communicates via websockets (seen via Chrome Developer Tools). It spins up a virtualized container such as docker or vagrant in order to minimize security risks such as a shell with too much privileges on a file system that has sensitive information. By using virtualized containers for each new client, there is no file system with sensitive information and the shell does not have too many privileges [22].

Codecademy sends an authentication token to a server along with other data via an xhr-request (in JavaScript parlance: they use AJAX) and send other relevant data with it in order to manage at what course the user is and if the user is logged in. The server answers back in JSON. Also Codecademy uses websockets to communicate with a terminal application (see figure F.2). Presumably, they use some form of sandboxing.

The common architecture in common with both websites are: client, server, websockets as communication and presumably sandboxing. So the first step I created was creating a command-line tutorial in HTML/JS/CSS. Once that was completed I realized I needed to implement the client-side facing part of the terminal as a XIMPEL media type. Moreover, that media type needs to connect via websockets to a server.

The current implementation has its server-side part implemented in NodeJS with the web server micro framework ExpressJS, which currently receives input data from the XIMPEL terminal media type via websockets. It furthermore spawns a bash shell and sends the data back to the terminal media type. Once it is there, the output of the bash shell will be displayed in the XIMPEL presentation. It has to be stated that this implementation is a proof of concept. It is a prototype. Therefore, it is not secure. The bash shell is not sandboxed with, for example, Vagrant or Docker. Figure F.3 illustrates the client-server communication.

To conclude this section, creating a command-line tutorial has a couple of implications for XIMPEL. First, a lot of server-side web technology needs to be added such as NodeJS, ExpressJS and websockets. Furthermore, the devops tool Docker also needs to be understood. These web technologies are relatively new and not understood by all web developers and also not all web developers of XIMPEL which makes it harder to maintain XIMPEL. Second, it seems that XIMPEL is really suited for a micro service architecture. The reason for this is mostly because some media types require a server. To have a monolithic server for all possible media types in existence would harm the extensibility of XIMPEL, with microservices this danger is mitigated. Third, it is the question whether creating it as a media type is the way to go. XIMPEL also supports iframes, which means it could also have been implemented as an iframe for more control over the terminal.

Finally to bring it back to education: XIMPEL its strength is in presenting non-linear paths, which could also be done for command-line tutorials. A serious student could perform its own manual dynamic difficulty adjustment by clicking the overlays which present possible harder or easier command-line challenges. The non-linearity of hypermedia is so built into the framework of XIMPEL that the idea is always a no brainer when one works with the framework. Yet, the idea of non-linearity is not seen in other popular online educational websites which means that hypermedia frameworks have an advantage in that aspect.

2.3 When is something hypermedia and when is it not?

Hypermedia frameworks claim to be frameworks for rendering media items to the screen. While this is true, in the case of SMIL and XIMPEL they do much more. This is confusing, because two questions arise from this: (1) what is hypermedia? (2) Are SMIL or XIMPEL hypermedia framework or not? With regards to XIMPEL the answer could be argued to be a clear no, since it is inspired by them but it is not one itself. It could also be argued that it is a hypermedia framework at its core, but not in its totality. SMIL 3.0 on the other hand is capable of interacting with parts of a web application and partially also created parts of a web application (see [50] around minute 25, it is a video). If all hypermedia frameworks are doing more than just enabling hypermedia content, then it needs to be established what hypermedia is and what it is not. Being able to make a distinction in that provides better vocabulary which aids categorical comparative analysis, much in the same way that the distinction between gender or sex provides categorical comparative analysis (e.g. females compared to males, two categories being compared). It may also shed light on the usefulness of non-hypermedia elements in hypermedia frameworks.

To begin answering the question: in order to answer when something is hypermedia and when something is not, it must first be asked what it means for something to be media on a computer. A literature search has been done, but unfortunately no relevant literature has been found. It is quite strange to not find literature on a very basic question such as: “what is media?” or its more slightly complicated version “what is hypermedia?”

The characterization of media falls into three distinct ideas, as I have seen so far: (1) mass media, (2) computer (multi)media and (3) social media. Common examples for each three is common knowledge for most people. For example, television, radio and the newspaper are seen as forms of mass media. Video, audio, images and text are seen as fundamental forms of computer media. Coincidentally, the two have a very strong mapping with each other. For example, text and images appear in a newspaper. With social media, the notion of what media is becomes more vague. Social media can be characterized as platforms where people are social on the internet. The usual suspects are: Facebook, Twitter and Instagram, among others. But public internet forums, newsgroups or a guestbook could also be seen as social media. In all of these cases it is possible for people to communicate with each other through the use of computer (multi)media such as: video, audio, images and text (including emojis). Interactivity is new here and that is because with the two more old-fashioned forms of media, it always was a one way communication.

Up until this point it could be argued that computation and manipulation of digital objects – other than writing text – do not need to be considered media. However, people call games media as well. And in games, everything is possible!

For our purposes, however, the characterization of media will not contain computation. A spreadsheet application is not a form of media and neither is a calculator. Other not media examples are: Finder, my text-editor, the command-line, a web browser, TeamViewer, Evernote, my FTP program, my mindmap editor, my colormeter, my screencast recording software, my vector graphics manipulation tool, Unity3D, Java, Preview, a Git viewer, Activity Monitor, my time tracker and paintbrush. This is more than eighty percent of my visible applications in my Mac OSX dock. Two applications could be considered social media, which are Colloquy (an IRC client) and Slack. However, since the information is not public (like guestbooks and forums) they may also not been seen as social media.

Finally, what all forms of media have in common is that they use the senses. For example, one hears twigs cracking slightly, above the soft noise of a fuzzing wind; furry hairs and two humongous toes are seen in the corner of an eye; a huge arm grabs your puny wrist in comparison and in a shock you realize: “It is Bigfoot![1]” If this would be a real experience, then it would not be (multi)media. However, if this experience is fabricated through: screens, paper and other various forms of technology then it is (multi)media. The multimedia researcher Anton Eliëns agrees with this idea. He says that multimedia is everything that uses (sensoric) media experiences such as vision and sound [29].

So what is hypermedia? Every form of media that uses vision and sound and does not contain a form of computation other than the computations that are relevant for displaying the media itself. Furthermore, hypermedia is linkable, just like hypertext.

[1]As much as I love furry huggable creatures I do not believe in most of them other than teddy bears for sale at the toy store.

This means that by creating a command-line tutorial in XIMPEL, the framework has expanded beyond hypermedia. It already has been expanded beyond hypermedia in recent editions by having iframe support, so the idea that XIMPEL is only an hypermedia framework is not the case anymore.
XIMPEL is, however, mostly a hypermedia framework in its philosophy, since very little programming knowledge is needed in order to create hypermedia presentations. For advanced XIMPEL users, it is more than only hypermedia. It is possibly anything that a web application could be. But XIMPEL does treat its extensions as media, even if it is an application like a command-line. This means that every extension (application or media) is able to: play, pause, stop and resume. Since it is arguable that XIMPEL is not a hypermedia framework the play, pause, stop and resume feature has less of a justification as well.
The conclusion for this part of the exploration is an unusual one. It is strange to wrap web applications like a terminal into custom media types. It does not seem useful to add multimedia features to this (i.e. play, stop and pause functionality). While it is possible and it works relatively well, the play, pause and stop functionality seems redundant. From a development point of view, it needs to be considered to what extent stop, play and resume functionalities should be obligatory for media types. On the other hand, a fairly easy solution is to leave an empty function body for the resume functionality.
A second conclusion is that the power of XIMPEL is its playlist. For example, just by typing `<terminal>` one is able to create a complete terminal experience! This idea sparked exploration 3: recreating XIMPEL with ReactJS of which the rationale will be discussed in its relevant chapter (chapter 4). One of the ideas in ReactJS is similar compared to the XIMPEL playlist, which is: everything is a component. Everything in the XIMPEL playlist is a component as well.
In closing, the distinction of when something is hypermedia or not justifies and delineates the relevance of hypermedia frameworks. Hypermedia frameworks keep themselves relevant by having extensibility with application components or interactibility with application parts of any platform it resides on. Furthermore, hypermedia frameworks with extensibility in mind seem not to trade any performance penalty compared to pure hypermedia frameworks, other than losing development and design focus on hypermedia features. However, a not well-known trade-off is that by doing this, a hypermedia framework becomes more complex and one could study whether it is not pedagogical to keep hypermedia frameworks simplistic, or fork hypermedia frameworks to keep the forked version simplistic. In this section it has been found that sticking too much to the hypermedia ideal harms extensibility. An unexpected conclusion is that the power of XIMPEL is in its playlist. Hence exploring this question led to the exploration of a software engineering question described in chapter 4.

2.3.1 Conclusion

How would XIMPEL need to be extended in order to create a command-line tutorial? A server-side architecture needed to be researched, implemented and a parallel player needed to be created (see chapter 3). This changes XIMPEL from being a front-end framework built with hypermedia philosophy to being a full-stack framework built with a hypermedia philosophy. It is perhaps obvious that the relationship of XIMPEL pertains to online education. What is perhaps less obvious is that XIMPEL can also pertain to offline education, by easing students into environments where they do not need to install anything. They simply need to watch or interact with a XIMPEL presentation or XIMPEL application.
In retrospect, while this shows the extensibility of XIMPEL, there is a much simpler way to do this. Unfortunately, the idea does need to pop in my mind for me to realize this on time, which did not happen. The idea is: it is possible to use the `<iframe>` tag to link to any service that provides an online terminal, such as [2] which emulates the x86 processor and allows to run Linux purely on the client. Therefore, it could have simply been implemented via the `<iframe>` tag and a 3rd-party service. Moreover, this leads to the realization that ever since the `<iframe>` tag has been added, it generalizes the `<youtube>` tag and makes it more or less superfluous. To think of a `<youtube>` tag as a video tag would be wrong. It is a *3rd-party service* that happens to provide video. The fact it is a 3rd-party service is more important. Tags that stay relevant are tags related to local: text, images, video and

audio. These stay relevant, because it takes more work to implement them via an iframe, compared to typing `<video>` and filling in the respective attributes of it. In that sense, media types should be seen as primitives. The `<iframe>` tag, while more advanced, is part of these primitives. Knowing this earlier would likely have meant that this exploration would be about remixing the web with iframes instead of architecting and implementing a terminal media type.
Does XIMPEL need to have custom media types at all? Why not just use an iframe? Potential downsides of iframes are: responsiveness (media queries need to be added), bookmarking the iframe only, not knowing whether a bug is in an iframe or the actual page and possible SEO issues. Most of these issues do not apply to a XIMPEL application that much. The only reason for which custom media types could stay is to have a very fine-grained control over a certain form of media. For example, the `Youtube` media type could have a pre-loading feature, which is likely harder to achieve by using an `<iframe>` tag within a XIMPEL playlist.
Another conclusion relates to the question: what are the implications of mixing a hypermedia application with another application? It leads to the realization that a pure hypermedia approach has more disadvantages than advantages. This furthermore means that research should be done into the interplay of hypermedia and non-hypermedia elements in a hypermedia framework. Many hypermedia or hypermedia-like frameworks are use-case driven. As future work, an exhaustive literature review of comparing hypermedia frameworks and its use cases, will shed light on a possible more abstract or generic classification of topics for which hypermedia frameworks are useful for.

3

Exploration 2: extending XIMPEL for playing media types concurrently

While reading Stefan Bruins his thesis [8] I detected a form of sadness in it, a regret perhaps. There was a longing to play media items in parallel. Everything was set for it: the architecture expected it, Stefan wrote about it in his thesis and even in the comments of the XIMPEL source code it was written about. However, it was not there. And in a pirate voice I would like to say: Fortunately, it is I, Melvin Roest who has read his thesis and is excited to continue his work! Let's continue in a normal tone.

Let's start from the beginning: Bruins explored in his thesis whether XIMPEL could be ported to the web without the use of plugins like it was programmed in the past (with the Flex SDK, AS3 and Flash). In other words could it be ported to the web with: HTML5, CSS and JavaScript? The answer was a resounding yes [8]. Bruins also suggested future work to be done on the XIMPEL platform. One of the most prominent examples is the creation of a parallel media player. XIMPEL has been architected with a parallel media player in mind. Therefore, programming a parallel media player will give an even more conclusive answer to his research question of how one can be programmed. Furthermore, implementing a parallel media player will allow us to have a closer look at possible technical difficulties regarding HTML5 and attempting to play multiple videos in sync.

On another note, creating a parallel media player gives much more insight as to what hypermedia is. Simply creating a framework that plays one media element at the time is a rather limited experience in the form of hypermedia. Creating a parallel media player for XIMPEL would mean that more video and audio and custom media types could be played (note: it was already possible to display multiple images and text via the overlay mechanism and there was iframe support). Parallel media playback is possible in SMIL since version 1.0. However, SMIL and other hypermedia frameworks do not seem to offer a plugin like extensibility which means that the implications of parallel media playback and application media types (e.g. `<terminal>`) are left unexplored.

Research Questions and Contribution

In this exploration the main question has been: what fundamental future work regarding hypermedia exists regarding XIMPEL? The answer to that has been obvious from the start. The answer is: parallel media playback. This means that the question implicitly transforms into an implementation question of: what are the technical challenges to create parallel media playback? It relates to research question 2 in the sense that an implementation of parallel media playback directly gives a surface level answer on the technical implications of parallel media playback. Remember, research question 2 is: "what are the (technical and design) implications of parallel playback?" The implementation issues of creating such a feature sets up the stage in order to be able to conceive an answer regarding the technical or design implications regarding parallel media playback.
For comparison, in exploration 3 a different approach has been taken to uncover the technical challenges of parallel play. The architecture of exploration 2 has not been used as a reference. Despite that, there are similarities regarding the implementation of the parallel player.
Parallel playback is a necessary condition in order to have more expressive power regarding online education content creation. The implemented solutions in exploration 1 and 6 depend on parallel media playback. Exploration 1 and 6 both have their own contributions to online education. Without parallel media playback this would not have been possible.
Contributions of this exploration are (a) a simple check on the claims of previous work by Bruins [8]. One minor claim by Bruins turned out to be false. (b) A description of the implementation on parallel media playback in XIMPEL. (c) An actual implementation (see Github [86]). (d) Setting up the necessary foundation in order to ask research question 2. From a research standpoint the contribution is very humble since parallel media playback exist in other hypermedia frameworks. However, (e) the architecture of this particular kind of parallel media playback is likely to be different, which adds to the body of knowledge regarding software architecture and parallel media playback in hypermedia frameworks.

3.1 Architecture of parallel playback

By looking at the architecture of XIMPEL and how the sequence player was implemented, implementing a prototype of the parallel player has been more doable than without it. However, there were a couple of caveats in the original architecture of XIMPEL as stated in Bruins his thesis (see figure F.4). First of all, the figure suggests that a `SequencePlayer` is able to play another `SequencePlayer`. After testing the following playlist (see playlist 3.1) this is demonstrably false. Fortunately, it is false because there is no need for a `SequencePlayer` playing yet another `SequencePlayer` that will then play media types in sequence. The reason is: one `SequencePlayer` already plays media types in sequence!

```xml
<ximpel>
<config>
    <enableControls>true</enableControls>
    <controlsDisplayMethod>overlay</controlsDisplayMethod>
</config>
<playlist>
    <subject id="lesson1">
        <sequence>
            <sequence>
                <message text="hey" />
            </sequence>
        </sequence>
    </subject>
</playlist>
</ximpel>
```

Playlist 3.1: This playlist contains a nested sequence tag and therefore does not work. This demonstrates that the architecture in figure F.4 as outlined in Stefan Bruins [8] his thesis (fortunately) does not work as described.

```
<ximpel>
<config>
    <enableControls>true</enableControls>
    <controlsDisplayMethod>overlay</controlsDisplayMethod>
</config>
<playlist>
    <subject id="lesson1">
        <sequence>
            <message text="hey" />
        </sequence>
    </subject>
</playlist>
</ximpel>
```

Playlist 3.2: This playlist does work and serves as a counter example to playlist 3.1 which has a nested sequence tag.

The current architecture has been updated to reflect the implemented changes as well as the intention of XIMPEL (see figure F.5). In its current architecture the XIMPEL player always calls *the* `SequencePlayer`. And a `SequencePlayer` can either call a `ParallelPlayer` or a `MediaPlayer`. A `MediaPlayer` plays a `MediaType` (e.g. video or audio). A `ParallelPlayer` plays multiple `SequencePlayers`. By doing it this way it is possible to allow nesting of parallel tags. In some cases nested parallel tags is desirable since at every level of the multimedia presentation an author will able to decide which parts of the presentation have to play in parallel and which parts of the presentation have to play in sequence.
These players have corresponding models. So a `SequencePlayer` plays a `SequenceModel`, a `ParallelPlayer` plays a `ParallelModel` and a `MediaPlayer` plays a `MediaModel`. From `MediaModels`, `MediaTypes` are constructed.
The extensibility of XIMPEL and playing media types in parallel allows for one unexpected idea: one can extend XIMPEL with applications that are not multimedia. This question has been explored in another part of this thesis (see chapter 2).

3.1.1 Implementation of parallel playback

For the architecture of any player in XIMPEL (e.g. a parallel, sequence or media player) there is a pipe line. (1) It starts with parsing the XIMPEL playlist, which is in XML. (2) The data parsed by the XML syntax get put into an in-memory configuration object, which holds the details of the XIMPEL playlist. (3) The player itself uses, in most cases, a subset of the in-memory configuration object[1]. Each individual part of the pipe line is explained in its own section.

3.1.1.1 The parallel player and parsing

The first part of implementation that needed to be programmed was the parser. The parallel player needed to be registered as a valid child, since it otherwise would refuse to parse the `<parallel>` tag. In the `processSubjectNode` method the `processParallelNode` method is called which will return a parallel model. Such a parallel model is part of the in-memory configuration object. The method will determine the info of the `<parallel>` tag (e.g. its parent and children) which is needed to loop through its children. While looping through its children, the method will determine whether this particular

[1]Since the whole object is the complete playlist, so a part of the in-memory configuration object it is the relevant part of the playlist that needs to be played

`<parallel>` tag has: `<sequence>` tags, `<media>` tags or custom defined media type tags. It will then add those to the parallel model.

3.1.1.2 The in-memory configuration of the parallel player

The `ParallelPlayer` uses a `ParallelModel`. The `ParallelModel` itself is nothing more but a glorified array, as of now. It has a property called `list`. It also has a method called `add`, which means: add the children of the `<parallel>` tag to this list. These children are wrapped in a `SequenceModel` or `MediaModel` depending on what type the child is. It furthermore inherits the `get` and `set` method from `ximpel.Model`. The methods and variables in every method are completely identical to the `SequenceModel`. It only differs from the `SequenceModel` with the idea that the playing order of the `ParallelModel` is to play everything at once, instead of in a default or random sequential order.

3.1.1.3 The parallel player itself

The major code revisions were regarding the `SequencePlayer` and creating the `ParallelPlayer` itself. XIMPEL always has a top `SequencePlayer` (i.e. *the* `SequencePlayer`), it needs to be aware of how to start a `ParallelPlayer`. For example, when the `SequencePlayer` needs to stop it also needs to stop the `MediaPlayer` and the `ParallelPlayer`. Another example is that there needs to be a method in the `SequencePlayer` to play a `ParallelModel`. All the method really does is passing it on to the actual `ParallelPlayer`, but the `SequencePlayer` still needs to know about it.
The implementation of the `ParallelPlayer` itself is inspired by the `SequencePlayer`. Unlike the `SequencePlayer` which plays media types in sequence, the idea is to play everything all at once. The architecture of XIMPEL already allows for players to start playback for other players. In the architecture outlined in figure F.5, a `ParallelPlayer` will have multiple players as its children of some kind.
The `ParallelPlayer` holds a `players` array. These `players` can be either a `SequencePlayer` or a `MediaPlayer`. Since these `players` follow the same API, it is not always needed to distinguish what player is playing something. Just like the `SequencePlayer`, the `ParallelPlayer` has a `playbackController()` method. In the `SequencePlayer` this controller keeps track of which media item is played at this moment. The `playbackController()` method in the `ParallelPlayer` starts parallel playback of all `SequencePlayers` and `MediaPlayers` if it is not in a playing state. It furthermore attaches event handlers to all players to listen for when they stop playing their respective `SequenceModels` and `MediaModels`. This event handler is called `handlePlayersEnd()`. At every call the `playbackController()` checks how many media items already stopped playing, and does not play if, for some reason, all media items are in a stopped state.
For all the other functionalities the core idea of the `ParallelPlayer` is that all `SequencePlayers` and `MediaPlayers` need to be: played, stopped, paused or resumed. In other words: everything a normal `SequencePlayer` or `MediaPlayer` does, a `ParallelPlayer` does too but then for multiple `SequencePlayers` and for multiple `MediaPlayers`. The diagram below shows the new architecture of XIMPEL on a high level including the newly implemented `ParallelPlayer` (see figure F.5).
Applying variable modifiers (i.e. scores with an arithmetic operation such as `<score id="test" operation="add" value="5" />`) seems to work via their respective players. The only thing that needed to be added was that when a `MediaPlayer` would call its `play()` method, an `applyVariableModifiers()` method needed to be called in order to apply the scores that were in the `MediaModel` of the respective `MediaPlayer`. During manual testing regarding this feature, no bugs were found.

3.1.1.4 Testing and debugging the parallel player

Formal testing methods have not been used. The method that has been used is creating difficult playlists such as the playlist in appendix C. However, it seemed that this was not enough. What really helped to make the player more robust is by stretching XIMPEL to its limits. This has been done in exploration 7, an exploration outside of the scope of this thesis (see appendix A). In this exploration a Turing Machine has been created, as well as a simple shooter game and an incarnation of Flappy Bird. By stretching XIMPEL to its limits and trying to use it for purposes other than its intention (i.e. creating a Turing Machine), some critical bugs of the parallel player became very clear. These bugs have been resolved, and this chapter has been updated to reflect the implementation changes.

3.2 Conclusion

Parallel media playback works and other than the bugs written about, no issues have been observed as of yet. Other designs were possible but they have not been investigated. On another note, the creation of the parallel player allows for parallel playback, which allows the terminal media type of exploration 1 to be used in a much more useful manner since it is now possible to add video playback with it. Parallel playback allows for multiple media items of the same or different media types to complement each other through the ways such media items present their information.
However, such a conclusion is too short to be satisfying. Moreover, since this thesis is an explorative one it may be fruitful to know how far we have come. So in the remainder of the conclusion we are going to take a step back.
Since exploration 1, I started with the question of how XIMPEL and education relate and how it needs to be improved, I now stumbled upon a new question. What are the implications of parallel media playback? It is mostly the latter question that will be answered in the remainder of these pages. This is not because I wanted to answer it, it is because I stumbled upon many questions and issues going forward with parallel media playback.
Let us look in the future regardless. The `ParallelPlayer` has a lot of implications such as the user need for a media item surviving a subject switch (MISSS) when a user jumps to a new subject (see exploration 6) or the implications for time scrubbing are so numerous that it poses serious design issues (see exploration 5). The biggest implication of all is that everything is a lot less complicated when the developer only has one playing media item at a time to worry about instead of as many as a `ParallelPlayer` allows. Currently, XIMPEL is now able to play playlist C.1, which is in appendix C, which means it is able to play parallel media.
Let us look at the next exploration. Exploration 1 sparked exploration 3, porting XIMPEL to React. So for the next exploration, ideas about parallel media playback will be ported. However, this exploration will not be used as a basis. The programming philosophy in exploration 3 is too different. The implications regarding parallel media playback of this particular version of XIMPEL will come back in exploration 5 and 6.
So dear reader, I now give you a choice. Explorative reading is an art in itself as well.
Option A: You can either start reading exploration 5 and 6 if you want to know more about parallel media playback.
Option B: Or, you can start reading exploration 3 and read how I recreate the majority of XIMPEL with ReactJS and show how this particular library speeds up development.
The choice itself centers around the following question: do you want to thematically read all the themes in such a way that it is easier to digest? Choose option A. Do you want to read how the thesis happened in chronological order? Choose option B. Regarding option A, it is my suggestion to read the thesis in the order of: exploration 5, exploration 6, exploration 4 and exploration 3. Exploration 3 and 4 could be swapped since it is exploration 1, 2, 5 and 6 that are thematically linked. Regarding option B, I would like to remind the reader that the correct chronological order of how the explorations have been done are: exploration 3 (read only until the first subsection), exploration 4 to 6 and back to the rest of exploration 3.
Addendum. As an added bonus there is now a seventh exploration to be read, see appendix A.

4

Exploration 3: assessing the benefits for porting XIMPEL to React

This chapter was originally named as *Exploration 3 (part 3): assessing the benefits for porting XIMPEL to React*. This has confused some people reading my draft versions. Why would a chapter start with part 3? There is of course only one sensible explanation: part 1 and part 2 – also described as first attempt and second attempt – have been kidnapped. On a more serious note, the relevant elements of part 1 and part 2 are in this chapter as well, but it is part 3 that presents how XIMPEL could be effectively and efficiently be ported to React. In the first attempt, the port failed. In the second attempt, the port succeeded, but the architecture had too many issues. Moreover, it was observed that the architecture could be heavily improved by (interestingly) simplifying code, a lot of deep recursive calls have been removed. In an attempt to preserve how the exploration went from a chronological time line, the other two written parts are in the appendix.
With that said, ReactJS is hip, new and shiny. It introduces a relatively unknown paradigm to web developers called reactive programming. I wanted to work with this library because the web community seems to settle on this as a best practice for creating the view layer of web applications. Moreover, if my thesis supervisor can demand that I work on his framework pure for promotional reasons, then I can decide I learn a new framework for self-promotional reasons. Later on I also justified the use of React academically but I do not want to shy away from the inherent selfishness that academia has. If science is a form of truth discovery, then the truth of its process should not be hidden.
Related to this is that I wanted to create the same type of exploration that Stefan Bruins did. If Stefan Bruins can port XIMPEL to JavaScript and call it academic, then I can do the same for porting the JavaScript version of XIMPEL to React. I believe in both cases the research question of "is it possible?" could be answered a priori with a "yes." However, like Stefan I am an empiricist and the strongest form of evidence is through physical demonstration. Moreover, I altered the research focus to whether XIMPEL written in ReactJS (from now on called XIMPEL React) has more advantages than XIMPEL written in JavaScript + jQuery (from now on called XIMPEL JS). That answer is a lot tougher to answer a priori and hence an actual implementation, including a write up of the whole implementation experience is needed.
On another note, this started out as exploration 3, but part 3 of the exploration occurred later than exploration 4, 5 and 6 (addendum: not later than exploration 7, which was outside the project scope of this thesis, see appendix A). Therefore, part 3, the successful port including a decent architecture, could also been seen as the final exploration.
The reason this exploration started in the first place is because XIMPEL shares an idea that ReactJS has as well. They both have components. XIMPEL has it in the form of a playlist, where every component could be seen as a sort of invocation towards execution of the actual component. It is analogous to a function invocation and function definition. ReactJS is not bounded by a playlist, it has components everywhere; it also has the idea of component invocation and component definition,

like XIMPEL. Figure F.6 shows this thought. I was able to observe this, because I used ReactJS at the coding school I taught and had some understanding of how it worked, when one day looking at a XIMPEL playlist and React at the same time, it is hard not to make this observation. If I had not known ReactJS beforehand, I doubt that I would be capable of making such an observation.
My questions specifically regarding this exploration are: is it possible to make use React and its ecosystem to port XIMPEL to React efficiently and effectively? Moreover, what are other possible benefits or disadvantages regarding this port?
For porting XIMPEL to ReactJS I expected apriori that it had the following advantages:

- A lot of parsing logic could be done via ReactJS and Webpack[1] by transforming the XIMPEL playlist to a declarative language that is completely compliant with JSX[2]. So there is no need to create a parser.
- The virtual DOM would replace the in-memory configuration code that has been written for XIMPEL. So there is no need to write in-memory configuration code.
- Cross-browser support is suddenly managed by the maintainers of the ReactJS framework.
- By teaching XIMPEL to students, it is needed to teach about ReactJS to students who want to extend XIMPEL. This introduces them to some of computer science concepts implicitly.

In short, the idea was to see if it is possible to make the ReactJS framework work for us. If this is possible, then we as XIMPEL developers have a free lunch! Who does not want a free lunch? Bagels are on me!
To validate these assumptions, I made a very simple prototype of creating a custom React component in an XML file. The XML file would be read in by a React class that I programmed. What I found is that this exploration failed so dramatically that for more than 6 months I believed to have invalidated most advantages. You can read a full write up of the failure in appendix D.
However, by writing the chapter 6 months later, I was forced to take a closer look at the source code. Because I needed to look at the source code, I needed to look at the dependencies. And when I looked at the dependencies, I noticed that the XML parser of Webpack uses a library in order to parse the XML. It may seem silly, but at the time I believed Webpack had innate code for doing this task. Knowing this was a library, there seemed to be a small detail that I missed. Upon further inspection, this dependency of the XML parser of Webpack showed that the XML parser was configurable! Unfortunately, I did not see this before. So I could do a lot more in my second attempt, which is described in the next section.

[1]The following quote is from `https://github.com/webpack/webpack`: “Webpack is a module bundler. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of transforming, bundling, or packaging just about any resource or asset.”

[2]HTML-like syntax that describes a user interface and is transformed to pure JavaScript, see `https://reactjs.org/docs/introducing-jsx.html`

Research Questions and Contribution

All research questions (1, 2 and 3) are relevant for this exploration.
The first research question is: (1) how does XIMPEL need to be extended to contribute to online education? This exploration contributes in that regard by being as educationally useful as the default implementation of XIMPEL. The extra way in which the React implementation is useful is that it could prompt students to learn about React and code reusability – since code reusability is a big design philosophy in React. Furthermore, by having 2 implementations, students will more likely understand the role of XML, since it is easier to see that XML first needs to be interpreted. Moreover, the biggest two contributions are that hypermedia may now be easily ported to mobile and the longevity of the XIMPEL project may have increased because of creating a hedge between plain JavaScript and having an implementation in ReactJS. The codebase of XIMPEL is small enough to keep it maintainable. Finally, by showing that XIMPEL can be ported in a quick fashion means that it is able to have an up to date implementation for the years to come.
Research question 2: (2) what are the (technical and design) implications of parallel media playback? Since XIMPEL React has its own form of parallel media playback, it contributes in the same way as exploration 2 does. This means that: the implementation issues of creating parallel media playback sets up the stage in order to be able to conceive an answer regarding the technical or design implications regarding parallel media playback. It has to be noted that creating parallel media playback in XIMPEL React has been much more straightforward.
Research question 3: (3) what areas of research could XIMPEL benefit from, and how? This exploration allows the reader to compare two implementations against each other. In this sense XIMPEL JS acts as a control and XIMPEL React as an experiment. Furthermore, this comparison is partially done in exploration 3 itself. The discipline that this particular exploration benefits from is the academic discipline of software engineering.

4.1 Webpack XML parser setup

There were two modifications that allowed for a much more successful exploration compared to the first attempt. First, I modified the parser to have an explicit tree structure by preserving order between siblings and parent-child relationships. In general, it is the question whether an XML document needs this type of order preserved since some XML documents could be treated as an unordered set.
For XIMPEL, it is needed that the XML document would be parsed as an ordered array. This feature gives more power to the programmer, in the same sense that a Turing complete system has more computational power than a finite state machine. Without order there is chaos, and in this particular case there would sometimes be no way to determine which media item (such as a video or image) or subject should be loaded first.
The second modification helped a lot in code readability. While it is not as game changing or groundbreaking as the first, code readability is a necessary requirement for a fruitful collaboration on any software project. For example, the default setting to denote attributes of a parsed XML tag itself was `$`. To denote inner text it was `_`. And children was `$$`. I renamed this to: `attributes`, `text` and `children` respectively.
The parser in XIMPEL JS is 812 lines of code. This particular setup is 15 lines of code and works just as well. For this particular part of the port, ReactJS including its ecosystem (e.g. Webpack) definitely wins. While a similar library could have been used for XIMPEL JS, the in-memory configuration object still would needed to be written. In ReactJS this is not the case since React props handle any passed XML data. The in-memory configuration object is 316 lines of code in XIMPEL JS, in XIMPEL React, the React library completely handles that. Table F.1 shows the comparison. This particular success has been made during the second attempt of porting XIMPEL JS to XIMPEL React.

4.2 Implementation methodology of the second and third attempt

In the second attempt of this exploration I tried to explore if I could re-implement XIMPEL quickly since the possibility of it is one of the main questions. I did not consider architecture, or even best practices for programming React. I also did not consider the architecture or implementation of XIMPEL JS, in order to really test if ReactJS would help. This has led to quite a bit of technical debt. Creating technical debt has also been done in order to find out what architectural patterns work and which does not. Iterative development, therfore, has been a conscious choice. Another reason to create technical debt is because I follow the programming philosophy of Jonathan Blow, which is: try to produce as fast as possible and then when you hit performance bottlenecks or any other bottleneck of any kind, only then try to be smart about solving that bottleneck. In some of his YouTube videos he goes in-depth how the code that he created for his award winning games like Braid and The Witness only have 6 to 10 percent of optimized code [3]. It has also been done in order to find in which regards one needs to fight the React framework and in which cases React gives the developer wings to fly and develop certain features of XIMPEL a lot faster.
This approach worked. However, while it worked it did create an issue too big to ignore when it came to infinitely nesting the `media` tag (which replaces the `<parallel>` tag, making parallel playback the default option) and the `<sequence>` tag (which allows for sequential play)[4]. Since infinitely nesting parallel and sequential play is a feature essential to any hypermedia framework, I decided to revise the architecture on the best practices of ReactJS. This also meant that I almost completely restarted from scratch, except for the Webpack XML parser setup, which is exactly the same. The write up of my second attempt is also put in the appendix which can be read in appendix E. This write up is not complete. The incompleteness shows the sudden transition to the third attempt. Besides the necessity of revising the architecture and learning more about the best development principles regarding React, I still had the same programming philosophy in mind inspired by Jonathan Blow.
To summarize, there have been three attempts in porting XIMPEL to ReactJS. The first attempt did not go anywhere (see appendix D), the second attempt did but had an architectural flaw (see appendix E) and the third attempt has an architecture that is clear and one that works. The third attempt will be exclusively presented in the next sections (without the first and second attempt). The final two attempts do have a couple of things in common (e.g. overlays, Webpack setup and data flow philosophy). So some paragraphs in the appendix are the same as they appear in the chapter here.

4.3 Designing XIMPEL React

While porting XIMPEL JS to XIMPEL React, I decided to leave some features out. I did this in the interest of time and in the interest of vision. The requirements of a fixed width and height 1920 by 1080 has been dropped. This is because there is more focus on displaying (hyper)media than XIMPEL being an interactive video player. The `<quiz>` tag has been left out since it can be modelled already through other language constructs that XIMPEL provides. More importantly, prominent XIMPEL authors do not use the tag and they model quizzes with overlays[5]. One way to model it can be seen in figure F.16. The playback bar on the bottom which ahs the pause, stop and play functionality has been dropped, because there has been some indication by students and the observations of Hugo Huurdeman that these features make users assume that time scrubbing is an available feature, leading to UI-frustration. Moreover, they are not used much, if at all. The error message system has been altered as well. Tags that are not allowed are shown as an error message on the page, because not every XIMPEL author knows about the JavaScript console. Unknown attributes are not shown as

[3]I forgot which video, but here is his channel: `https://www.youtube.com/channel/UCCuoqzrsHlwv1YyPKLuMDUQ`.

[4]The rationale of this design change is explained later on.

[5]When looking at all playlists for all showcases on `https://www.ximpel.net/showcase`, it is visible that all quizzes are implemented through overlays.

an error message, it would be more productive to do so but experienced authors can find these bugs quite quickly and inexperienced authors learn a thing or two about debugging. This is an educational experience that every XIMPEL author should be familiar with. In this sense the XML language of XIMPEL React tries to be pedagogical, just like XIMPEL JS but to a slightly further degree.
Every exploration that I wrote contributed at least a small feature to XIMPEL React, most of which are also in XIMPEL JS. The ability to play parallel media (exploration 2), integrate a terminal (exploration 1), log user data and capture their facial expressions (exploration 4), media items surviving a subject switch (MISSS, see exploration 6) and even simply leaving the time scrub bars in for audio and video (inspired by exploration 5, this feature is not in XIMPEL JS).
The process of the design happened intuitively. The requirements of porting are quite clear and so is the problem definition regarding re-implementing XIMPEL. Normally, the first major step is devizing the architecture and I decided to program it iteratively. By programming an architecture iteratively, it is possible to see what works and what does not work in earlier versions of XIMPEL React. Furthermore, a lot of the design process also occurred by virtue of writing this thesis and revising this chapter more than any other chapter.

4.4 Architecture

In this section it is described how the third attempt improved upon the second attempt from an architectural standpoint, since improving and implementing the architecture is the biggest difference between the two attempts. Then, a small explanation of some terminology known in compiler construction will be added to our vocabulary in order to explain this architecture better. Then, the rules of all components are described. While it is not a comprehensive rule list it does give a good idea of what XIMPEL React is capable of on a per component basis. Describing rules on a per component basis does not give the full picture, which is done in the subsequent sections.

4.4.1 Improvements compared to second attempt

The general idea of the architecture is that each XML tag has its own component. Every component is responsible for rendering its own level, which is an improvement compared to the second attempt where there was one top level component trying to render everything. For example, a media type like the `<image>` tag would render its own image via the HTML5 `<img ... >` tag. Intangible tags such as the `<sequence>` tag would render as a `<div class="sequence">` in HTML5. Another improvement is regarding the in-memory configuration which is an object that is created by Webpack which parsed the XML file. In the second attempt, the in-memory configuration consisted of actual React elements. In the third attempt the program uses the parsed XML by Webpack as the in-memory configuration. This eliminated a whole host of introspection issues since React elements (2nd attempt) are tougher to inspect compared to an XML in-memory configuration (3rd attempt). One potential tried solution was to stringify React elements to JSON and do a string match. This did not work, because a React element has circular dependencies, which means the stringification to JSON will never complete. Contrast that to parsed XML which had a `#name` property (the hashtag is not a typo it is part of the identifier) which indicated the type of the parsed XIMPEL tag (e.g. `audio`[6]). A third improvement is regarding the communication, which has been streamlined. Specifically, there are fewer `publish(...)` and `subscribe(...)` calls.

[6]Indeed, it is with a lowercase `a`, React elements and components would be with an uppercase `A` but in XML form it is a lowercase `a`

4.4.2 XIMPEL tags mapped to React component and the link to compiler construction

To start off this section: consider the following excerpt of the De Zaanse Schans playlist (see code section 4.3).

```xml
<ximpel>
    <playlist>
        <subject id="MenuDeBonteHen">
            <media>
                <video repeat="true">
                    <source extensions="mp4" file="videos/MenuDeBonteHen"/>
                    <overlay height="100px" leadsTo="WalkToHetJongeSchaap" width="350
    px" x="200px" y="970px"/>
                    <overlay height="100px" leadsTo="QuizBonteHenIntro" width="350px"
    x="800px" y="970px"/>
                    <overlay height="100px" leadsTo="TourOfMolenDeBonteHen" width="350
    px" x="1470px" y="970px"/>
                </video>
            </media>
        </subject>
    </playlist>
</ximpel>
```

Playlist 4.1: An excerpt of the De Zaanse Schans playlist.

Every tag here as its own React component. The `<ximpel>` tag has the `Ximpel` component, the `<playlist>` tag has the `Playlist` component and so on.
This design makes it possible to devise rules for every component, just like it is possible to devise rules for every context free grammar (CFG) when parsing a programming language. Since a XIMPEL playlist is a declarative language, borrowing conceptual ideas from syntax analysis in compiler construction may prove to be useful for describing XIMPEL's architecture.
A CFG has four elements to it. A set of non-terminals, a set of terminals, a set of production rules and a starting symbol. Of relevance here are the ideas of: non-terminals, terminals and production rules. Knowing which are which tells more about the architecture. These terms will not be used in a strict sense, but the essence of these ideas will remain. When I clearly deviate from the strictness of one of these ideas it will be written in advance. The idea of production rules will be renamed to component rules since the rules of what every XML tag should do is encoded in the React component that maps to it and strictly speaking they are not production rules (though they have some similarities).
The terminals are the easiest to understand: the media items displayed on the web page are the terminals, such as a video element or audio element. These terminal symbols are not strictly terminals in the CFG sense since they can nest elements as overlays and supporting elements (e.g. `<source>` for the `<video>` and `<audio>` tag). They are terminating symbols in the sense that the endless nesting stops and a media item will be displayed. Other terminating symbols are: `<source>` (as a part of `<video>`) and `<overlay>` (as a part of any media type).
The non-terminating symbols are: `<ximpel>`, `<playlist>`, `<subject>`, `<media>` and `<sequence>`. Writing a playlist with only these tags will yield in a meaningless playlist. The idea of non-terminating symbols are that they eventually lead to terminating symbols and show the intention of the playlist.

4.4.3 Component Rules

The component rules are written per React component in the source code. For architectural purposes, I will present the rules per component, but they may not be fully comprehensive. They will, however, serve for having a strong understanding of the architecture underlying the code base. The rules will be discussed from top to bottom. It is also possible to read from the documentation what the component rules are[7].

[7]See `http://melvinroest.com/ximpel/documentation/ximpel_react.htm`

4.4.3.1 Ximpel

The `Ximpel` component checks whether the `<ximpel>` tag has been written.
It sees if there is a `<playlist>` tag underneath it and will output a warning on the page and not render anything else.
It does not check if there is one `<ximpel>` tag.

4.4.3.2 Playlist

This component tracks which subject is being played.
It also tracks all the global media items (media items that survive a subject switch, i.e. a global media type) and renders them if needed.
It initializes the logging framework the `enableLogging` attribute is set to `"true"`.
It also has two subscribed methods in the publish-subscribe system that XIMPEL React uses.
It listens to any component publishing a topic on `leadsToUpdate` and it will update its own state to render a new subject.
It also listens to when a `Media` or `Sequence` component signifies the topic of `addGlobalMediaItem`, in which case it adds a media item to its own `globalMediaItems` array so the media item is able to survive the subject switch.
It renders the global media items and the underlying `<subject>` tag. If no `<subject>` tag has been specified it will display an error message on the page.

4.4.3.3 Subject

This component looks for the underlying `<media>` or `<sequence>` tag. If it is not there it will display an error message on the page.

4.4.3.4 Media

This component plays media items in parallel.
It is able to render a `<sequence>` tag or any media tag.
It will give an error message if no media type tags or `<sequence>` tags are children.
It counts how many of its children have stopped playing and notifies this to the parent. This notification is intended for the `<sequence>` tag. With it, the `<sequence>` tag knows that it is able to continue playing the next media type or `Media` component. If the parent is a `<subject>` then this notification is not relevant.
Before playing it checks to see if its first media item is a global media item. If it is, it will increment the stop counter since the responsibility of a global media item does not belong to the `Media` component but to the `Playlist` component.
When it renders a media tag, it also renders an element that tracks time for the media item as a parent. As of now this is called `MediaType` but it may also be called `MediaManager` later on. Here is a piece of example code.

```
<MediaType stopCounter={this.stopCounter} {...element.attributes} playlist={
element} key={this.state.key + i} render={mediatype => (
    <Youtube {...element.attributes} mediatype={mediatype} text={element.text}
playlist={element} />
)}/>;
```

Playlist 4.2: An example of how each media item has a general component to manage itself for time tracking (among other things).

4.4.3.5 Sequence

This component plays a media item (or `<media>` tag) one at a time and will go to the next one when the duration of its current media item is finished. If a media item has no duration specified in the playlist, this media item will be playing it for an infinite amount of time. A `<media>` tag does not need a duration, it only needs to notify the `Sequence` component that it is done rendering all its children. The `<video>` and `<audio>` tags are also an exception for this duration rule since they have a custom `handleEnd()` method[8].
It will give an error message if no media type tags or `<media>` tags are children.
It counts which child it is playing and whether it is finished playing.
When it is finished playing it will notify its parent that it stopped. This notification is intended for the `<media>` tag so it knows that this component is done with rendering everything. If the parent is a `<subject>` then this notification is not relevant.
When it renders a media tag, it also renders an element that tracks time for the media item as a parent. As of now this is called `MediaType` but it may also be called `MediaManager` later on.

4.4.3.6 MediaType

The `MediaType` component is the only component that does not have a direct one to one mapping with the XIMPEL playlist. This component demonstrates how the abstraction of a one to one mapping from XML tags to React does not hold. The condition of grouping functionality of XML tags together breaks the abstraction. This component applies to all media types and acts a manager for every media type. All media types have a couple of requirements in common which is why this component was created.
It tracks: duration, the amount of seconds elapsed, whether it has to render itself, it also tracks an inner state of whether it should play and the underlying media type and when a media item needs to start playing.
It is able to get information from the underlying media type regarding the amount of seconds elapsed. This feature is also seen in XIMPEL JS, where it has been argued in the comments of the source code that the video and YouTube API are better able to track time than the JavaScript implementation of XIMPEL JS itself.
It notifies the `Media` or `Sequence` component when it stopped playing.
It renders overlays that are children of any media type tags. In the general architecture of XIMPEL React it is a rule that an overlay can only be rendered by a media type.

4.4.3.7 Media Types

Most media types simply render themselves, take in the attributes of what was written in the playlist. `Video`, `Terminal` and (in future work) `YouTube` have more than only a call to a render method.
All media types are able to render overlays and overlays are only allowed to be rendered as children of media types.
The `Video` component also tells `MediaType` when it is done playing itself when it is not on repeat and it gives a reference of itself to `MediaType` in order for `MediaType` to improve time tracking (e.g. when a video is paused there is no time elapsing). The `Youtube` component could be programmed to do these things too, but as of now this has not been done yet.
The `Terminal` component connects to a server that runs a bash shell and communicates via web sockets. Because of this, it is able to handle form submission.

[8]In retrospect, I did not design this part of the architecture well. It is also evidence that I did not look at the architecture of XIMPEL JS. Perhaps a similar approach could be taken that XIMPEL JS does. XIMPEL JS attaches methods that handle ending events on the moment when a media item, or player starts playback.

4.4.3.8 Overlay

This component has a static variable called `score` which is an object that tracks all the scores that are put as an attribute in the `<overlay>` tag.
It tracks its own time, except when `MediaType` passes down a reference of the HTML5 video player. Then the time will be tracked for the overlay.
Other than tracking time it also tracks start time and duration.
It renders itself and has no children.
The most important feature of an overlay is that when it is clicked, it will publish a `leadsToUpdate` to any subscriber willing to listen which is the `Playlist` component, since it will start rendering a new `Subject` component, forcing a rerender of a whole new part of the playlist.

4.4.3.9 Rule

The rule component is the replacement of the `<leadsTo>` tag. It checks if a score is big enough in order to do conditional rendering.
Rules are only possible to be nested in overlays.
The parsing of the `if` attribute has been directly copied and slightly modified from XIMPEL JS due to its very specific use-case and specific syntax.
It is renamed to rule because creating a `<leadsTo>` tag is confusing since it is already available as an attribute. Moreover, renaming it to `<rule>` showcases a more general intend that is made more specific with the attributes `if` and `leadsTo`. Here is an example:

```
<overlay>
    <rule if="{{baby_dolphins}} > 5" leadsto="lesson2"></rule>
</overlay>
```

Playlist 4.3: An example of how the rule tag looks like

4.4.4 Beyond the component rules

It is important to know that while these rules describe the components fairly adequately, two things have been left unsaid. Certain edge case behavior has not been discussed and lifecycle management issues that needed to be programmed against. An example of one edge case is that the logging framework is written in ES5 and jQuery. Because of this the `Playlist` component needs to check if jQuery exist and use it in order to attach the facial expression classifier part of the framework to the DOM. Another example is key management, which is needed for many React components. If React needs to use deep diffing into the DOM (one of the advantages of using React, performance wise), then the lifecycle methods will be called at unfortunate moments. This is an issue if a new developer does not understand key management and how it influences the lifecycle methods. Currently it is not a problem anymore, but it used to big enough to mention it as a potential temporary drawback.

4.4.5 Data flow within XIMPEL React

The data flow within XIMPEL React tries to follow conventional ReactJS philosophy: try to have a unidirectional data flow as much as possible. Which means data flowing from parent to child. However, sometimes this is not possible. If a child gets a state change earlier which a parent also would need to know, then it in some cases becomes difficult to do so. Conventional React best practice suggests to lift state. However, this has not always seemed to be possible, but furthermore it makes code readability worse.
For child to parent communication I used callbacks or render props (through which I could use callbacks). For great great great ... great grandchild communication to the `Playlist` component I used a publish subscribe library called PubSub.js [82]. Some people reading this may ask themselves why I did not use the Redux library. The answer simply is: it was not needed. I know that a lot of developers use

the flux architecture for solving data flow communication issues, but this is only needed when current data flow communication solutions become unwieldy. This is not the case for XIMPEL React. It may be the case in the future, when the codebase passes 5000 lines or 25000 lines (who can tell?) and then refactoring will be needed. I also did not use the PubSub library of XIMPEL JS, simply because I prefer to concentrate open source efforts and the popularity of PubSub.js [82] is clear.

4.5 Conclusion

So who is hungry? It is about time to assess whether porting XIMPEL in React provides a free lunch! The assessment works as follows. First, unexpected advantages and disadvantages will be taken into consideration. Then, the expected advantages will be evaluated by asking: to what extent were the expected advantages positive development factors in reality?

4.5.1 Unexpected advantages and disadvantages

The biggest downside of React was not the library itself but the learning curve of it. The fine-grained understanding over the lifecycle and key management has produced hundreds of lines of code that were ultimitaly unecessary and have been refactored out. Without this fine-grained understanding, more control over the DOM would be nice, as well as knowing when the lifecycle methods would be triggered. Fortunately, it was a learning curve issue and nothing else. For developers new to React this will be a temporary drawback.
The pro's of the React and Webpack combination were plentiful. One advantage mentioned earlier is that React components map really well on XML elements of the XIMPEL playlist. Since the component abstraction rarely breaks, it is fairly easy to reason what each XML element is responsible for regarding which React component.
However, it did break one time. The `MediaType` component is not an XML tag. This indicates that in XIMPEL a media item belongs implictly to a media type. It is the question whether this should be implicit or not. By making it explicit by creating a `mediatype` tag, for example, the abstraction with React does not break at all. This could be evidence for how the architecture plus the React framework really forces us to stay consistent with mapping XML tags one to one with React components.
Another advantage that came to light later on was the lack of DOM manipulation needed. React does this. Normally, a developer needs to be concerned about: state, rendering HTML (most likely with jQuery) and when to render the HTML. With ReactJS the only concerns are having the right state and rendering it, which simplifies reasoning about the whole application since the question of when to render HTML is gone.
One future advantage is when XIMPEL or concepts like XIMPEL will be used for mobile applications. React Native shares a lot of similar code with ReactJS and to port it over to mobile should be possible. One interesting caveat is that mobile applications need to be reviewed by their respective app stores. However, a playlist that is sent over the server does not need to be reviewed. So a technique to extend an app is to create one's own custom declarative language and load it in as a playlist, like XIMPEL does. A related advantage is that it is almost within reach to create mobile apps with XIMPEL. The framework has to be adapted to React Native, which has a lot of code in common with ReactJS, in general.

4.5.2 Evaluating the expected advantages

The first pro is that a lot of parsing logic could be done with React and Webpack. This is true, but interestingly enough, not by transforming the XIMPEL playlist to a language that is completely compliant with JSX. Therefore, the expected benefit was right but the expected mechanism by which it would be achieved was not. The XML loader of webpack had (in hindsight) a strong XML parser that was a good library to use [54]. This parser created a workable in-memory configuration object.

Which brings us to the second pro. The virtual DOM would replace the in-memory configuration code. So there is no need to write in-memory configuration code. While it has been tried to use the virtual DOM to replace the in-memory configuration code, this has been tedious – this has been done in the second attempt. As stated in the previous paragraph, the actual parsed XML tags were used, which provided the in-memory configuration that was needed – this has been done in the third attempt. Again, the expected benefit has been realized but the expected mechanism by which it would be achieved was not.
The third pro is that cross-browser support is managed by the maintainers of the ReactJS framework. This is unfortunately not entirely true since the logging framework has not been ported to React. ReactJS itself support Internet Explorer 9 and higher through the use of some polyfills. It used to support Internet Explorer 8, but it has discontinued support since the beginning of 2016 [1]. What has not been taken into this consideration is add-ons such as the logging framework being written in an older language.
The final advantage was a didactic one. Students who want to extend XIMPEL React need to learn a thing or two about ReactJS. This in turn leads them to learn some computer science concepts. It is hard to assess this potential advantage, but since I needed to brush up on my ReactJS skills, I can review which programming concepts I have gained a better understanding of. These are: lifecycle methods, diffing, performance between a virtual DOM and actual DOM, the DOM itself, some basics of functional programming (I did not follow the course in my bachelor degree), the render and state cycle, the lifecycle of a component (it is reminiscent of the iOS framework Cocoa and Cocoa Touch which also has lifecycle methods) and key management (unique identifiers). It could be argued that some of these concepts are also computer science concepts. Furthermore, students who want to extend or play with the core of XIMPEL will learn a lot about web development. So with some certainty it can be claimed that students who dive into the React version of XIMPEL will learn some additional computer science concepts compared to XIMPEL JS.

4.5.3 Concluding the evaluation

In summarized fashion, the advantages of ReactJS are: no DOM manipulation, XML attributes easily included as props (thanks to a combination of React and the XML parser), strong one to one mapping to the XIMPEL playlist, cross-browser support, strong error reporting, a strong ecosystem, and porting options to mobile and tablet. The strong one to one mapping to the XIMPEL playlist has been very useful for development. Focusing on solely one tag and writing out the component rules for that tag is helpful. Regarding the props of ReactJS, it has not been needed to attach the XML attributes to anything, unlike in XIMPEL JS. The ecosystem of React is perhaps its strongest advantage since it offered easy discoverability for libraries that saved a lot of time, such as the XML parsing library or the ability to write ES6 with Babel. This means: less code to write, an architecture that is fairly easy to reason about, maintainability out of the box and rapid software development options on other platforms. This is a lot. A speculative advantage is that ReactJS might segment its position as a best practice for the web.
The only downside is that people need to learn it and become proficient in it. Fortunately, the learning curve should not be that high since XIMPEL React only uses ReactJS its core library and DOM library. It does not use other popular libraries from the React ecosystem that invite a learning overhead (e.g. Redux). Hence, the downside is relatively contained compared to other React applications.
Does this mean that we should abandon ship and stop the development of XIMPEL JS? Compared to the development time of XIMPEL React, the creation of XIMPEL JS has also been realized relatively quickly. While I do not have data on the matter, my guess would be that it has been made within 336 to 672 hours of development time. Bruins developed XIMPEL JS for his master thesis, and had (officially) 1004 hours time for the whole thesis. The point is: that is still relatively quick.
As of now, XIMPEL React is a subset of XIMPEL JS and has a couple of different design decisions. Therefore, no version of the framework replaces the other. The design decisions of XIMPEL React easily shows whether the design decisions taken were a good idea. XIMPEL React has no: pause and

resume functionality on a per subject basis, quiz tags, constant dimensions (i.e. 1920 x 1080). Some media items have rudimentary video and audio time scrubbing. Preliminary results show that: quizzes can still be modeled; even quizzes with feedback and not having constant dimensions has consequences of the positioning system in such a way that it is identical to positioning web elements. This makes positioning a bit problematic, a proposed solution is in the future work section.

4.5.3.1 Architectural similarities between XIMPEL JS and XIMPEL React

One strong reason for making a strong decision for XIMPEL React is an architectural one. After creating XIMPEL React, I observed that XIMPEL JS has an architecture that is quite close to the architecture of XIMPEL React[9]. Interestingly, the idea of mapping one tag to one file or one object type has not been explicitly mentioned in Bruins his thesis [8] because he framed the architecture in a different manner. Nevertheless, that is more or less what he did.
In XIMPEL JS there is much more concern with a player, a sequence player, parallel player and media player. These players do not have any UI logic, the counterparts in XIMPEL render a simple div. It was surprising at all that XIMPEL React has counterparts! Furthermore, for most components in XIMPEL React a similar JavaScript file has been found in XIMPEL JS.
In XIMPEL JS, every media type has its own file. In XIMPEL React, every media type has its own component. The rendering logic in React is cleaner, because React is meant for such a task. XIMPEL JS uses jQuery which works, but has worse code readability because the HTML-strings cannot be syntax highlighted. In jQuery HTML-strings are simply seen as strings in JavaScript and not as JSX like in ReactJS. Furthermore, they both have some form of code that manages shared responsibilities of all media types. In XIMPEL JS this responsibility is shared between `MediaType.js` and `MediaPlayer.js`, in XIMPEL React is called the `MediaType` component.
What does not exist in XIMPEL React is `XimpelApp.js`, mostly because it is not needed. There is also no code registering the XIMPEL name space or media type registration. It might be needed, it might not be. The `XimpelAppView.js` file has no counterpart, this file draws the video player-like UI. In XIMPEL React it has been decided that such a UI created false expectations. It does not have a `View.js` file because this file abstracts common view functionalities away for overlays and questions. Since XIMPEL React has no questions, there is no need to abstract it away.
`OverlayView.js` is the `Overlay` component. And that is the whole comparison: players, media types and overlays. Tougher to compare are: the `Rule` component and the scoring part in the `Overlay` component. The scoring part in `Overlay` could have a better architecture in XIMPEL React and has been added relatively quickly. The `Rule` component seems to have an equivalent in `Player.js`[10].
I have shown that there exists a rough one to one mapping between many React components and many XIMPEL JS files. If there is not a one to one mapping, then there is a one to one mapping between the concepts ReactJS uses itself or it simply was out the design scope of XIMPEL React. Therefore, using ReactJS might be preferable, provided that the design changes are accepted by the development team. A table which compares all filenames of XIMPEL JS to XIMPEL React components, concepts and design decisions has been made (see table F.2). This table could also be consulted to compare in what manner the functionality relates to each other in each codebase.

4.5.3.2 Features that still need to be implemented for feature parity

XIMPEL React has a couple of features that XIMPEL JS does not have, such as scrub bars on video and audio media items. It also has a property that allows a mediatype to determine when it should be played. In XIMPEL JS this issue is solved by using a media type called `Filler.js` and play that for a certain amount of time to offset the starting time of when a media item could be played. XIMPEL React does not have strong event handling for ending events which XIMPEL JS does have.

[9]Which is why this part is written in the conclusion, it strictly has been observed after the creation of XIMPEL React. If it had not, then part 1 and part 2 of this chapter, see the appendices, would not have existed.

[10]It may seem that `Model.js` also seems to have a part of the `Rule` component. However, the functionality of `Models.js` is simply taken over by React props.

The YouTube mediatype in XIMPEL JS has a mute attribute, which XIMPEL React does not have. Most importantly XIMPEL React does not support touch gestures, which XIMPEL JS does.
Another functionality that XIMPEL React does not have is conditional subject rendering when a user arrives at a subject. It only has conditional subject rendering when a user clicks on an overlay. The same goes for scores setting scores or performing arithmetic operations on them, it only works when a user clicks on an overlay. This design decision has been intentional in order to keep the language minimal. However, in the extracurricular exploration (exploration 7, see appendix A), it has been demonstrated that these language features are needed.
All other features have either been intentionally designed away or built. It seems that XIMPEL React is not a pure subset of XIMPEL JS because of its own unique features and seems to have nearly all features regarding its own design goals. See section 4.3 to read how XIMPEL React differs in its design from XIMPEL JS.

4.5.3.3 In closing

This exploration is a final exploration in disguise. Since it had three attempts, only the first attempt was the actual third exploration which is now in appendix D. The second and third attempt can be seen as the seventh exploration respectively. When I ported XIMPEL to React successfully in my second attempt and fine-tuned it in my third attempt, the first rough draft of everything (including this chapter) was already written and everything else was already programmed. So other than an assessment to see whether XIMPEL could be ported to React, this port also showcases a particular vision of XIMPEL. The other explorations serve as puzzle pieces of programming and design explorations and they serve as an inspiration for this implementation of XIMPEL React.
Addendum. While it is true that this is the final exploration, I am constantly think about XIMPEL. If I find something interesting, then I decide to explore the topic in a shallow manner or in-depth. Shallow explorations do not make this thesis, in-depth explorations do. Because of this, another exploration has been made – exploration 7 (see appendix A). Requirements creep and project scope expansion is something that has happened in this master project quite frequently. Unfortunately, due to the project scope, this epxloration will only be promoted in addendums or very last minute edits such as this one.

4.5.4 Future work

Since XIMPEL React is more web-based in terms of layout compared to XIMPEL JS, it would benefit from having a different positional system. To this end, it may be an idea to implement CSS for XIMPEL React. CSS already gets rendered through the browsers so the language does not need to be defined. Fortunately, the architecture serendipitously allows for CSS already since every React component renders at least a div with a distinguishable class. Adding XIMPEL author created identifiers or class names as attributes via XML tags would be helpful. Another task that needs to be accomplished is documenting per tag in XIMPEL which CSS elements are already used for the old positioning system. By extension it needs to be written about how CSS differs because of this.
Other future development work would be to get feature parity with XIMPEL JS. Currently XIMPEL React is clearly in a prototypical stage and, according to its own design goals, it is quite far with regards to feature parity, but it is not there yet.

5
Exploration 4: creating the necessary requirements to measure the frustration of users in XIMPEL

At one point while I was playing with XIMPEL I noticed that I found it to be a shame that every playthrough would be forgotten after it had ended. I wanted some form of memory. I wanted some form of logging. I then realized that creating a logging framework and creating a theoretical framework on how to measure frustration with it would be valuable for improving the user experience of XIMPEL. So in this exploration I am finding out what kind of theoretical framework for frustration (and to a lesser extent engagement) we should have and from that framework what the minimal requirements are regarding data capture.
One reason I chose frustration as an important metric for user experience is because it seems that the experience of frustration is still quite misunderstood in academia. Having written earlier about frustration from a psycho-neurobiological perspective, I had the idea that I would be most productive for the academic community if I would resume my work in this area of research. Another reason is: this is my first attempt and experience at answering an artificial intelligence question while having a background in it through other disciplines, which is exciting!

Research Questions and Contribution

How could a semi-automatic system regarding a positive or negative user experience be implemented? This question has proved to be too much to answer within one thesis that is structured in an exploratory manner. The actual question that has been answered are: what are the minimal requirements needed in order to implement such a system? The answer to that is: capturing data and understanding how to classify frustration. This data capturing facility for XIMPEL has been implemented. Ideas regarding how to classify frustration have been put forward and they are backed by literature.
The answers to these specific questions for this exploration are contributions towards research question 1 and 3. Research question 1 is: (1) how does XIMPEL need to be extended to contribute to online education? Research question 3 is: (3) what areas of research could XIMPEL benefit from, and how?
It contributes to research question 1 since frustration and engagement are important indicators to keep progress of. Students seem to experience them quite a bit. Unsurprisingly, some of the academic literature is precisely the intersection between programming education and frustration. Having a core understanding whether frustration is useful or not useful, when such a thing is the case, how to semi-automatically detect a useful frustration or a not so useful frustration will help students following introductory programming courses.
It contributes to research question 3, because the two fields that are outside the area of hypermedia are a blend of Artificial Intelligence and Psychology.

5.1 Justification for creating a logging framework

One justification is: to understand your users. Hugo Huurdeman shows this justification in work that happened at more or less the same time [70]. In order to understand more about the user experience (UX) that XIMPEL users go through I have created the beginning of a framework that logs data needed for classifying frustration or engagement. Understanding whether a user is engaged or frustrated helps to improve the UX for authors creating XIMPEL presentations. I chose for frustration and engagement because it will give insight into whether a user wants to quit in the middle of a XIMPEL presentation (frustration) or is driven to see all of it (engagement). Moreover, it begs the question: how does one measure and classify frustration and engagement in hypermedia frameworks? For this question I emphasize a focus on frustration, because quitting an application out of frustration is a more urgent problem to solve than to keep users fully engaged. Another question that came up is: does frustration always lead to quitting behavior? The answer to that is no. It does not always lead to quitting [89]. A follow-up question would then be: is it possible to differentiate between different frustrations? To which I found a tentative slightly surprising answer.
Since hypermedia frameworks are interactive they have on some dimensions more in common with digital games than with multimedia. A single video does not allow for much interactivity. However, with XIMPEL it is possible to create a point and click adventure game. Therefore, I chose to review the literature on engagement and frustration regarding digital games. A second motivation to do this is because XIMPEL is an intentional intersection between hypermedia and gaming.
My approach for reviewing this literature is as follows: I draw from my own previous literature review, written in 2015 about what frustration and engagement is and add to that review with new research findings. I furthermore, review literature on how engagement but most notably frustration are measured and classified. The result of this literature review will influence my choice of measures that I will include in creating a logging framework for XIMPEL. Furthermore, these measures will have a focus on accessible non-customized hardware, which means conventional hardware available on most laptops and tables. Another result is a literature review itself, which is guided for design and not for comprehensive review but because of that an example of bridging literature to implementation.

5.2 What is frustration

In gaming, to frustrate a player means to block their progress. Or as Gilleade and Dix state it, it is: “that which arises when the progress a user is making towards achieving a given goal is impeded.” [38] In general, players will feel frustrated when they realize that the blocking of their progress has happened.
Despite that this definition has been made in a gaming context, it seems to be quite general. Studies about completely different topics (e.g. human motivation) give very similar definitions, albeit implicit. Take for example (emphasis by me): “Recently, it has further been recognized that beyond measuring need satisfaction versus the lack thereof, needs can also be *actively blocked* or the growth potential of individuals, *the frustration of these needs* would elicit defensiveness, ill-being, and even psychopathology.” [13] Because of this generality, the definition of Gilleade and Dix will be used.

5.2.1 Some nuances regarding frustration

It matters to whom or what the frustration is attributed to. If the frustration is attributed to oneself it will not cause quitting a game. If, however, it is attributed to the game itself (e.g. lag or too difficult AI) or to another person (e.g. the co-player did not perform well in a task) then the player will quit. This is indeed a quite subjective experience. More forgiving players might attribute the frustration more to themselves than to their co-player, for example. In my literature review, I provided some evidence for this claim but the strongest evidence for it has been found in a recent 2017 study about near-misses in Candy Crush.
The near-miss is a form of frustration where attribution of a failure, which was almost a success, is attributed to oneself. The canonical example of a near-miss is a slot machine that almost shows three lucky number sevens, but the third seven goes a little bit too far and reaches over the line, leaving itself at the right bottom corner of the slot machine (see figure F.7).
In the study, researchers found that the near-misses in Candy Crush increase the wanting to play and frustration [57]. This is clear evidence for the idea that the attribution of frustration matters since players believe they are able to achieve their goal. For XIMPEL authors, understanding this distinction is important when their application resembles a game.
However, when the XIMPEL application resembles multimedia more in the sense of that it is a XIMPEL application in which the user is passive, then the attribution of frustration is in almost all cases on the XIMPEL application itself. When a user is passive, he or she cannot do much in the first place and will therefore most likely place the blame on the application. That blame may be placed on several factors such as: topic mismatch, badly designed applications, the XIMPEL control buttons, or a terrible quality of experience (when watching YouTube videos or other networked media types).
XIMPEL authors need to consider to what extent they deem frustration and the willingness to quit important. In some contexts, frustration matters less. For example, when students are made to watch any presentation during their student lifetime at university, frustration will only lead to disengagement but not to quitting. One might think that this is just as bad, but one study shows this is not always the case.
Computer scientists at the University of Saskatchewa in Canada created a game to create interpersonal trust in which they showcase: a literature review on what interpersonal trust is, a literature review on how it is developed in real-life and via digital games and an experiment if their game fosters trust. The interesting result is that their game was found to be frustrating. They remark: “Our game was strongly affected by networking issues, which made the game more frustrating and difficult than we expected. ... participants in the game condition scored low on competence and high on tension. Comments from the debrief as well as the recordings of the game session confirm that many participants experienced a frustrating, ‘buggy’ game, rather than the playful experience we intended.” [25] They also state that this situation caused the game to become an out-group and the players to become an in-group, which

fosters trust[1]. A game does not have to be fun to reach its designed goal [25]. It could even be very frustrating, it could even be a negative experience.

5.3 Capturing data for analyzing frustration

Since XIMPEL is a framework with the browser in mind, I have assumed that most users only have access to standardized hardware. This is because at this moment XIMPEL is used by students at the Vrije Universiteit Amsterdam and by organizations who use it in giant tablet installations. Because of time-constraints I used measures that I could only find on my MacBook Pro (2015), since that is the hardware available to me.
This constraint takes precedence over evaluating academic literature. No organization has given me the tools to do something else and I do not have the funds available to buy specialized hardware. However, some of the reviewed literature (see the next section) does use specialized hardware. Despite this mismatch, these articles provided inspiration nonetheless on the nature of frustration, measuring it or classifying it.

5.3.1 Frustration measures: clicks, mouse speed, user XIMPEL subject history and facial expressions

XIMPEL users tap or click on overlays, which means that taps (not available on my MacBook Pro) or mouse clicks need to be measures. Furthermore, measuring mouse moves at the exact moment when they are made allows for the inference of acceleration or deceleration of the mouse and some estimated average speed in general. Since XIMPEL developers also know on which x and y coordinates their overlays are, they can also analyze through the mouse position if the user is standing on an overlay or media item.
Finally, measuring the starting time and which subjects the users are in at any moment allows for the construction of a path where users have been and for how long. This used to be in the old XIMPEL framework (written in ActionScript) but has not yet been implemented when it was ported to JavaScript.
Measuring: mouse clicks (x, y and time), mouse moves (x, y and time), starting time of playback per subject (in milliseconds) and the subject ID forms the basis of measuring frustration. It is quite unfortunate that these measures are relatively indirect. Frustration is a feeling first and foremost and gets expressed through the user in a variety of ways.
The need to obtain more direct data was there since these measures are perhaps too indirect to be useful for classifying frustration. Maybe it would be possible to capture frustration through the use of a web cam. While it is not as standardized (or unobtrusive) it may help greatly in the situations in which it can be used. So I decided to make that an optional measure. Sending web cam data directly is a lot of data to store. Therefore, I decided to classify facial expressions directly through the CLMtrackr JavaScript library. It detects: joy, anger, surprise and sadness. It has to be noted that this is a trade-off to make. Does a data capturing server store video streams or does it store classified data? The disadvantage of the former is that there is a lot of data to store, the disadvantage of the latter is that the data contains less information since there is a lot more one can do with pure video data than classified facial expressions. How this classification works is explained later in the chapter.
The use of CLMtrackr and classifying emotions related to frustration have two reasons. (1) Detecting facial expressions in the browser in general is difficult, so it is better to work with libraries already in place for this. As of writing, this is the only library I came across. And (2) in the reviewed literature on facial expressions (see section 5.4.2.3), there is a strong argument to be made to not measure the

[1]I will skim over the fact that they implied on some level that a game is a person (an out-group in this case). Media like games, are social things that we project human values on. Unfortunately, I know too little to academically defend this position and will defer to my media studies and other social science colleagues. These types of things should be discussed more often in computer science literature in order to have stronger assumptions and assertions.

facial expression of frustration. With that said, the detection of anger may be useful since frustration and anger are very related through the frustration-aggression hypothesis [89]. Sadness may be useful as well since it can be caused by a hopeless variant of frustration [55].

5.3.2 Software architecture and implementation for capturing data

All the mouse click, mouse moves, starting time, subject ID and classified facial expressions are sent to a NodeJS server. That server created a session ID to the client beforehand (including the date and the time) and stores everything in a postgres database. The data stored can later be retrieved for classification. The reason for not doing it on the fly is because it is computationally expensive and perhaps even wasteful. The detection of frustration is only relevant when developers want to improve the UX of XIMPEL. NodeJS with Express has been used because developing with an unopinionated microframework allows for the quick creation of routes that stores data in a database. The data can be queried from the database using any kind of program.
Independent parallel work has been done in similar fashion by the programmers of the University of Oslo who develop XIMPEL applications specifically for tablets. Instead of using NodeJS and Express, Python was used in combination with Flask, which leads to a more or less similar communication pattern between XIMPEL and the server-side data capture application [43].

5.4 Classifying the measures as frustration

Now that we have a way of measuring raw data for frustration, the next question is: how to determine when a user is frustrated with this data? The approach for this is to first look at previous research and then conceptualize a possible classification method.
I do need to note that I did not implement a classification algorithm. It is outside of the scope of this project to implement it. However, conceptualizing a first iteration of such an algorithm or algorithm design approach leaves opportunity for further realization in future work.

5.4.1 Related Literature

Depping, Mandryk, Johanson, Bowey and Thomson used four inputs (galvanic skin response, heart rate and EMG frowning and EMG smiling) in order to calculate arousal and valence values. A certain combination of arousal and valence (which is a range from displeasure to pleasure) would yield a particular emotional state: boredom, challenge, excitement, frustration, and fun. Each input is a time series that would be used to create a time series for arousal and valence, called AV-space (figure 1, 3, 14 and 15 in their paper [59] are helpful visual aids for this explanation).
For example, increasing galvanic skin response would be mapped to increasing arousal. Low or high galvanic skin response would be modulated with heart rate data, where a low heart rate would result with a lower arousal level than high heart rate data. EMG Smiling would mean that valence levels increased and EMG frowning would mean that valence decreased. They had other rules which have been written in more detail in [59].
Then, they created a mapping from AV-space to emotions. In particular, frustration needs to have a high arousal (e.g. high galvanic skin response and high heart rate) and low to medium valence (i.e. displeasure, e.g. frowning on the EMG). They discuss all their emotional states in their paper [60] and go into more detail in another paper [59]. Through a user study they found that their predicted model of frustration did not coincide with self-reported frustration. This was not the case with for the emotions fun and excitement [60].
Researchers in Cyprus and Portugal created a new mouse called CogniMouse [78] for older computer users. With this mouse they hoped to measure frustration since some older computer users have some difficulties using computers at work. While I am not developing my own computer mouse, it is interesting to see which extra measures the research team was trying to get. The sensors in the

mouse are a: "galvanic skin response sensor, temperature sensor, inertial measurement unit (IMU), grip/pressure sensor, and heart rate sensor." [78] They used a classification algorithm that employs "Bayesian-based formalism inspired on conditional probability distributions." [78] This means that they used a formula that constantly calculated a probability to what extent the user was frustrated. They used: grip force, acceleration vector and click stream frequency as inputs. The formula itself is:

$$P(Frus|Grip, Acc, Click) = \frac{P(Frus) * P(Grip|Frus) * P(Acc|Frus) * P(Click|Frus)}{P(Grip) * P(Acc) * P(Click)}$$

[78]
The formula clearly shows the Bayesian nature of the classifier. They have a planned user study and present a use case scenario of a middle aged woman who would experience issues using a new computer system. The CogniMouse noticed her frustration and helped her by presenting a graphical help wizard. In their future work they will add more ways to track the user, such as an eye tracker.
In the study of [58] an annotator annotated the signs of frustration from web cam recordings, where students in those recordings made use of web based programming tutoring meant for a Java programming course for beginners within University of Singapore. Students mouse clicks, keystrokes and actions were written into log files. This was compared to the annotations on a time scale. After this comparison, relevant extracted metrics were: "the mean and median key latencies, number of keys, wait time (duration longer than 1 second with no key inputs), back space and delete key latency and frequencies and the frequencies of mouse clicks. The interaction features include the number of compilations, number of errors encountered, number of exercises completed and the duration of time spent working on the exercises." [58]
The extracted features were aggregated into a sliding window with sizes of 30, 60, 90, 120, 150 and 180 seconds; where windows would overlap each other for one-third of whatever window size was determined. Frustration would be observed per sliding window. If the frustration occurred in the overlapping sliding window part, then both sliding windows were annotated as a frustrating event. Changing the window width from 30 seconds to a 180 seconds − and everything in between − would have consequences for the detection of frustration. If someone gets frustrated within every 3 minutes, then a window size at the highest width would always be marked as frustrating, this would not be the case for lower window width sizes. The problem with lower window width sizes is that there might be insufficient data to determine anything at all.
To classify frustration within these sliding windows they trained a Bayesian network and compared it to a naive Bayes model. The performance of the Bayesian network compared to the naive Bayes model was a significant improvement. The performance improvements were 32.79% (under the curve, the authors did not define what this means), 32.73% (accuracy, "the number of correctly identified instances divided by the total number of instances" [58]) and 2.53% (sensitivity which "measures the true positive rate or the proportion of positives that are correctly identified as such while specificity measures the proportion of negatives that are correctly identified as such" [58]).
Instead of classifying frustration, one could also classify something else such as stress. **Rodrigues, Gonçalves, Carneiro, Novais and Fdez-Riverola** did this [83] in a study for which they created a stress detection tool in for the e-learning platform Moodle. They created a system called the Dynamic Student Assessment Module (DSAM) which has all kinds of components for detecting the mood of a student. The DSAM does this in two ways: by asking students through a questionnaire and by looking at their behavior. They looked at their behavior through "facial analysis, mouse analysis, keyboard analysis, and log analysis (through Moodle logs)." [83] The features they analyze are: "the number of mouse clicks per minute, the average duration of mouse clicks (from the button-down to the button-up event), the maximum, minimum and average mouse speeds, the keystroke rate (strokes per second), the average duration of a keystroke (from the key-down to the key-up event) and performance measurements." [83] They asked 10 students to do two similar programming tasks. In one programming task they were stressed through the use of a time limit and being told that this assignment was very important for their future (academic) career. They found that the following JavaScript events were fired a lot more when students were stressed: key down, key up, mouse down, mouse up, mouse wheel

and mouse movement. In some cases it was 5 times as high, in other cases about 1.5 times as high. From these differences they claim they can detect stress.

5.4.2 Proposed approach for classifying frustration in XIMPEL

In the related work section a lot of input sources have been described (too many to list). Classification techniques were: rule-based, Bayesian and comparing it to a control group. Furthermore, [60] wrote about another author using Markov chains. It seems to be the case that any probabilistic classification technique can be tried. Other than probabilistic classification techniques, rule-based techniques sometimes seem to work – comparing results to a control group is a specific example of a rule-based technique.

In XIMPEL we measure: mouse clicks, mouse moves, a historical trail (with time) of which subjects a user goes to and facial expressions. Since a hypermedia-like application could be different than a game or a generic application due to watching video, some measurements will not always be the same. For example, someone could get frustrated at miss-clicking on an overlay, finally the user will click on the overlay. Compared to a non-frustrated user, the frustrated user will likely have more clicks. But if a user is frustrated because a certain video is boring and he or she cannot go to the next subject, the frustration may only show via their facial expression and not through their use of how they use the computer mouse.

Since it seems more important to give insight to XIMPEL authors when a user is *possibly* frustrated, I put more emphasis on detecting all true positives. In order to detect all true positives, one must be willing to accept that there will be possible false positives. In order to get all true positives one must minimize false negative occurrences of frustration and thus be forced to have more false positives. At the same time, looking into a particular possible frustrating episode must be worthwhile so having a signal to noise ratio of 10 to 1 might be a bit much since it might take 2 to 5 minutes to check a possible occurrence of frustration (e.g. checking the data for a particular person for a possible frustrating occurrence). A signal to noise ratio of 1 to 3 would be much more acceptable. For example, if the signal to noise ratio is 1 to 3, then it would take 20 minutes at worst to look at instances of possible frustration.

So the requirements for our classifier are:

- A signal to noise ratio of 1 to 3 or lower. A signal is a frustrating event, noise is the complement of it.
- If the first requirement has no surpassed threshold (i.e. a ratio lower than 1 to 3), then get more, or even all, occurrences of frustration.
- Be able to pickup on frustration when the user is not using computer peripherals (e.g. listening to audio or watching a video).
- Be able to pickup on frustration when the user is using computer peripherals.

Time constraints prevent me to create such a classifier. Nevertheless, I will write my ideas down for future work related purposes. Since interactive video, hypermedia and XIMPEL applications are a bit different than other applications, an observation study should be done as to what makes users frustrated. From such an observation study it should be possible to see what frustrated behaviors users have and how this could be possibly measured. The measures that I programmed for are a mix of a best guess inspired by literature and hardware constraints.

Classification should depend on the requirements for detecting frustration within XIMPEL. These requirements are listed above and were made with the rationale of why we should detect user frustration within XIMPEL applications in the first place which is to improve UX. The requirements are fairly broad, which means that a potential solution does not have to be perfect. It is likely that multiple solutions are possible.

My approach to classifying frustration would be to create a simple classifier per input type. So there is a mouse movement frustration classifier, for example, and also a facial expression frustration classifier. If one of these classifiers gives an alert, then frustration is possibly detected. This is possibly too straightforward, but it allows to get more experience as to which input types explain frustration

independently from the other input types. Testing which classifiers give too many false positives is future work. The question that I will attempt to answer now is how does each classifier work?
While this is one question and directly applies to mouse clicks, mouse moves, user trials and time between subjects; it applies differently to detecting frustration in facial expressions. CLMtrackr already classifies facial expressions. So the question regarding facial expressions goes much deeper and this depth is written down. Then, I realized that in my first literature search on detecting frustration in users I did not find anything about how to detect frustration in facial expressions, so I did a separate literature search for that specific question. So the specific questions regarding frustration in facial expressions are: (1) how does the current classification algorithm work regarding basic emotions? (2) do classified basic emotions help classify frustration, if so, how? And (3) why not detect frustration as a facial expression directly? Perhaps noticeable, questions 2 and 3 are related to how does the facial expression classifier work?

5.4.2.1 Facial expressions

The JavaScript library CLMtrackr classifies anger, sadness, happiness and surprise and gives all four a value between 0 and 1. CLMtrackr does this through constrained local models (CLM) among other things. It technically also classifies contempt and fear but not good enough according to a quick discussion with the CLMtrackr author on the Github Issues page [75], which is why these facial expressions are not classified. CLMtrackr trained its facial recognition abilities through the MUCT Landmarked Face Database (MUCT: Millborrow – the author – University of Cape Town) [63]. This database has 76 points (called landmarks) annotated on 3755 faces. It is possible to see an example image of such an annotated face on their homepage [63].
What follows now is a high level explanation of how this works. However, the mathematics will not really be explained since they have been explained quite well in the online resource of [72], which I recommend to anyone interested in the technical details of facial detection. That article also points to amazing resources on CLM itself. One does need to read the article critically since I captured it on a subtle error (which has been resolved since) [90].
The algorithm that Clmtracker uses is called subspace constrained mean-shifts, which is a form of a CLM and is authored by [91]. CLM algorithms in general have two strategies that happen iteratively until some optimum has been found as explained in [91].

- 1. Exhaustive local search: perform a local search for each landmark in the model around their current point estimate. Use a classifier that will generate a probability map and associate the landmark to the highest probability pixel (or collection of pixels) on the map.
- 2. Optimization: it could be that some landmark to pixel associations are a bad fit because there were no high probability pixels on the map. Shift their point estimate to somewhere else by calculating where there are likely higher probabilities.

The mathematical techniques that make this possible are explained below. I do have to caution the reader that I intuitively understand what is happening, but I do not understand the mean-shift algorithm and associated algorithms well enough to explain its details. Such details can be read in [91]. Furthermore, data cleaning techniques that are used are not presented at all in this thesis (see [91]).
The author of CLMtrackr built a model through the use of Principal Component Analysis (PCA), which informally speaking is a method by summarizing as much variance into a component. This is done by first calculating the mean points of all the landmarks of all the annotated faces, then PCA is used to extract the variations as components (they can also be seen as linear combinations of vectors). PCA has the property that the first component accounts for the most variance, the second component for the second most variance and so on. This means that with a few components most variance has been accounted for. PCA is therefore really useful as a data summarization technique. The components

themselves are uncorrelated to each other[2]. The PCA model of a generalized face can be viewed at [73]. Playing with it gives a better intuitive understanding of what PCA does regarding finding the average model of a face and the variations of that on a per component basis.
The model has 70 points. Each point in the model has been associated to one classifier during training, meaning there are 70 classifiers for the model in total. Each classifier has been trained by cropping a small patch of its associated annotated point per facial image (3755 facial patches per annotated point in total). The classifier for such a patch is a logistic regression classifier with a support vector machine linear kernel (and a MOSSE filter which is of less importance). The reason it is a logistic classifier with a linear kernel[3] is because it "is what the original paper suggests." [72]
Exhaustive local search. When the classifiers are used it crops a local (i.e. small) search rectangle around its initial position and calculates a probability map of pixels which are aligned with the landmark. When this works, such a map has a few high probabilities lying around in the center, of which one of them is the highest and that pixel value (or grid of pixels) will be associated to the landmark. This is done for all 70 landmarks in the model.
Optimization. However, doing this there is one problem. Since the local rectangle might be too local. The problem with this is that some classifiers cast their local rectangle in a different area compared to where the actual facial feature is (e.g. an eye). In order to counter this, a second step is being done that is akin to gradient descent. This is done by shifting the point of a landmark, which means when it will cast a new rectangle to search for the highest probability it will find new values. The method is called a mean-shift. This mean-shift is determined by using expectation-maximization.
By doing all of this we have found faces. What we have not found are facial expressions. These are classified using logistic regression. The parameters of the facial model are used as features. To get training data for the regression, images of people expressing emotions have been annotated and projected on the PCA decomposition. By doing this, the closest parametrization is achieved. What is not done but could be done for future work is to first determine a neutral baseline, since it is not clear how expressive emotional faces are beforehand. The emotion classifier can also be viewed since the regression coefficients can be used as parameters for the facial model. You can view it at [74].

5.4.2.2 Detecting frustration through CLMtrackr

The facial expression of anger is interpreted as frustration. The facial expressions of happiness and surprise is interpreted as its opposite: engagement. Sadness might be a meaningful measure of frustration as well since it has shown to correlate as much to frustration as anger [55].
It is important to keep the classification of engagement since it may give extra data if another classifier does classify that the user is frustrated. In most cases facial expressions should take precedence over mouse clicks, mouse moves and history graphs because a facial expression conveys how a user feels, much more so than the other measurements. The reason for that is that facial expressions are much more connected to emotions than the other units of measurements since facial expressions are a display of emotions.
It may seem a bit of a cop out to use anger and sadness as a proxy of frustration. However, perhaps it is not. Anger is the result of a particular frustration. When a user is frustrated at anything other than him or herself, he or she will become angry [89]. Therefore, the detection of anger is a possible detection of frustration. Other research (not included in the literature review) that point to this direction is research about consumer anger [36]. The authors do not explicitly write about frustration, but in this study all the sub-categories of anger are related to frustration in the conceptual sense,

[2]An in-depth mathematical student explanation on can be found on `http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf`. The best visual explanation about PCA is to be found here: `http://setosa.io/ev/principal-component-analysis/`. The best visual explanation for eigenvectors is here: `http://setosa.io/ev/eigenvectors-and-eigenvalues/`.

[3]The author Audun Mathias Øygard calls it an SVM kernel. However, digging into the source code it seems that the so-called SVM kernel is a linear kernel. I had to dig deeper since there is no such thing as an SVM kernel. The kernel can be seen here with a search query in the Github repository: `https://github.com/auduno/clmtools/search?utf8=%E2%9C%93&q=kernel` This has been resolved during a brief discussion with him on Github Issues [90].

they are: broken promises, unfairness and expressed hostility (by service employees to customers). Neuroscientific research shows that "poorly controlled frustration, manifested as exaggerated anger." [102] Research on emotion regulation implies the same, in one paper it is stated that "a low frustration tolerance is related to trait anger and a higher level of frustration tolerance is related to lower levels of anger and longer persistence on difficult tasks." [94]
Kuppens, van Mechelen, Smits and de Boeck gave strong nuances to this view. In their article the appraisal basis of anger was evaluated to which components were necessary and sufficient. The researchers conducted two studies and they found that "other accountability and arrogant entitlement, as instance of unfairness, are specific appraisals for anger." [55] They suggest that frustration and having an antagonistic action tendency co-occurs. This is easier to believe for having an antagonistic action tendency, since only one study found an association with anger. However, the reason they state for the feeling frustration leaves much to be desired. They studied the *feeling* of frustration in their first study and *goal blocking* in their second study. They found that goal blocking is not associated with anger but feeling frustrated is. They also found that feeling frustrated is associated with sadness [55].
Putting anger and sadness side by side from their first study, it is observed that sadness has frustration as an appraisal-action tendency component and did not have any significant correlation with the other appraisal-action tendencies. Interpreting these results it could mean that sadness occurs when one feels hopeless because of frustration. Whereas if one feels angry because of frustration it is because of someone (or perhaps something) else.
The result of Peter Kuppens et al. give an indication that indicators on sadness may be important as well. Therefore, the final conclusion of detecting frustration in facial expressions is that sadness and anger combined are an interesting proxy for frustration and joy and hope for engagement.

5.4.2.3 Detecting the facial expression of frustration

The complexity of writing a thesis while programming is that a program can become obsolete when new literature is found. It is quite possible for a researcher to believe they are up to date for a certain topic, only to discover there was extra information out there available by a slight alteration of their search queries which they had not considered. This arguably happened for this project. Indeed, an exploratory thesis such as this one is quite prone to requirements creep. Additional found literature seemed to answer one particular question: how does one detect the facial expression of frustration directly?
The following articles have not been found initially, because the importance of detecting facial expressions at all – on a practical level – had more importance. Both studies used the Facial Action Coding System (FACS) which divides the face up in certain sections. These sections are called action units. Humans code these action units on a 1 to 5 point scale in terms intensity. These 5 points are labeled as: (1) trace, (2) slight, (3) marked or pronounced, (4) severe, (5) maximum. Other modifiers are R and L which convey information that assymetrical movements happen on the right or left side of the face.
In one article action units (AUs): 1, 2 and 14 (14 is of secondary importance) were associated with frustration. AUs are respectively named after their movements. For example, AU 1 is called the inner brow raiser, AU 2 is called the outer brow raiser and AU 14 is called the dimpler. Moreover, action unit 1 and 2 often occured together and "triggered each other" [18] which means that "a raised inner brow tends to trigger a raised outer brow, and vice versa." [18]
An article written five years later (in 2013) compared the results of the authors of [18], though not specifically from the paper that is outlined in the previous paragraph but a paper written by the same authors a year later [21], which describes two studies one of them being the study of [18]. They found completely different results since they noted that frustration has a strong association with AU 4. They note that AU 1 was associated with whether a student self-reported their tutoring session (with JavaTutor) to be worthwhile. AU 2 was associated with whether a student self-reported the feeling of being rushed during a tutoring session. AU 14 was associated with whether a student self-

reported with whether how successful he or she felt in the accomplishment of a task. All the AUs they found for their dependent variables were: AU 1, AU 2, AU 4, AU 7 and AU 14. They measured self-reported frustration as: “How insecure, discouraged, irritated, stressed and annoyed were you?” [40] It could be the case that they measured something different than frustration since: “discouraged, irritated, stressed and annoyed” is not particular to only frustration, whereas the other questions in their post-session survey seem to be related to frustration. The authors note a similarity with how they detected AU: 1, 2 and 14; with its relationship to frustration in previous research as well. They seem to suggest that these action units are related to frustration. Finally, their research shows how difficult it is to replicate previous research in this area.
This difficulty is also seen in the final article found for this section, which has been peer-reviewed but will be published later in 2018. The topic of research was detecting frustration in frustrated drivers. In this study the authors found that they were mostly in line with the action units that also have been found by Ekman, which are AU: 4, 14, 23 and 24 [47]. These action units are only partially in line with research from the previous paragraphs written. The authors from the previous paragraphs found evidence for AU: 1, 2 and 14 [18, 40]. The authors of [47] note that these action units have been linked to other emotions such as surprise (AU 1 and AU 2) [47]. In their final results they found no significant result for AU 4 and AU 14. They did find a significant result for the association of AU 23 and frustration. They also did find a significant result for the association of AU 24 and frustration. They furthermore did an exploratory analysis, meaning they also looked between a difference of their control condition and experimental condition while considering all the other action units. In this analysis they found that AU: 10, 12, 17 and 20 also had significant differences between the experimental and control condition.
It could be that frustrated driving does not capture the same type of frustration as a facial expression compared to computer-mediated learning – which the other two studies in this section were about. It could be that there are issues with the methodology. What it at the very least shows is that finding the facial expression of frustration directly might be possible for specific domains, such as computer-mediated learning or frustrated driving, but it proves to be much more difficult for detecting frustration as a facial expression in general.
Since it may or may not be possible to directly detect frustration it may seem to be a better idea to stay with the idea of having proxies for frustration such as anger and sadness. Anger and sadness as facial expressions have been studied more in-depth compared to frustration. While it will increase the signal to noise ratio, it is better to have a signal at all since research on finding frustration directly argue on what the signal is. User studies would need to be done to show if this approach has any merit.

5.4.2.4 Mouse clicks and mouse moves

In XIMPEL a user navigates with the mouse either to interact with the XIMPEL presentation (through overlays, form input or multiple choice questions), to use the XIMPEL controls (stop, pause and play) or to use a feature of the browser they are using. By being aware of this, it is obvious that a classifier for mouse clicks and mouse moves is only able to detect frustration when the user interacts with the video. This frustration is hypothesized to occur in two ways.
The first one is that for some reason an interactive feature of XIMPEL does not fully work. For example, a user types out their nickname in the XIMPEL application and when the user clicks on submit nothing happens. Sometimes this can happen in any application and most users would click again. Yet, if nothing happens, the chance of frustration is really high since the user is not successful in reaching his or her goal. In the data one would see a lot of mouse clicks and little mouse moves at the same location. This form of frustration is obviously UI-frustration and is possibly as easy to classify by filtering the data for a lot of mouse clicks on the same place. If a lot of users have issues at the same location, then it is abundantly clear that the UX need to be improved.
The research of [83] showed that there was more mouse movement, mouse down and mouse up events when programming students were stressed. Programming is a more interactive experience than a

hypermedia presentation so not all metrics may show up in the same way when it comes to hypermedia presentations. However, having experienced hypermedia presentations myself, one hypothesis is that it could be the case that mouse movement is more rapid in a frustrating experience compared to a non-frustrating experience. In these cases a user feels he or she is waiting too long for going to the next subject. This is a form of content-frustration – arguably also UI-frustration, since there is no time-scrubbing.

5.4.2.5 User trails and time between subjects

When it comes to analyzing the history graphs, the most important questions to ask are: (1) did the user spend the amount of intended time for each XIMPEL subject? And (2) if the user did not, then did the user follow an acceptable trial within that graph? It could be presumed that users feel relatively engaged when they follow a XIMPEL presentation as intended as opposed to when they do not. For example, some XIMPEL presentations present overlays within any subject for the reason that a user can always opt to go to the next subject. This is to alleviate frustration, but in some presentations this is also a problem, depending on the intent and structure of the presentation. Some XIMPEL presentations intend that every piece of content is read and seen and some XIMPEL presentations do not. For those that do, the second question seems more relevant.
An example of this is the Zaanse Schans (a place in The Netherlands) example on the ximpel.net, where it is possible to do quizzes for every informative video shown, but users are also able to skip them. The intended message would be a bit lost if users did not take the quiz since that helps for memory retention. However, it could be argued that it is acceptable that all quizzes were not experienced by users as long as they view al the video content. If users do not view all video content and quit prematurely, then that may be seen as a sign of frustration.
User trails and time between subjects have two questions that seem quite XIMPEL presentation independent. This is another reason why frustration classification has to be done after the fact since acceptable paths and acceptable times can only be determined after a story graph and design of a XIMPEL presentation is finished. Moreover, the idea of what is an acceptable history trail and time between subjects may change in the mind of the XIMPEL author. This could be even years later. For example, a new improvement for the XIMPEL presentation may be added and it changes the idea of what is an acceptable history graph.

5.4.2.6 Conclusion

The philosophy of XIMPEL is being pragmatic. In this case the pragmatic approach is to classify certain features that I am able to do and for other features I offer my best approximations of it. It would be best if frustration could be automated, but in detecting frustration for XIMPEL it would be as interesting to be able to filter data instead of full classification. This approach that you do not let the computer do all the work is an approach in AI-based game-design as well (e.g. see [51]).
The approach that I outlined is to create a simple classifier per input type. Having multiple classifiers being good at one type of classification is not uncommon since that is also part of the algorithm regarding how facial expressions are classified [72]. The input types are: facial expressions, mouse clicks, mouse moves and user trails per subject (how long have they been at each subject in a XIMPEL presentation?), also called a history graph. My approach for facial expressions is to detect anger and sadness as a proxy for frustration and joy and surprise as a proxy for engagement. A baseline needs to be determined, since it could be the case that the classification of certain facial expressions could have high values already at baseline (e.g. 0.8). The suggestion for mouse clicks for detecting frustration is (1) local and frequent clicks; (2) faster mouse moves. For mouse clicks, a baseline needs to be determined beforehand to see the normal frequency of clicks and speed of mouse moves. Detecting frustration in user trials seems not possible but dissatisfaction may be. The outlined approach is to see if the user (1) spends the intended amount of time per subject and (2) follows a trial deemed acceptable by the author. If this is not the case, then the user might be dissatisfied, especially if other

classifiers point towards frustration. It most likely is a thesis in itself to implement this fully and do a user study on it.

5.5 Future work

Future work could be done in various areas. There are a couple of areas of improvement. These are (1) detecting frustration and engagement directly, (2) researching the associations between mouse moves and mouse clicks with facial expressions, (3) improving the classification abilities of the current approach (e.g. better facial expressions detection), (4) implementing the current approach in code and (5) rewrite the unique parts of the NodeJS server-side application to Python with Flask and merge it with the Microtiks server made by Hugo Huurdeman and Dan Michael O. Heggø (the data capturing server made for XIMPEL in Norway). Some of these opportunities will be outlined more in-depth.

5.5.1 Detecting frustration directly

Research on detecting frustration directly is tough. There are different experimental outcomes with detecting frustration through facial expressions [18, 40, 47]. One outstanding question that generates more questions than answers is: to what extent is frustration a universal emotion or facial expression? Many emotion researchers claim it is not a basic emotion [69]. Is it then universal in western or eastern cultures? Or are there even too many differences per country? Or is it even the case that frustration, as a facial expression, is only measurable per topic? Maybe frustrated drivers look differently compared to frustrated programmers.
If the extent of the universality or lack of it is unknown regarding frustration, then research on detecting it will be more difficult to compare. Assume that it is true that the facial expression of frustration is different per topic but researchers do not know this. Then, it could be the case that researchers on frustration simply think that other frustration researchers that study frustration through another topic are simply wrong. Just assuming that frustration has some form of universality is assuming something very fundamental. If that fundament turns out to be false, then we are not standing on the shoulders of giants but on towers of air castles.
So future work regarding detecting frustration directly is possible, but from a scientific standpoint also a big risk, because the aforementioned assumption can cause a lot of confusion. Therefore, more fundamental research is needed.

5.5.2 Researching the associations between mouse moves, mouse clicks and facial expressions

Another line of future research is researching the system that already has been conceptualized. Looking for associations between mouse moves, mouse clicks, joy, anger, sadness and surprise is a way to do this. How to best go about this research is a tougher question. The main vague answer is to train some model ("some" being the vaguest part).
For example, users could interact with a XIMPEL presentation online. Everything would need to be recorded, which the current system is capable of doing. Then, we could use any machine learning algorithm to see if it is able to predict one of the four facial expressions given mouse clicks and mouse moves. The danger of this approach is that spurious associations probably exist. Therefore, this approach may only benefit the predictability for detecting one of the four facial expressions, regarding mouse clicks and mouse moves. If we are not able to directly detect when a user is frustrated, then detecting sadness or anger seems like a good way to move forward.
One issue with this approach is that it assumes correct classification of: joy, anger, sadness and surprise. The current method does not always do this. So there needs to be a self-report check from the user to see if they think the classifier correctly classifies, as the current classifier does so better on some humans than others.

5.5.3 Improving facial expression classification

Current issues regarding facial expression classification is that it uses straightforward logistic regression as a classifier. This can be improved by looking into different classification algorithms. Note, this has nothing to do with face detection, which uses constrained local models combined with a mean shift[4], but with the classification of joy, sadness, anger and surprise. Secondly, no baseline is made. For example, when I look in the web cam, the reading always is that I look very angry or sad (0.8). Two things are possible here: (1) I am angry and I do not know it, or (2) my eyebrows frown down because those are the type of eyebrows I have which are immediately classified as angry. This issue has also been stated in the final paragraph of section 5.5.2. If a baseline check would be made, then the 0.8 angry value would be the new neutral.

[4]Of which logistic regression with a linear kernel plays a part, but that in itself is not straightforward logistic regression.

6

Exploration 5: exploring what time scrubbing mechanisms XIMPEL needs

During my talks with Hugo Huurdeman we both came independently to the conclusion that XIMPEL should have a time scrubbing feature that is related to all media types, especially for video. I talked about how to possibly implement this. Then, as I went on to implement it, I gradually realized that a time scrubbing feature for multiple parallel playing media is not straightforward at all. For this reason, I staved off development and thought about the different possible conceptual implementations of this. Time scrubbing in multiple parallel playing media is not the same compared to time scrubbing for one video. Different conceptual implementations will lead to very different behavioral outcomes when a user starts to use the time scrubbing feature. Since few people seem to have thought about this before, I will attempt to outline different time scrubbing scenarios. Unfortunately, no time scrubbing mechanism is implemented since there are too many design questions. The presentation and illustration of these questions is the result of this exploration.

6.1 Time scrubbing with videos

In normal videos time scrubbing is straightforward. There is a slider. With this slider the user is able to control at what time playback starts or skips to (see figure F.8). From a user experience standpoint the benefits are: being able to skim videos by time scrubbing through it, skipping unnecessary parts by showing small relevant segments and replaying a small segment after having not fully seen it due to (for example) a lack of focus.

The straightforwardness comes from the fact that videos are linear media. Time scrubbing allows for random access of linear media. Furthermore, videos are only one instantiation of media. Multiple videos playing at the same time would be multiple instantiations of media. Contrast that with media played in XIMPEL which could be: non-linear and have multiple instantiations of different forms of media such as one text block, two images, three audio tracks, four videos and five custom defined animations.

Research Questions and Contribution

What time scrubbing mechanisms are possible? The design questions that need to be asked will happen through the structure of XIMPEL. On the highest level this is a XIMPEL presentation seen as one entity (analogous to a book). The second highest level would be a subject (analogous to a chapter). For the third highest level, parallel playback will be assumed, because the tougher questions are found there – as can be read in this exploration. The third highest level are the media types itself (analogous to a paragraph). These levels do not exist in isolation and can interact.
The relevance to education is the ability to browse through content by a per second basis. This has not been possible with XIMPEL or other hypermedia frameworks such as SMIL. The reason it has not been possible is perhaps because it is a tough question to answer. It has not been possible to find an implementation because there are too many questions, which this chapter will reflect.
The relevance regarding the implications of parallel media playback is that without parallel media playback, time scrubbing in XIMPEL would be an easier problem to solve. It would still not be easy since parallel media playback is not the sole contributor for all the complexity regarding time scrubbing, but it is a big contributor of it.
The area of research this exploration is benefiting from is design and human-computer interaction.

6.2 Related work

Before we start getting into the wonderland that is time scrubbing, let us first look at related work in order to have some background knowledge. Some researchers also decided to present their work on YouTube. This approach seems to give a more intuitive introduction compared to reading the article written about their work. The in-text citations of these papers are made **bold** (e.g. [**1337**]). When you, dear reader, look at the bibliography a YouTube URL will be there for you, so you can watch a demonstration of their work. Since we are going to look at time scrubbing systems it may be useful to first see systems in action rather than read academic articles.
Previous research on time scrubbing and related video browsing methods has mostly – if not only – been done for linear videos. In my own literature research I found that the biggest trend in time scrubbing research has been manipulating objects in video [**67**, 93, 39, 27, 53]. The idea is that by manipulating an object within the video, a user is able to scrub along.
Thumbnail like strategies are another way of doing this. For example, one study presented an interface in which a user sees still images of the video in a grid on their smartphone. When a user taps on one of these images, another set of still images are shown that are closer in time around the first tapped still image [45]. Another example is a study that used still images in a tree like fashion. The deeper levels presented still images for more fine-grained time scrubbing control whereas the levels closer to the root node were more coarse [23].
Then, there is a line of research that improves the experience of time scrubbing in some sort of fashion. For example, by implementing multiple time scrub bars [81], altering the time scrubbing experience [**77**], by having automatic labels above or below the time scrub bar [37] or manual ones[1] or by altering fast-forwarding functionality [14].
Specifically for our interest, time scrubbing or related video browsing techniques have been developed purely for education and MOOCs. In one study, researchers improved the time scrub bar by looking at user statistics. Sections on the scrub bar where it has been stopped at or played at most were highlighted and when a user went over it with the mouse cursor it felt a bit resistant to continue to a non-highlighted part. It furthermore offers a search bar to search text in the transcript, which is

[1]The custom-made video player of the Harvard course CS50: Introduction to Computer Science does this.

mapped to the time it was said in the video. Finally, the most watched parts of the lecture have been automatically summarized [52]. In another study, researchers created a complete system meant for YouTube educational videos. They created a word cloud in which they utilized the x-axis (temporal ordering in the video), y-axis (temporal spread), font-size (prominence) and font-color (level of acoustic stress). They furthermore created a second word cloud that summarizes a part of the video. A third feature they have are video slides that show points of interest for the next and previous slides, they also have a time scrub bar, which indicates on some points the auditory stress on certain concepts [101].
More information on this topic can be found in the literature review of Schoeffmann, Hudelist and Huber [92]. The previous information has been found by conducting my own literature search. The literature survey of Schoeffmann et al. is more comprehensive. However, the work of the following authors is not included in the review [93, 39, 27, 53, 81, 37, 14, 52, 101]. This means that some work of 2013 and 2014 that I cited is not included and all works that I cited before 2009 and the year 2015 are not included.
Most conclusions of the reviewers are relevant for our purposes with XIMPEL. The reviewers found that many of the video browsing tools "try to follow metaphors from real life, such as film strips, multiple parallel, displays and simulated tape recorders." Also, "many interfaces make use of the third dimension." Most important of all, "it is also observable that many tools try to make navigation in videos more convenient by using concepts like sliders with rubber band effects, direct manipulation of objects and synchronized display of thumbnails." [92] It is important to note that these are all conclusions based on what researchers and developers create. The conclusions cannot necessarily be drawn for the user or user experience. As of now, it is still not known to what extent users are waiting for these type of improvements.
The reviewers formulated their biggest critique related to it: many studies did not perform a user evaluation. This means that a lot of studies have been peer reviewed and published on the idea of: having a justification based on an intuition, looking for related work, designing an own system and – in most but not all cases – presenting a prototype.
This criticism also applies to this thesis unfortunately. In my case there are two obstacles preventing me from doing a user study. (1) My research direction is different and therefore there is a lack of time to do user studies. (2) The only moment I would have corrected course on my thesis with regards to reading this information would have been the beginning, around exploration 1. Exploration 5, according to the time line of this thesis (no pun intended) is towards the end of it.
On another note, my own observation from the literature review is that very few theories and conceptual ideas about user behavior have been formed. There is, however, one study that did look at user behavior and attempted to infer a simple model out of that behavior. The researchers noticed that users mainly use the time scrub bar or simply watch the video from beginning and forward skip in time by a little bit, mostly using the mouse, and possibly restart the video to find what they need. What they did not do was make an educated guess and immediately jump to a specific point in time [16].
Unrelated, but worth mentioning is that one study in the literature review called 360 degree viewable video by the term *hypervideo* [66]. This may indicate that previous research on hypervideo between 1990 and 2009 is a bit forgotten and now everything that extends the idea of video could potentially be called hypervideo. Perhaps a better name, to avoid confusion, would be surround video or 360 video.

6.3 Within subject time scrubbing in XIMPEL

Without the parallel media player, time scrubbing in XIMPEL is a straightforward concept. The way one would scrub with YouTube can be copied and pasted to any media type in XIMPEL – including text, images and audio.

6.3.1 Time scrubbing: the difficulties introduced by the parallel player

However, combining time scrubbing with the parallel player makes it more complicated. Now, a subject has control over multiple media types playing at the same time. The first question that make this question complicated is: (idea 1) should the user be able to control time scrub bars of individual media items? (idea 2) Or should the user be able to manipulate a global time scrub bar that will slide all the media items at once? The next two figures show the distinction visually. The visual example is only with videos. It is left as an exercise to the reader to imagine this with mixed media types (see figure F.9 and F.10).
Manipulating the individual time scrub bars is still fairly straightforward (idea 1 and figure F.9), since the interface does not change. The time scrub bar interface is simply displayed more often in the XIMPEL application. The complicatedness, increases when one tries to answer the second question (idea 2 and figure F.10). It could be argued to choose for either ideas (1, the local multiple time scrub bars or 2, the global time scrub bar), both ideas need to have a conceptual realization.
Moreover, idea 1 and 2 could be combined together by having a global scrub bar and local scrub bars (see figure F.11). However, user studies would need to be done in order to validate the concept. Most web applications do not have multiple scrub bars, and therefore it might be a bit too overwhelming for the user.
The limitation of marrying these ideas is that when a user time scrubs the global, one a design decision needs to be taken. If a user already scrubbed one time scrub bar locally and then scrubs globally, three different scrubbing situations could occur (see figure F.12, F.13 and F.14). Either the global time scrub bar resets the local time scrub bar to the original mapping where the global time scrub bar believes it to be. For example, a global time scrub bar is at 50%, so all the local time scrub bars will be reset to 50%. Another possibility is that local time scrub bars that have been meddled with, will move along according to the offset of the global time scrub bar. For example, the global time scrub bar is scrubbed from 30% to 50%, representing a 20% increase. One local scrub bar has been scrubbed to 75% prior. Then, it will increase to 95% and the other local scrub bars, that have not been meddled with, will move to 50%. The final possibility is that the meddled local scrub bars do not move if they are ahead of the global scrub bar, and will catch up and reset to the value of the global scrub bar when they are behind. Figure F.12, F.13 and F.14 visually clarify these difficulties.

6.3.2 Time scrubbing: the difficulties introduced by users having choice

The real difficulty is introduced not by the parallel player but by XIMPEL and the nature of interactive video and choice itself. One could ask herself or himself: if one scrubs from 30% to 50% but on the 40th percentile of the time line an overlay should be presented with a choice, should that choice be known to the user or not? The user may not know that there would be an overlay on the 40th percentile and therefore may not be aware of the choice. The user may also be specifically skipping to the 50th percentile, because the user knows there is an overlay there and wants to skip it.
In a single YouTube video this is less of an issue. YouTube overlays tend to be links to different videos that are less coherent compared to a certain group of hypermedia applications in XIMPEL. The user experience (UX) designers of YouTube presumably have decided that it is okay to skip these overlays. But should the user experience of XIMPEL be similar?
Both UX proposals would need to be tested. And while it is easy to copy the UX of YouTube, it is a less easy question to create a new UX in the case that overlays should be presented to a time scrubbing user. In order to answer this question I am forced to take inspiration from other systems or create something of my own and justify it.
Furthermore, is it desirable in XIMPEL presentations to inform the user that he or she is skipping overlays? Or is it more desirable to inform the user it is skipping a moment of choice. In the first scenario all overlays would need to be known by the user in advance. In the second scenario the user only needs to know whether there is any overlay present at certain moments in time, it does not matter how many overlays there are.

For this question, I take inspiration from the time scrubbing mechanism that professor David J. Malan and his colleagues have developed for the course CS50 at Harvard. The reason for that is because it solves a similar problem and it was usable for me when I took the course years ago. The problem that they solved is that during a lecture a student would be introduced to multiple topics or sub-topics. Since the level difference between the students is quite big, some students may want to skip certain topics. By creating a time scrub bar that allows students to knowingly skip a certain lecture topic, they give the power (the choice) to students to do so. In other words, their problem is also a problem about choice, albeit a different form of choice.
Figure F.15 will show the user interface design. Like CS50 we highlight certain points of the time scrub bar. By doing this, the user is able to know when there will be a potential choice. When the user hovers over such a point, the user will see all possible choices that he or she can make at such a point. This interface is relevant for people who already went through a XIMPEL presentation. In some cases, the XIMPEL author may want to disable this feature since not knowing when certain decisions ought to be made potentially add value to the UX of a XIMPEL application.
Does this interface work well for one media item? Would such an interface scale for multiple media items? For that dear reader I ask you to use your imagination. Imagine multiple media items, for example, two videos, one audio and one image somewhere on the screen. Got it? Amazing! When all media items have their own time scrub bar – or in the case of the image at least a time line which you cannot interact with – then highlighting certain points on the time scrub bar or time line is not an issue. In this way it does scale. However, it is also possible to put all the highlighted points at a global time scrub bar. The possible disadvantage is that a user cannot identify to which media item the choice belongs to, which is a piece of information that may or may not be important depending on the XIMPEL presentation being played.

6.4 Between subject time scrubbing in XIMPEL

Now that I have outlined which possible design dilemma's by having choice within XIMPEL and the newly built parallel player introduces within a subject, let us look what interactive video or rather XIMPEL itself introduces. What if we do not want to time scrub within a subject but between subjects? From a user experience point of view this is entirely possible because users may experience the interactive video as a whole, and perhaps they would want to skip certain scenes, or maybe skip halfway.
It is important to understand that this is a wicked problem, meaning: “a problem with multiple plausible solutions as well as multiple subjective interpretations of such solutions.” [68] The paper of Carl Magnus Olsson, Staffan Björk and Steve Dahlskog goes into detail about what wicked problems are, how it relates to design (and game-design in particular) and how to deal with them [68]. For now, it suffices to understand what a wicked problem is. In our case devizing solutions really means making compromises between ease of use and having more information. I will present two different designs which emphasizes one or the other. In order to see which design would be better, user studies would need to be conducted for which I unfortunately lack the time.
We could show more information by having a screen that displays the full interaction graph of XIMPEL. The user would be able to see at exactly which point a choice would be made and would be able to scrub along the time graph (it is not a time line anymore) and land at the point where he or she wants to be. The issue with this is that this may, in some cases, neglect the idea of interactive video. By giving a user the possibility of seeing the interaction points, the author of a XIMPEL application would be forced to give away a part of the surprise. An example of such a graph is shown in figure F.16.
In normal time scrubbing this is not a problem. Consider a horror movie – where there are a lot of surprises. When a person scrubs further along the time line, he or she has no idea what to expect. This is because nothing is highlighted other than the time. With showing the full interaction graph it may not only be needed to show where the interaction points are, but also which interaction points.

Whether users prefer a full interaction graph with only nodes and edges going out of nodes or also see labeled possible interactions on the nodes would require user testing.
The simpler way of doing time scrubbing in XIMPEL is creating a scene skipping feature. With the current capabilities of XIMPEL this is already possible. It is even possible to show all possible options for the next scene as an overlay that the user could click or tap on. While it is a crude way of time scrubbing, it is an easier interface and perhaps therefore more usable. An example of this is done in the Zaanse Schans XIMPEL presentation[2].

6.5 Interaction of within and between subject time scrubbing

When a user scrubs within a subject, as soon as the user moves to scrub time between subjects it begs the question whether a XIMPEL application should remember the scrubbed state within the subject or whether it should not. If it should, then the concept of time becomes slightly different. For example, a user scrubs the 50th percentile of the fifth subject with the between subject scrubber, then if the user left the state of that subject in a certain way it needs to be recalculated what it means to scrub to the 50th percentile of it with respect to the offset of where the user left it. Imagine a user clicking on the middle of the upper arrow between *Menu De Bonte Hen* and *Tour of windmill De Bonte Hen* in figure F.16 (lets call this user action A). Then the user would be transported to the middle of the video that is associated to *Menu De Bonte Hen.* In this hypothetical scenario the user then suddenly clicks somewhere completely differently on an edge in the time graph (user action B). The user then performs user action A again after a couple of second – perhaps of boredom. After these actions, should a XIMPEL presentation remember that it already was playing from the 50th percentile, and therefore conclude that since the user clicked on the middle of the edge again replay the 50th percentile or add it as an offset? It depends on the intent of the author with their XIMPEL presentation, but both are options.
There are two other possibilities that make the interaction for between subject time scrubbing and within subject time scrubbing easier. The first possibility is alluded to in the previous paragraph, which is not to remember the within subject state. For example, when a user scrubs to the 50th percentile of the fifth subject, then a user would see the fifth subject playing at the 50th percentile point in time, which is always the same, namely: the 50th percentile point in time of the fifth subject. A second option could be to not intermix between subject time scrubbing and within subject time scrubbing, but having two layers. In this design a user would first need to choose to which subject they want to travel to and when that subject loads, only then is it possible to time scrub within that subject. The latter option is easier to implement since separated conceptual concerns translates to compartmentalized code (e.g. different classes and different files). One could imagine figure F.16 still being shown in the upper right corner when a button is clicked. When the user clicks on a node, then the subject is loaded and only then is it possible to scrub within the subject using time line sliders as seen earlier in the section *Time scrubbing: the difficulties introduced by the parallel player.*

6.6 Conclusion

Implementing time scrubbing within hypermedia frameworks generates more questions and almost no answers. This topic has not been explicitly studied and because of that this chapter has been written. Specifically three areas with design related questions have been identified:

- Within subject time scrubbing.
- Between subject time scrubbing.
- The interaction of within subject time scrubbing and between subject time scrubbing.

[2]see `http://classic.ximpelapps.nl/zaanseschans_html5`

Another design topic written about – albeit in a less structured way – has to do with the overlay in XIMPEL. Does the possibility of choice need to show up in a time scrub bar? If not, then we are done. If it does, then there are three areas to consider:

- Presenting possible future overlays within a single media item (related to within subject time scrubbing).
- Presenting possible future overlays within multiple media items and different media types (related to within subject time scrubbing).
- Presenting possible future overlays while time scrubbing between subjects (related to between subject time scrubbing).

Possible designs have been proposed and have been inspired from previous research. Presenting possible overlays in the future could be implemented in a similar way the video player of CS50 or [37] do. Except instead of showcasing summarized labels of what the video is about at that current point in time, the labels will be about what overlays they are and what for effect they may have on the user. Inspiration for within subject time scrubbing mostly came from the study that had multiple time lines in order to fine-tune time scrubbing [81]. Knowing that it sometimes is useful to have multiple time scrub bars gave rise to the idea to split time scrubbing up into a global time scrub bar and local scrub bars (one per media item). The design approaches for between subject scrubbing have been inspired by the research of [23]. Abstracting subjects away as a node and conceiving that it could be possible to time scrub a whole graph has been sparked by the idea that it is possible to manipulate time through multiple trees, such as the ones presented in [23].
A limitation of this exploration is that this means that a lot of research did not directly go into the design of time-scrubbing within XIMPEL. The biggest example is manipulating objects within a video [**67**, 93, 39, 27, 53]. Another example is research done on time-scrubbing systems within the context of MOOCs. Ideas of user statistics [52], word clouds [101] did not make it. What did make it was showing points of interests through overlays [101].
Another glaring limitation is that there is the implicit assumption that all media items have the same time! This limitation has been found out too late in order to change the analysis, and it would make matters most likely even more complicated. It would be odd to have a global time scrub bar track everything in a relative fashion if one media item only takes 5 seconds of playback and another media item 500 seconds of playback. The full length of the global time scrub bar has to be pegged to the longest playing media item perhaps, but the implications of doing so would not be clear, especially not if some media items have infinite time.
Future work could be done on doing user studies or by choosing a subset of all the possible designs outlined in this chapter and directly implementing it. In the first case, user studies would need to indicate to what extent users want to be able to have time scrubbing abilities. And high-fidelity mockups are possible by using XIMPEL and local scrubbing. A target group of interest to test these mockups with would be people who like to go to museums, since XIMPEL as a framework seems to be rising in popularity. Another one would be students since they seem to be one of the most tech-savvy mainstream general groups, and they are relatively easy to recruit.
Regarding implementation, the idea of what a media item is changes. Not only is a media item able to track time but it is also able to scrub time. Moreover, in some design ideas this information needs to be passed to something controlling the subject and when it changes to another. So the XIMPEL player in XIMPEL JS or the `Subject` component in XIMPEL React need time-tracking and time-scrubbing capabilities. These code entities need information from the media items, since it needs to know what their internal playback position is.
One design and implementation approach is to stay close to the XIMPEL design philosophy, which is to keep things simple and not to overthink it. At first I thought this meant: there is a global time scrub bar which determine the time scrub bars of media items (this is exactly F.8), no possible future overlays are shown at the time scrub bar and overlays could already be used to skip individual scenes. However, because of the limitation regarding the global time scrub bar a more pragmatic approach is to only allow for local time scrub bars. As future work this will be programmed into XIMPEL JS and XIMPEL React as the attribute `timescrubbing="true"` (the default is false).

Another form of future work would be to do a conceptual analysis on a partial global time scrub bar. This would be a global time scrub bar that is linked to media items that a XIMPEL author believes it should be linked with, and the unlinked media items have a local scrub bar (not connected to the global one) or none at all. Showcasing the implications of this may yield to fruitful research efforts regarding time scrubbing and hypermedia.
So, what needs the ability to go foreward and backward in time? A media item? A subject? A whole graph? These are the three levels that have been discussed in this exploration. Despite that, there are many stones left unturned and many corners regarding this research area uncovered. Indeed, it might be clear: in this problem there is no rest for the wicked.

7

Exploration 6: extending the YouTube media type for media item subject switch survival

Note: this whole exploration is written as a story. Why? Because it is the final exploration of course[1]. Ending it on a more festive note seems appropriate.

Research Questions and Contribution

What possibilities do 3rd party developers have in order to extend XIMPEL? Before this exploration 3rd party developers did not have the ability to dynamically alter the playlist in-memory configuration. This is unfortunate, because to implement innovative functionality at the media type level such a possibility needs to exist. This exploration shows one example of an innovative feature that could be created with it: the media item subject switch survival (MISSS) feature. By having this ability as a possibility for each media type, and by extension for each media item, dedicated 3rd party XIMPEL developers are able to design a wider array of educational experiences. This is also its contribution to the first research question regarding education.
There is a contribution to the second research question. The second research question is: (2) what are the (technical and design) implications of parallel media playback? Without parallel media playback this exploration would simply not have been possible for the MISSS feature would not have meant anything relevant. If one media item survives the next subject switch and goes to a next subject, then two media items would need to play. This implies that parallel media playback would need to exist since more than one media item playing is the idea behind parallel media playback. Therefore, a media item surviving a subject switch cannot play itself since parallel playback is not supported.
There is no contribution towards the third research question.

As I was playing with the XIMPEL playlist I wanted to have some background music with my videos via YouTube. I then noticed that I wanted the background music to continue, but it could not. It could not continue because I was changing subjects. So I wanted for a media item to survive a subject switch. I call this a MISSS, a media item subject switch survivor. But how could I do this? Hmm... I wonder how. Could I signal to HTML5 that I want to keep a certain media item playing? All it needs to do is not to detach.

[1] In terms of chapters, not counting appendix A, or the chronological final exploration which is arguably exploration 3 or exploration 7.

After fiddling for hours I figured out that one way to do this is to put the model of the `mediatype` into the `playlistModel` for a second time in the `subjectModel`, just before the `subjectModel` came for which I wanted it to stop.
This was the code in `YouTube.js`:

```
var sqModel = new ximpel.SequenceModel();
sqModel.add(this.player.subjectModels["lesson1"].sequenceModel.list[0].list[0].list
    [0]);
this.player.subjectModels["lesson2"].sequenceModel.list[0].add(sqModel);
```

I also had to modify the stop function in the YouTube media type.

```
  if( this.player.currentSubjectModel.subjectId !== "lesson3") {
    this.state = this.STATE_PLAYING;
    console.log(this.mediaModel);
    this.onEnd(this.player.sequencePlayer.mediaPlayer.handlePlaybackEnd.bind(this));
    return;
  }
```

So how would I solve this? I wanted to let the implementation of MISSS rest on the shoulders of the media type developer, which is a pretty big ask since they develop plugin-like code and do not want to change the core of XIMPEL. What was missing was internal knowledge about the `mediaModel` in-memory configuration object for extending XIMPEL. Having access to that means that a media type developer is able to add and remove this `mediaModel` to or from the in-memory configuration object on the fly. The media type developer could now dynamically alter the completem XIMPEL playlist.
In the `constructMediaItems` method I added a fifth argument, the entire `mediaModel`. Since the media type developer has access to the complete player object he or she could traverse the whole in-memory configuration (i.e. the playlist) recursively and eventually add the media model wherever their heart desires. The code on how to extend the media type is in appendix F and implemented in XIMPEL JS.
I tested if nested parallel players would work and they do. The reason is because the first `traverse` function searches for the media model that is needed. Therefore, it will always find what it needs (the media model) and is at that point unconcerned with how nested this media model is. Furthermore, the `insertModel` function will add the model to the nearest parallel model (e.g. `"lesson3"`) of the subject model just before the final subject model where it needs to stop (e.g. `"lesson4"`). In this case the method of adding the `mediaModel` is a bit crude, since there is no concept of nesting. However, this is not needed, since all that is required is for the `mediaModel` to signal the stop event to the XIMPEL player after the subject switch from where it is dynamically inserted. For example, suppose a media item needs to stop at a subject called *lesson 4* and before *lesson 4* there is a preceding subject called *lesson 3*. By dynamically inserting the `mediaModel` into *lesson 3*, the stop event will be triggered after the subject switch from *lesson 3* to *lesson 4*, meaning the media item will be removed when the XIMPEL player switches to *lesson 4*.
Finally, after writing all my code I could listen to my background music. I had a real MISSS since YouTube videos could survive a subject change. I then thought about how to do this for non-parallel (i.e. sequential) media and realized that it would be rather silly. A single playing media item does not need to survive a changing subject since it is the only media playing! This means that this feature is fundamentally made possible by the parallel player!
The only question that remains is: should this become a core feature of XIMPEL? I do not think so. XIMPEL should stay true to its name and origin and be simple. I consider this feature to be intermediate to advanced. Consider this, for more than 10 years the framework relied on the idea that when a subject changes, all the playing media items change as well. It may confuse novice people – new to XML – that it is also possible to let media items survive a subject change. It redefines the idea

of what a subject is and the definition may be less clear, which is fine if a media type provides that as an advanced option for its instantiated media items, but not the framework itself. It should stay XIMPEL, I mean simple.
Since we have one media type programmed in this manner it is possible to see how many people will catch onto this idea in the workshops where XIMPEL is presented and demonstrated. If in those workshops, people would like to have this as a core feature, then it may be a good idea to change the framework and add it.
One way of changing the core framework is to not change it at all, but to increase the power of the extensibility of the framework. Now, it is only possible to extend the framework through media types. This is wonderful but has the limitation that a developer needs to sometimes duplicate code if he or she wants to program the same functionality in other media types. Regarding a MISSS, what we could also do is to create a JavaScript file that share utility functions – like the ones I wrote for this exploration – and expose these functions for all media types.
Two types of future work could still be considered. (1) It currently is available as an attribute. Perhaps there is a more intuitive syntactic form so that the MISSS follows more intuitively by reading the code. (2) Currently, it is only possible to let a media item survive a subject switch for only one subject. Maybe there are multiple subjects at which a media item should stop. If this feature gets requested in upcoming XIMPEL workshops, then such a change should be made.
Addendum (months later). This feature has been added to the core of XIMPEL React, since I wanted background music there as well. The reason it is added to the core of XIMPEL React and not via a plugin functionality like in XIMPEL JS is because it simply was a lot easier to add it in XIMPEL React as a core feature. Just as it was easier for XIMPEL JS to implement it as part of a plugin extension. And while XIMPEL needs to stay simple, there is something to be said for implementing a feature that has been explicitly described as part of the Amsterdam Hypermedia Model [42]. This does mean that the architecture of XIMPEL React has a disadvantage, or perhaps not since it shows XIMPEL React is less hackable. XIMPEL React would have been more hackable if it would have seemed (or been) easier to implement the MISSS feature in its media types.
Addendum 2 (two months later after the first addendum). In the creation of a simple music sampler in XIMPEL, one bug has been found regarding removing the media items when they were stopped. The bug was that while the YouTube media type was able to dynamically add in-memory configuration objects of its own model to the playlist, it was not able to remove them. A method has been created to make this removal possible. Moreover, when models are dynamically added, a property called `isGlobal` is set to `true`. This bug has not been found in XIMPEL React.

8 Discussion

The questions from the beginning were: what is the relationship between XIMPEL and education? And how does XIMPEL need to be improved for it to better serve education? These questions have been treated in a slightly more general sense by applying the question to hypermedia. Even though XIMPEL is not a strict hypermedia framework, in this thesis it has mostly been treated as such. The philosophy of developing XIMPEL is one where pragmatism and interactivity is favored over the ideals of hypermedia. Therefore, it is quite hard to pinpoint what XIMPEL really is. Not to forget, it is also a framework to gain a poor man's immersion for games and one that is between storytelling and gameplay [30]. Perhaps it is best described as a hypermedia framework for non-linear storytelling and simple gameplay possibilities. With that said, it is clear it is mostly inspired on the ideals that hypermedia had in the beginning phases of the world wide web.

Research Questions and Contribution

The discussion summarizes the explorations and presents future research. For this reason all research questions are relevant. To remind the reader, all explorations have been relevant to the first research question, which is: (1) how does XIMPEL need to be extended to contribute to online education? For the second research question, only the fourth exploration did not provide any contribution. The second research question is: (2) what are the (technical and design) implications of parallel media playback? For the third research question, the first exploration (to some extent), second exploration and sixth exploration did not provide any contribution towards the third research question, which was: (3) what areas of research could XIMPEL benefit from, and how?
The contribution of the discussion in particular is focusing on the future. The following topics discussed in this chapter contribute to the first research question: (1) ways to extend XIMPEL in its programmatic expression so that web developers have more control; (2) research on media synchronization; (3) including media studies academics in the conversation about hypermedia; (4) creating educational content and finding users; (5) future work regarding (a) gamification, (b) animation, (c) augmented reality and (d) virtual reality; (5) going cross-platform on tablet and mobile.
The following topics discussed in this chapter contribute to the second research question: (1) time scrubbing from a mathematical perspective; (2) optional media playback.
The following topics discussed in this chapter contribute to the third research question: (1) XIMPEL and the microservice architecture.

It is with pragmatism that the questions about education were implicitly answered. The relationship between XIMPEL and education is the creation of online education. It has been possible to recreate a lot of psychology massive online open courses (MOOCs) with XIMPEL since all they need are: video

lectures, quizzes, scoring quizzes and text. The Zaanse Schans video shows one way of how this could be implemented. However, it would not have been possible to recreate a computer science course. This is why the terminal extension has been built in exploration 1, to explore and prototype how it should be done. Prototyping computer science courses with a hypermedia framework is a new way to look at computer science education. Another implication with parallel media play is that XIMPEL looks like a LaTeX for PowerPoint, meaning that it can be used for general presentations. I personally tested this to teach my mother about file systems, it also had quizzes to test whether she understood me. One could create this with PowerPoint and now it is also possible to with XIMPEL.
However, only having a terminal window open without anything else is not really informative. The second question on how XIMPEL needed to be improved for educational purposes answered itself in two ways in exploration 2. First of all, the hypermedia ideal should be relaxed and be used as inspiration, as it already was by the XIMPEL developers. This allows for the development of, for example, a `<terminal>` tag as shown in exploration 1. Second, XIMPEL needed the ability to play parallel media. This feature has also been identified as future work by Stefan Bruins, who ported XIMPEL from ActionScript to JavaScript.
Now with these two explorations both questions were put on the background regarding this thesis. One could always research deeper into a question but the theme of this thesis was exploring XIMPEL (and hypermedia), not answering research questions as deeply as possible. This also was needed since research on hypermedia seems to be a bit forgotten. The decision to look at another question rose to the surface in the form of recreating XIMPEL in React in exploration 3. Switching to different questions reveals more about the nature of hypermedia, which could lead to more potential avenues of new research, which might mean that the research topic will be looked after a bit more. In short: asking more questions is its own contribution.
Nevertheless, my supervisor and second reader asked me to reflect on the biggest research question regarding education and put them in blue reflection boxes. One of the reasons to do this was because I was still contributing to education, despite that the questions were operating in the background. Another reason was to make me reflect. This reflection never stops. For example, just before handing this thesis in, I realized that XIMPEL is a very good framework for demonstrating forms of procedural rhetoric [6] and it might be an avant garde framework for developing applications that train tacit knowledge (e.g. learning the process of user experience design or any other fuzzy process related skill). Courses like human-computer interaction will benefit from XIMPEL showcasing all kinds of scenarios regarding performing a week long design sprint, condensed in a couple of hours.
The question that came up during exploration 1 and 2 is: since the XIMPEL playlist almost looks like a return statement from a render method in ReactJS, would it mean that ReactJS would help the development of XIMPEL? The idea here is that it may help since XIMPEL has something in common with the core philosophy of React: every tag has its own rules. Moreover, would the React ecosystem and best practices help the development of XIMPEL?
The answer is: React does help. It also does not. It helps in the sense that mapping components to XML tags makes the development of XIMPEL easier to reason about, mostly because this mapping is explicit in the architecture of the React version of XIMPEL. Another advantage is that the XML attributes are already parsed as props per React component. On another note, the ability to write React Native and have cross-browser compatibility without thinking too much about it is exciting. However, two areas of difficulty in the beginning for a new React developer are the fine-grained understanding of lifecycle methods and best coding practices in React. Especially if a developer does not have knowledge on both of these topics, it will take some time to become productive. The biggest disadvantage is when components become too complex and they need to be compartmentalized. This has happened a little bit by creating specific media type components (e.g. `Image`) as a render prop for a `MediaType` component that had time tracking abilities which all specific media type components needed. All media types have common tasks (e.g. time tracking), but this is not expressed in the XML playlist, which breaks the one component mapped to one XML tag abstraction a bit. Another view could be that this is not a disadvantage but a diagnosis that the XML specification of XIMPEL is not explicit enough.
The answer to the React ecosystem question is a definitive yes since the ecosystem allowed for ES6

and using an XML parser in the form of a library. Writing ES6 helps because combined with React it forces one to write quality code, which is the intention of the React library developers[1]. Having an XML parser that does what one wants it to do means it saves writing code. Furthermore, the technology is relevant today which means that motivated students could learn a relevant technology[2]. It ties back to the first question as well: developing with XIMPEL helps education, and studying the source code of the framework is educational as well. Credits have to be given to Anton Eliëns for this idea, since he has nudged students into studying the ActionScript code of XIMPEL when I took his class – and I did in fact study it. To conclude for exploration 3: while React on itself would not have been enough of an improvement, combined with its ecosystem it (perhaps) is.

Does this mean that all XIMPEL development need to happen in React? This depends very much on the team of the XIMPEL developers and mostly their personal goals. If the XIMPEL developers are neutral, then yes since it has been argued that the XML specification of XIMPEL lends itself for ReactJS. If they are not for some non-technical reason, then perhaps not. It could also be that two versions co-exist of XIMPEL which is fine. XIMPEL has been mostly developed in an explorative user-centric way after all [5].

There is no connection from exploration 3 to exploration 4. As stated before in this thesis I was daydreaming. It furthermore is the topic which is removed from hypermedia as much as possible. It is also the first exploration where the conceptualization of ideas has played a much more important role than pure implementation. The question was: how to measure frustration within the users of XIMPEL? A similar question has been asked for engagement, albeit a lot less emphasized.

In this exploration the minimal conditions needed to capture the data has been presented. One of these minimal conditions is that facial expressions are captured since mouse data is not enough. Furthermore, research in capturing frustration via facial expressions seemed confusing at best, which is why capturing anger and sadness seem to be a better way to capture frustration. Not all instances of anger and sadness have to do with frustration, but anger and sadness are possible consequences of frustration [36, 102, 94, 89], are more universal [69] and also more well-studied.

Regarding the other metrics, some hypotheses have been formulated on how to detect frustration there. Research on it has been scarce so it has been a mixture of literature and my own thoughts on how to detect it. Future work of simply training models have been eluded to.

Exploration 5 and 6 have been natural consequences of the result in exploration 2: the ability to play multiple media items. In exploration 5 it has been explored what the consequences are for time scrubbing. Not all design issues with time scrubbing come from the parallel media player, the biggest design issue starts to appear when one wants to time scrub between subjects. The parallel media player does interact with the other design issues and makes them more complicated by default.

Exploration 6 asked the question on how to implement a media type that survives a subject switch (MISSS). Normally, in XIMPEL it is impossible for a media item to survive a subject switch. Even if the media would be played in the new subject as well it would suffer from refreshing issues and rendering issues. It furthermore would forget its state, which for a YouTube media type might be simple enough to save, but for a terminal media type this might not be the case. This feature is called media item subject switch survival. This question could not have been asked if XIMPEL did not play multiple media types within a subject.

And now we are here. To summarize these are the questions that have been asked:

- what is the relationship between XIMPEL and education?
- How does XIMPEL need to be improved for it to better serve education?
- Does React and its ecosystem help for developing XIMPEL? If so, how?
- How does one measure and classify frustration and engagement in hypermedia frameworks?
- How should a time scrubbing feature be designed in XIMPEL?

[1]A good example of that this really is their intention is seen in their update on async rendering. See: `https://reactjs.org/blog/2018/03/27/update-on-async-rendering.html`

[2]This used to be the case for when XIMPEL was written in ActionScript and it eventually faded with the rise of HTML5. This may happen again as well, especially since Adobe fought hard to keep it alive and still failed. However, there is hope it may not since Facebook is behind React and has perhaps more resources than Adobe.

- How does a media type developer implement a media item surviving a subject switch?

In the style of the `whatis` command, here are one line answers:

- The quick yet effective creation of MOOCs through a simple declarative domain specific language (via XML) that anyone can learn.
- It needs the ability to play parallel media and needs to have built-in applications as a tag.
- Yes, it helps because of useful libraries, better development practices, cross-browser compatibility and the possibility to create mobile applications with a XIMPEL playlist.
- By capturing data from the mouse, what the user is viewing now and his or her facial expression.
- It depends, there are a variety of ways and it is not clear which one has a better user experience.
- By interjecting the model of the media item in the subject which has the `leadsTo` of the subject on which you want the media type to stop.

8.1 Future Work

XIMPEL is a frontend framework. It used to have no connection to any backend. In some of the explorations (1 and 4) it needed a connection with a backend. In both cases the solution has been to create a NodeJS server and use websockets or XHR to connect with it. Any developer who has a background in NodeJS or any other server-side micro framework will find this relatively easy to do. Therefore, future work could go into specifically developing XIMPEL so that it would fit with a microservice architecture. It could also be researched what the drawbacks and advantages are. Independently, in Norway some backend programming has been done in Flask, which at the very least hints at the idea that it is an intuitive architecture to have.

A more fundamental area of research and development is to make a hypermedia or hypermedia-like framework that focuses on web developers. HTML5 has made hypermedia applications more approachable through the use of HTML, CSS and JavaScript. More specifically, the video and audio HTML tags and their respective JavaScript APIs make this possible. What has not been made easy is the concept of overlays, which is what forms the hyper part in *hyper*media. So a library that focuses on this might help. There are already solutions out there [65, 7]. However, they do not focus on hypermedia. They focus on more specific use cases.

Other fundamental future research is a literature review on media synchronization. The AHM and SMIL made this a huge topic. XIMPEL does not make this a huge topic at all. However, the justification regarding not making that a huge topic seems to be lacking. Hence, a literature review may help to elucidate what justifications there are for either perspective, and perhaps there are more perspectives to consider.

Partially fundamental and partially not, the implementation of exploration 4 (classifying frustration) and 5 (implementing time scrubbing) are future work. Both are bachelor or, depending on how far one goes, master theses on their own. Other than the further implementation of exploration 4, there is more future work regarding that area (see the future work section of exploration 4). Exploration 5 has its own challenge in that it is mostly a requirements issue and design issue, which falls in the domain of human-computer interaction.

Other extension features that are more in the vein of applied research is the ability to automatically switch subjects without the user driving it. See exploration 7 for that claim in appendix A [3]. It is possible to switch from a subject to another by branching with a `leadsTo` tag. The limitation of this tag is that it only switches once automatically. If it would switch more in an automatic fashion, then media items surviving subjects could be manipulated better, by being deleted one by one (if needed) in automatic fashion.

The first paragraph about fundamental future research was about focusing on web developers. A similar brand of future work would be to recreate XIMPEL through the use of web components. This would not mean that XIMPEL would be exactly the same. The benefit that one would attempt to have using this approach is the ability to create a playlist or several playlists within an HTML file.

[3]While a relevant exploration regarding the research question of education, it clearly was outside of the project scope.

This would be useful for web developers who want the full ability of HTML, JavaScript and CSS but also want the functionality that XIMPEL tags provide. Another possible way to achieve this is reuse the media type components from XIMPEL React and develop a ReactJS application, which may very well be a much quicker way to achieve this since the media type components already exist – they may need to be adapted a little.

Moving away from fundamental research and the web, a collaboration between media studies researchers and computer scientists could be made. The amount of knowledge about media from a more conceptual point of view taught to computer science students is close to zero. Because of this, people with a background in computer science know the technicalities of media but not really what it is. It is likely that this is also the case for hypermedia. Interdisciplinary research on the boundaries between (hyper)media and applications could be done. Or maybe it is not research but simply cross-over lectures at universities. For example, teaching XIMPEL to media students and having a discussion with them about (hyper)media.

From a more technical point of view the idea of MISSS challenges the idea of subjects. There are different ideas to think about hypermedia, for example, the AHM thought about media items needed to be put in different tracks, akin to musical instrument tracks in a digital audio workstation such as GarageBand, Cubase or Logic Pro. A question could be: how can one fully re-imagine a hypermedia framework from the ground up?

Regarding education, it is simple what needs to be done for XIMPEL. Content is king. Right now XIMPEL has no king. Therefore, XIMPEL needs content, educational content. This will show how usable XIMPEL is right now to current XIMPEL authors. Strengths and weaknesses will surface. It is clear that XIMPEL will occupy a niche in the prototyping space. The real question is: could it also occupy a space in production, and if so, is that already the case or does it need to be improved?

Another avenue of future research regarding education is to investigate the cross section between hypermedia and website annotation. The web is, for some reason, almost not annotated! Why do users not remix the web more? It should be possible. The value that XIMPEL would bring is to have a simple language that would allow for annotation. Creating a list of iframes with additional media items from other media types is simple.

The thesis itself showed a lot of implications regarding the parallel player. More future studies could be done on focusing what the implications of sequential media playback, parallel media playback and parallel media playback including applications are. One could say that this thesis is a contribution to that, but there is more to explore. Especially mathematically, for example, what if all time scrubbers knew the playback position of all the other objects it has a relationship with, how many connections would there be? Take for example, a scrub time graph, a global subject scrub time line, and 5 media items within that subject having scrub bars. 7 time scrubbers in total being connected to each other is $n(n-1)/2$ with $n = 7$ so it is 21. Perhaps the mathematical implications are already solved problems in mathematics, but they are not yet uncovered within the realm of hypermedia.

Another area of future research regarding media playback is to introduce a third type of media playback, which is the optional media item. This feature exists in SMIL [50]. Though before implementing this an analysis needs to be done. The question would be: to what extent could optional media playback be modelled with a combination of MISSS, subjects and conditional subject switches via a score? It may be the case that it already is possible since the conditionality can be modelled via scores and different subjects. Despite the fact that it may be possible, another related question is if it is possible to model the same level of fine-grained control since SMIL has it on a media item basis and with XIMPEL it is one level up: on a subject level basis.

XIMPEL is an amazing framework to do gamification studies with regarding score. At the time of writing my thesis I did not realize this, otherwise I would have done it. Gamification research seems to focus too much on points[4]. And in my experience non-game designers who do not play digital games believe that simply showing points is enough. With XIMPEL the claim of showing points being motivational in itself can easily be tested. They can be tested against: no points and points with

[4]A claim I remember from my game studies master, unfortunately I do not know any articles that back this claim up.

meaning. What are points with meaning in XIMPEL? Simple, points that are related to conditional subject switches. When points are connected to subject switches they inherently mean something, because points are part of a mechanism of determining what content a user gets to see. This study could be done in collaboration with psychologists and would have three testable conditions: no points, showing points, connecting points to conditionals (which is basically the equivalent of an if-statement).
Gamification and animation go hand in hand. SMIL has animation capabilities. It needs to be researched whether XIMPEL could use that specific language as a media plugin, or that SVG is a better format. Adding animation capabilities to XIMPEL will create a richer experience to users. It furthermore may be needed for hypermedia presentations or applications used in production, since some of these may need a certain type of polish that only animation can provide.
Other than animation, perhaps something closely related that I did not look at are augmented reality, mixed reality and virtual reality. It could be researched how they intersect with hypermedia and perhaps added to the framework. A simple example is: having a smartphone that is aware of your location, and then overlaying some piece of information next to the designated landmark of said location on the smartphone. This research could go hand in hand with XIMPEL being ported for binary mobile applications.
Finally, this section could be a thesis on its own. A lot of topics have been touched in this thesis which all was about exploring XIMPEL and exploring the nature of hypermedia. The final future work recommendation is the one that may enjoy a high priority since it will elevate XIMPEL to another device type: mobile. Creating a version in which it is possible to create mobile phone applications with XIMPEL seems the most interesting use case of using a hypermedia framework like XIMPEL, simply because it makes a certain collection of mobile phone applications a lot easier to make. Therefore, XIMPEL needs to be ported to React Native. An amazing example of this is: tablet questionnaire applications for psychology research (ok, 2 device types!). There is a real need for this. Take the Vrije Universiteit as an example: it employs its own iOS programmers to create such applications for psychologists and education researchers. An example of such an application is the SIVT iPad app (Sociale Informatie Verwerkings Test, English: Social Information Processing Test). It is a test for young children with an IQ between 50 to 85 to see how well they recognize certain social situations [80]. For example, they see a clip of a social situation and they ask a question about it. With XIMPEL it would not take a team of developers a year building it[5]. It would be a month.

[5]I have built the first version of the iPad app and know from my ex-colleagues that it did indeed take a year to build.

9 Postface

Doing this thesis project has been an emotional rollercoaster. During most of my study program I focused on study speed. I was on track finishing 9 years worth of study within 5 to 6 years of studying. Then my master thesis for computer science got rejected because of a bureaucratic technicality. I asked permission to my supervisor to be relieved of that technicality a year prior, which I was. However, a year later that permission was revoked and my thesis was not considered to be up to the standards of a computer science thesis. This sent me in a tailspin and I do not know if I became better because of it. I choose to think of it as my first serious exercise in Stoicism.
At the time, it put me in a unique situation. Do I appeal to the examination board? It was a thought in my mind, but I wanted to go out like an academic, not like a lawyer. I decided to throw away my old game studies thesis about gender (it was an okay thesis but not stellar) and handed my computer science thesis in as my game studies thesis at the University of Amsterdam. It was the first time that I decided to focus on quality. The consequence had a good effect and a terrible one. The good: I graduated cum laude because of it and I could truly feel that I belong somewhere in the educational game creation space. The bad: studying has taken me 8 years. I am now 29 years old and I feel behind my peers.
It is this focus on quality that allowed me to be daring and as truthful as possible with my final thesis, or with this postface for that matter. It furthermore meant daring to say no to the conventions of academia. It is about being truthful about the process of science. Or being truthful about why something is not science, but why it still would be relevant to science and perhaps as important. In my interpretation: academia should be concerned about knowledge that could be considered objectively true or at least very reliable and helpful in the broadest sense possible. Being truthful about the process of science is part of that, as is creating tools that will help us to uncover more truths. Therefore, I believe a startup like Codecademy to be academic, because it teaches more people to program, which may help in gathering new knowledge. Another example is Mechanical Turk from Amazon. Improving the process of finding participants is a major innovation in the field of social sciences. Why has this been invented by Amazon? Why not by a university? To me it seems the incentives are wrong.
So I am daring according to myself. Yay! But a more interesting question might be: how do I evaluate my own work? At first, when I read through my own thesis it seems very confusing. This is partially because I secretly tried to do three theses. One was about extending XIMPEL (with emphasis on XIMPEL for education), one about porting XIMPEL (and evaluate whether such a port is useful) and one about classifying frustration. It furthermore seems confusing because when I read the thesis, I find that I hold conventional academic standards in my mind. What specifically is my research question? I seem to switch a bit. In this case I do not believe it matters too much because the findings uncovered are interesting in its own regard. But it does give a confusing feel.
Specifically, for every exploration I have the following to mention: my first two explorations demonstrated to me that having the option for extensibility through media types is a necessity for modern hypermedia since it allows hypermedia to integrate into a framework that embraces hypermedia, or subsumes it. It does not matter as long as it is useful. What I believe to be my most important contribution regarding the first two explorations is: hypermedia will die unless it can integrate into

something larger[1]. The document model of the web already seems non-existent, it is all about web applications. The biggest factor of this paradigm shift is the focus on the development of HTML5 and JavaScript. HTML5 was probably a direct contributor regarding why SMIL was not further developed. So it makes sense that for hypermedia to survive, it must be possible to create fully fledged hypermedia applications, not merely hypermedia presentations. One contribution made in this space is every hypermedia framework that has been ported to HTML5/CSS/JS since the output of such a framework is a mix of HTML5 and CSS, which could be further dynamically manipulated through JavaScript. This would not be possible with Flash and ActionScript.

Exploration 3 also showed something interesting: it shows that hypermedia on mobile may be a worthwhile pursuit. Then why did I port it to React and not React Native? Because only by writing it (after programming) did I realize it! I spent hours upon hours of programming only to come to such a seemingly trivial realization. I feel like the inventor of the paper clip, it is trivial once you know it but it takes a long time to get there. With that said, in order to reap the benefits of such interoperability, the web version (i.e. React) needed to be made as well.

Exploration 4 and 5 is where I believe more conventional academic research lies. This means that it is probably possible to get funding for it and it will teach humanity something truly new and unique in a more conventional academic sense. Pursuing the themes in exploration 4 will teach us something about the universality of frustration and how it relates to user experience. Pursuing exploration 5 will teach us a lot about time scrubbing and user experience. It may also lead to a more substantial justification of why the time line model may not be the best foundation regarding media and time, which the creators of SMIL have been saying for a long time[2].

Moreover, I believe that I contributed a lot towards a reflection on the idea and ideals of hypermedia. I know computer scientists are not philosophers, but to stand still for a bit and take a step back may help the research agenda to produce more valuable research.

So to conclude my evaluation: I believe my thesis is valuable and relevant for hypermedia research, though read with conventional academic standards in mind it may be a confusing read. And I tried to do too much. Focusing on a narrow research question does have its benefits.

However, focusing on a narrow research question brings me to my second point. I sincerely hope that exploration 4 will serve as one example of how to incorporate interdisciplinary literature. It is too often that computer scientists make assumptions about human behavior and do not acknowledge that they make such an assumption. Examples of that are in my bachelor thesis of information science about Twitter and sentiment analysis. I hope to prevent this by showcasing that if a psychological claim is made, then at the very least, do a literature search on it to potentially falsify the assumptions.

To close off, writing this thesis was ultimately the hardest part, not the programming. Having not much of a structure and needing to invent my own structure was a challenge in which I believe I partially succeeded, but there is also a lot of room for improvement. What I liked the most was to do amazing things with LaTeX. I would like to thank users *moewe* (a non-capitalized username) and *Heiko Oberdiek* from `tex.stackexchange.com` for their helpful answers on creating amazing things with LaTeX. The specific questions that were needed to create this document are put into an appendix.

[1]Though interactive video is very much alive!

[2]But I still do not know why, I do not remember that they explained themselves in their papers about the Amsterdam Hypermedia Model or papers and documentation about SMIL. They just stated that they did not use it or did not find it useful.

A
Exploration 7: Hypermedia and Gaming

During the semi-final thesis meeting we were down to the final details on how to end this project. Ideas such as: create online documentation for what you programmed, put all source code online and similar points of feedback were discussed. At one point while we were talking about exploration 4 and I told them that it is possible to create a point and click adventure with XIMPEL. In essence, I claimed that it is possible to create subset of all possible games that could be created. Saying this claim outright to my thesis supervisors made me reflect more on that statement. A question came to light: is it possible to create games with XIMPEL?

While this question came to light it also became apparent that my thesis was finished. Changing the thesis to reflect this particular exploration would require some rewriting. For example, exploration 3 would not be the secret final exploration anymore, exploration 7 would be. Moreover, it might have been slightly confusing since this particular exploration is not in the intended scope of the original thesis. This is because during the semi-final thesis meeting it did not become apparent that I should do a whole new exploration. For this reason, this particular exploration has been added as an appendix.

The definition of what constitutes a digital game is a hairy one. In terms of this exploration I choose the definition of Joris Dormans who defined a usable definition on what games are as formal systems. The definition is as follows: "A game is a system in which players engage in artificial conflict, defined by rules, that results in a variable, quantifiable outcome affected by player effort and ability." [26] The first part of the definition relates mostly to willing humans who want to play and content creation (creating an artificial conflict). The second part of the definition, however, is mostly influenced by a programming language. This is because it is (1) a system defined by rules, (2) the rules of such a system results in a variable quantifiable outcome and (3) this outcome is determined by player effort and ability. These three key elements are all able to be experienced by a certain subset of all computer programs.

In order to find the answer on whether it is possible to create games with XIMPEL, the most important question to answer is: to what extent is it possible to use XIMPEL as an intuitive programming language? If it is possible to use XIMPEL as an intuitive programming language, then it is possible to create at least a subset of games with it.

In order to demonstrate with this I decided to construct the following playlists in XIMPEL: finite state machines, pushdown automata, a Turing Machine and a more intuitive mechanism that is equivalent to a calculator. There was another reason why I decided upon this approach. My knowledge about theoretical computer science is as good as any other computer science student who took one course in it. Therefore, in order to augment my knowledge, recreating a Turing Machine in XIMPEL would help me to understand how computability can be achieved (for similar projects see [104][1]).

[1] I saw some of these projects years ago. In particular I knew about: Minecraft, Magic The Gathering, The Game of Life made by Conway, HTML + CSS3, Super Mario World and PowerPoint. These projects served as inspiration to start this analysis for XIMPEL.

Research Questions and Contribution

This exploration relates to the research question: (1) how does XIMPEL need to be extended to contribute to online education? While this exploration is not an extension of XIMPEL, simply finding the imaginable possibilities will be an extension in the conceptual framework for all the students who will learn about XIMPEL. Certainly other developers of XIMPEL did not know to what extent XIMPEL has been Turing complete. They did not know what the implications are towards education if it would be Turing complete.
Games are able to have an educational use, especially when already combined with a hypermedia presentation this could greatly help education. More specifically, games are a specific form of rhetoric that seems unlikely to be achieved by text, audio and video alone. This form of rhetoric is called procedural rhetoric. “Procedural rhetoric is the practice of persuasion through processes in general and computational processes in particular” [6, 4] People see certain processes emerging in digital games and what these processes are and the inferences players draw from it may influence their opinion one way or the other.
Research areas that XIMPEL may benefit from are: theoretical computer science and game studies.

A.0.1 The computationality of XIMPEL

Creating a finite state machine in XIMPEL is trivial. Every `subject` is a state. However, there is no explicit input field. A small extension to XIMPEL has been added, which displays what previous states the user clicked through. The user clicks on an overlay to switch to a different state and it is the content of the overlays that are shown on the screen. It is the user interaction that allows XIMPEL to make computation as a finite state machine possible. This construction is shown in the following video[2].
To create pushdown automata a stack needs to be created. XIMPEL is only able to track scores. Scores are individual key-value pairs in which the values are always represented as numbers. One way to create a stack with a number is to represent the number in binary. Pushing to the stack could be simulated by multiplying the score by 2 or 2 + 1. Popping a value of the stack could be simulated by dividing a number by 2. These arithmetic operations are possible in XIMPEL with one caveat. These arithmetic operations always need to floor the resulting answer. One method in XIMPEL needed to be lightly rewritten in order to make this possible (the modified code is saved in the following repository [85]). Like the finite state machine this form of computation must be driven by user interaction. The constructed playlist is created for $L = 0, 1|0^n 1^n forn >= 0$. This construction is shown in the following video[3].
A Turing Machine is able to be simulated by two stacks. The first stack could be the left side of the tape and the second stack the right side of the tape. In this particular construction the top of the second stack (right side of the tape) has been denoted as the head. The head is able to move left by popping the top of the first stack and pushing it on top of the second stack. The head is able to move right by popping a value of the second stack and pushing it on top of the first stack. However, this posed to be a problem. Popping was simulated by dividing a score by 2.
While this pops the value of a stack it does not allow for the value to be remembered so that it can potentially be pushed on top if the other stack. In this sense XIMPEL is not Turing complete. However, since all the other forms of computation has been driven by user interaction it could be interesting to see to what extent one needs to cheat in order to achieve Turing completeness. While XIMPEL does not have any good constructs to simulate pushing or popping, it is possible to put multiple overlays on a certain state. It turns out that since the language is small $(0, 1)$, only four overlays have to be shown in order to show all the options regarding popping one value of the stack

[2]http://www.youtube.com/watch?v=8BixNZulR24
[3]http://www.youtube.com/watch?v=-IV8jiufG_E

and pushing another value on the other stack. So the cheat is to use user interpretation in order to simulate proper popping of a stack. By doing this all the read, write and move operations could be supported. A construction of a Turing Machine is shown in the following video[4].
Before we move on, I want to give proper attribution to who invented this idea of making a Turing machine and how I learned about it. The idea of creating a Turing machine from 2 counters is from Marvin Minksy [64]. However, I learned this technique through Wikipedia and reading a couple of paragraphs (and then fiddling for a week to make it work) [79]. Finding when this information was written is hard to determine as are the authors. What is not hard to determine is that the information has been useful.
The biggest strength of XIMPEL (and other hypermedia frameworks such as SMIL) is that they are state based. This allows for an easy construction of a finite state machine provided it is driven by user interaction. The biggest issue regarding XIMPEL has been found by trying to construct pushdown automata and a Turing Machine. The biggest issue is memory. More specifically, it is almost impossible to allow one variable to know about the value of another variable. Could this be simulated?
The construction regarding the pushdown automata and the Turing Machine have been cumbersome. Simulating a stack by displaying the binary representation of a number will not be seen as straightforward by XIMPEL authors. What may be more intuitive for XIMPEL authors are counters. From this idea another question arose: is it possible to achieve Turing completeness with XIMPEL by using counters?
In the following construction it will be demonstrated that it is possible to simulate one variable knowing the value of another variable by using incrementers and decrementers. This answers the question of whether one variable is able to know about another variable. Since it is possible for one variable to know about another variable, it is also possible to subtract or add one variable value from another. From this a simple calculator has been made. The `leadsTo` tag has been needed in order to achieve this. And while it may be possible to demonstrate XIMPEL to be a language that is able to compute everything, the process became less intuitive. The issue is: XIMPEL presentation authors will likely understand the ideas of counters. However, they do not want a user to superfluously click tens of times in order to see something meaningful. Perhaps automatic `<leadsTo>`[5] chaining (or in XIMPEL React `<rule>` chaining) would help to make it a bit more intuitive and definitely more computational. A construction of the calculator is shown in the following video[6].

A.0.2 Conclusion

Since it has been demonstrated that, through user interaction, a Turing Machine is able to be made in an unintuitive way, it is possible to create games with XIMPEL. It is clear that some games are able to be made in an intuitive manner and others in a less intuitive manner. The question of what type of games are able to be created in an intuitive fashion is impossible to answer unfortunately. It seems already fairly unwieldy to create a point and click adventure. However, it is possible. One big limitation of this exploration is that Turing completeness is driven by user interaction.

A.0.3 Future Work

For the sake of completeness I decided to try to create a simple shooter[7] and recreate Flappy Bird[8]. While the shooter is limited, it works well when there is one enemy on screen, less well when there are multiple on screen (they do not detach themselves from the DOM when clicked). The Flappy Bird game should not be considered to be a recreation but a reinterpretation of the game. The Flappy Bird

[4]`http://www.youtube.com/watch?v=7NhRtKYOVzQ`

[5]In XIMPEL JS, leadsTo is as well a tag as an attribute

[6]`http://www.youtube.com/watch?v=xwzXrl3hyR4`

[7]See an example here: `https://youtu.be/smaaZpPGisA?t=2m23s` until 3:31. There is no adio commentary.

[8]See an example here: `https://youtu.be/BX1S_iaOIcQ?t=2m39s`, the commentary is in Dutch and not important. Credits go to PewDiePie for playing it

game is created by having a YouTube video as media, overlays on the exact same moment the bird is supposed to flap and an artificial mouse event with the keyboard (via JavaScript) to artificially click on the overlay. By doing this the game mechanics are the same, the user needs to tap the keyboard on exactly the same moment as the person in the video. An interesting side effect is that this seems to be a form of meta-gaming since the XIMPEL user needs to guess when the person on YouTube is going to tap the screen on his or her phone. Future work would be a deepened exploration in the attempt to recreate games with XIMPEL.

B
LaTeX Questions on tex.stackexchange.com

Can LaTeX remember from which page the user jumped when clicking on a reference?
`https://tex.stackexchange.com/questions/424779/can-latex-remember-from-which-page-the-user-jumped-when-clicking-on-a-reference`
BibLaTeX: generate which reference types you have in your .bib file for automatically generating bibliography headings?
`https://tex.stackexchange.com/questions/424984/biblatex-generate-which-reference-types-you-have-in-your-bib-file-for-automati`
BibLaTeX: automatically give one domain name its own category?
`https://tex.stackexchange.com/questions/425036/biblatex-automatically-give-one-domain-name-its-own-category`
How do I show programming keywords differently?
`https://tex.stackexchange.com/questions/416160/how-do-i-show-programming-keywords-differently`
Biblatex: How to automatically make in-text citations bold given a condition?
`https://tex.stackexchange.com/questions/429874/biblatex-how-to-automatically-make-in-text-citations-bold-given-a-condition`

C

Appendix Exploration 2: XIMPEL playlist that is possible with the ParallelPlayer

```
<ximpel>
<config>
    <enableControls>true</enableControls>
    <controlsDisplayMethod>overlay</controlsDisplayMethod>
</config>
<playlist>
    <subject id="lesson1">
            <sequence>

                <parallel>

                    <media>
                        <textblock text="Type in pwd and the terminal displays which
    directory you are working in." width="600px" height="800px" x="0px" y="0px" color="
    #0f0" fontsize='50px' fontcolor="#fff"/>
                    </media>

                    <media>
                        <terminal />
                    </media>

                    <media>
                        <video x="0px" y="400px" width="600px" height="900px">
                            <source file="ximpel/media_types/terminal_assets/pwd"
    extensions="mp4" types="video/mp4" />
                            <overlay leadsTo="lesson2" width="600px" height="75px" x="
    0px" y="500px" text="next lesson"/>
                        </video>
                    </media>

                </parallel>

            </sequence>
    </subject>

    <subject id="lesson2">
            <sequence>

                <parallel>

                    <sequence>
                        <textblock text="The command ls helps you to list the files
    that are in the current directory you are working in." width="600px" height="800px"
     x="0px" y="0px" color="#0f0" fontsize='50px' />
```

```
                    </sequence>

                    <sequence>
                        <terminal />
                    </sequence>

                    <sequence>
                        <video x="0px" y="400px" width="600px" height="900px">
                            <source file="ximpel/media_types/terminal_assets/ls"
    extensions="mp4" types="video/mp4" />
                            <overlay leadsTo="lesson3" width="600px" height="75px" x="
    0px" y="500px" text="next lesson"/>
                        </video>
                    </sequence>

                </parallel>

            </sequence>
    </subject>

    <subject id="lesson3">
        <sequence>
            <message text="You made it to the end of the course!" />
        </sequence>
    </subject>


</playlist>
</ximpel>
```

Playlist C.1: this playlist can now be played thanks to the ParallelPlayer.

D
Appendix Exploration 3 part 1: The first time I tried to port XIMPEL to React and failed

For porting XIMPEL to ReactJS I thought it had the following advantages:

- A lot of parsing logic could be done via ReactJS and Webpack by transforming the XIMPEL playlist to a declarative language that is completely compliant with JSX. So there is no need to create a parser.
- The virtual DOM would replace the in-memory configuration code that has been written for XIMPEL. So there is no need to write in-memory configuration code.
- Cross-browser support is suddenly managed by the maintainers of the ReactJS framework.
- By teaching XIMPEL to students, you have to teach about ReactJS to students who want to extend XIMPEL which introduces them to a lot of computer science concepts implicitly.

In short, the idea was to see if it is possible to make the ReactJS framework work for us. If this is possible, then we as XIMPEL developers have a free lunch! Who does not want a free lunch?
To validate these assumptions I made a very simple prototype of creating a custom React component in an XML file. The XML file would be read in by a React class that I programmed. Furthermore, that custom React component would later on then correspond to a media item (such as a video or image) in XIMPEL. What I quite quickly found is that I had to hack my way around the framework. The first assumption that I invalidated with this is that a lot of the virtual DOM would replace the in-memory configuration code. This has not been the case. While it is true that the XML-parser of Webpack loads the XML in a data structure it is an object. Therefore, it is unknown which element comes first. The advantage of the in-memory configuration that it is loaded in a tree structure. This ease of use, or lack of it, has its implications for writing a framework.
Another assumption that came with a lot of hairy problems was the idea that this would be useful for education since you could teach advanced students ReactJS. This is true. Unfortunately, I had to program my way around the framework so much that this would also have been true for mere beginners at XIMPEL. In order to explain this, I will have to explain the approach that I made and the code that I created.

D.1 Methodology and implementation

To test whether to see if XIMPEL would benefit from ReactJS I created a simple toy playlist. The idea was to create more and complex playlists as development would go along. If there would be a significant roadblock for any reason, then the conclusion would be that porting XIMPEL to ReactJS may not be a good idea.
In this toy playlist (see playlist D.1) the goal was to see if I could extract data from attributes and handle nesting. The XML tag names themselves are arbitrary and for that reason I prefer short names. The attributes `text` and `sup` both were meant for the display of text.

```
<ximpel>
  <hey text="You made it to the end of the course!" />
  <hey text="Hello World!" />
  <yolo sup="something">
    <hey text="yay!" />
  </yolo>
</ximpel>
```

Playlist D.1: A toy playlist

From an implementation standpoint (see Github), this playlist is loaded in as an attribute the `Playlist` component called `data`. The `Playlist` component takes all unique keys and iterates over the data via the `.map` array method. In this method the element name is taken and saved. The `.map` method returns a lookup of the key in `data` which is chained to a different `.map` as well because in this `.map` method one actual element is going to be rendered. In there it searches for a single child element (multiple childs has not been implemented). It returns the parent and child.
The biggest issues with this code – other than the one child limitation – is that in order to render the parent and child to `eval` functions need to be used. This is because the XML playlist needs to be interpreted as React code. First of all, `eval` is evil for security reasons, which is also outlined by Stefan Bruins who had to use it in order to port XIMPEL to JavaScript [8]. However, perhaps more problematic is that using `eval` also necessitates using `React.createElement` as opposed to the usual React syntax. This means that this code is fighting against the framework in order to push through what it wants.
From a programming standpoint it is fine, except for the requirement that ReactJS was supposed to help make development easier, not hinder it. Another issue that can be clearly seen in the code is the lack of an in-memory configuration object. A React element itself is an object. The element that needs to be rendered itself is referred to as `\$` and possible children are referred to via a key. This means that React element parents and React element children have no explicit hierarchy as XIMPEL does have with its in-memory configuration object.

D.2 Conclusion

As I stated at the beginning of this chapter: "the idea was to see if it is possible to make the ReactJS framework work for us. If this is possible, then we as XIMPEL developers have a free lunch! Who does not want a free lunch?" The answer is, everyone wants a free lunch but unfortunately porting XIMPEL to ReactJS is not a free lunch. It is possible but takes time and effort.
This time and effort is not worth it because ReactJS gives no benefit regarding the in-memory configuration object.

E

Exploration 3 part 2: assessing the benefits for porting XIMPEL to React (successfully porting it to React)

When I wrote the previous chapter I was finished with my exploration. However, by writing it up I was forced to take a close look at the source code. Because I needed to look up the source code I needed to look up the dependencies. And when I looked at the dependencies, I noticed that the XML parser of Webpack uses a library that I did not look at. Upon further inspection this dependency of the XML parser of Webpack showed that the XML parser was configurable! Unfortunately I did not see this before, because I started developing too soon while not carefully reading the small documentation.

E.1 Webpack XML parser setup

There were two modifications that allowed for a much more successful exploration compared to the first time. First, I modified the parser to have an explicit tree structure by preserving order between siblings and parent-child relationships. In general, it is the question whether an XML document needs this type of order preserved since some XML documents could be treated as an unordered set. But for XIMPEL it is desperately needed that the XML document would be parsed as an ordered array. This feature gives more power to the programmer, in the same sense that a Turing complete system has more expressional power than a finite state machine. Without order there is chaos, and in this particular case there would sometimes be no way to determine which media item (such as a video or image) or subject should be loaded first.
The second modification helped a lot in code readability. While it is not as game changing or ground-breaking as the first, code readability is a necessary requirement for a fruitful collaboration on any software project. For example, the default setting to denote attributes of a parsed XML tag itself was `$`, to denote inner text it is `_` and children `$$`. I renamed this to: `attributes`, `text` and `children` respectively.

E.2 Architecture and implementation

Since in this exploration I tried to explore if I could reimplement XIMPEL quickly, I did not consider architecture – or even React best practices – all that much. In that sense there is quite a bit of technical debt. Creating technical debt has also been done in order to find out what architectural patterns work and which does not. Another reason to create technical debt is because I follow the programming philosophy of Jonathan Blow, which is: try to produce as fast as possible and then when you hit performance bottlenecks or any other bottleneck of any kind, only then try to be smart about solving that bottleneck. In some of his YouTube videos he goes in-depth how the code that he created for his award winning games like Braid and The Witness only have 6 to 10 percent of optimized code [1] . It has also been done in order to find in which regards one needs to fight the React framework and in which cases React gives the developer wings to fly and develop certain features of XIMPEL a lot faster.

E.2.1 SubjectRenderer: the one component to render everything else

The architecture is therefore simple and because of it in some cases a bit convoluted. It starts off with the XIMPEL playlist already being parsed by Webpack which is exposed as the `playlist` variable in `App.js` in React. In there, there is a React Component called `SubjectRenderer`, which renders everything all at once in a given subject. It does this by having a method called `createChildren` (children of a subject) which transforms the parsed XML playlist into React Elements. The render function does not need to return a whole lot since the whole tree structure of React Elements that needs to be rendered is in one variable called `children` (see code snippet E.1). From an architectural standpoint, doing it this way makes parallel play immediately possible because the `subjectRenderer` renders everything within a subject, all at once.

```
<div className="playlist">
{
  <div className="subjectRenderer">
    { React.createElement(eval("Subject"), {...element.attributes, text: element.text
    }, children) }
    <a href="#" style={{position: 'absolute', right: '50px'}} onClick={(e) => this.
    handleMediaItemClick(e)}>>></a>
  </div>
}
</div>
```

Playlist E.1: This is what is returned from the render function in the `subjectRenderer` React Component

This approach poses one big problem: how does a sequence get rendered? In normal XIMPEL playlists this still would not be a problem since in normal XIMPEL playlist, each subject tag only has one media item that needs to be rendered (e.g. see the Zaanse Schans playlist). So the problem is not with old XIMPEL playlists, it is with intermixing parallel play as opposed to sequential play within a XIMPEL subject.
The normal convention for sequential play within XIMPEL is to have a `<sequence>` tag. I however, decided against this. In the newest example apps, no one is using the `<sequence>` tag, everyone is using the `<media>` tag. The current behavior of XIMPEL React media tags is that every media item put in there (e.g. video or images) all get rendered at once, since that is what the `subjectRenderer` does. I wanted to keep this functionality, because it makes sense, and in that way one does not need to type the `<parallel>` tag which saves a bit of typing.
Instead, I opted for the possibility of putting multiple media tags in sequence (e.g. see code snippet E.2) and programmed the ‘subjectRenderer‘ in such a way that if it detects multiple media tags, then it plays these media tags compared to each other. The ‘subjectRenderer‘ would play the next media collection (with media items such as video or images in it) under one of two conditions: (1) all media items have a duration element and after the longest duration has expired it goes to the next media collection and (2) the user decides manually that he or she wants to play the next media collection by clicking on a “next media collection” button.

```
<subject id="lesson1">
  <media>
    <message message="PWD Tutorial" duration="5" />
    <video x="0px" y="200px" width="400px" height="400px" duration="10">
```

[1] I forgot which video, but here is his channel: `https://www.youtube.com/channel/UCCuoqzrsHlwv1YyPKLuMDUQ`.

```
        <source file="./pwd" extensions="mp4" types="video/mp4" />
        <overlay score="+200" scoreId="yay" leadsTo="lesson2" width="600px" height="75
    px" x="500px" y="500px" message="next lesson"/>
      </video>
  </media>
  <media>
    <textblock duration="5" message="Type in pwd and the terminal displays which
    directory you are working in." width="600px" height="800px" x="0px" y="100px" color
    ="#0f0" fontsize='50px' fontcolor="#fff"/>
    <terminal x="800px" duration="25"/>
  </media>
  <media>
    <message message="test" />
  </media>
  <youtube duration="10" id="DrgML20YxaA" width="200px" height="400px" x="1100px" y="
    200px" stopAtSubjectId="lesson3"/>
</subject>
```

Playlist E.2: caption is in the comments **to do**

From an architectural standpoint the `subjectRenderer` is possibly a bit convoluted now. Its basic functionality is to render everything in a subject all at once, but it modifies the `children` tree full of React Elements in order to allow for the intermixing of parallel play and sequential play. The reason this is perhaps convoluted is because this `subjectRenderer` does not *only* render subjects, it also controls how it renders the subject. So, for a next implementation one needs to look if there needs to be some form of modularization here.

E.2.2 Creation of React Elements

This part explains how the `createChildren` method works in the `subjectRenderer` component. The conceptualization of this method happened many months prior before its implementation and is one of the big reasons why I wanted to explore if XIMPEL could be ported quickly to React.
The method starts with a parsed XML subject sub-tree. The method traverses and transforms every XML parsed tag to a React Element via depth-first tail recursion. The exception to this rule is the parsed subject tag, since that tag needs to be transformed at render time (see code snippet E.1), it is basically a peculiarity of React.
In order to transform an XML tag into a React Element, the following needs to happen: (a) the lowercase tag name needs to change to a name that starts with a capital letter, (b) the attributes need to be extracted from the XML tag and (c) there needs to be recursively determined if there are children of said XML tag. These three things are plugged into as three arguments (argument a, b and c respectively) into the `React.createElement` api method. The magic happens when the capitalized XML tag name (see a) gets evaluated by the `eval` function. By evaluating it, it allows a name to correspond to a React component. The actual manner of how this looks like is shown in E.3.

```
children.push(React.createElement(eval(childName), {...child.attributes, text: child.
    text}, grandChildren));
```

Playlist E.3: caption is in the comments **to do**

And that is the magic of `createChildren`. Every XML tag name corresponds to a React component! When I started with exploration one and realized the similarities between the idea behind the XIMPEL playlist and React, I realized that a XIMPEL playlist is a huge set of React Component invocations. The XIMPEL playlist denotes which React component needs to be called.

E.2.3 Difference between media types in JS XIMPEL and XIMPEL React

This simple one to one mapping, which is also done in the JavaScript version of XIMPEL (JS XIMPEL)[2] allows for a clear separation of concerns. The difference between the JS XIMPEL and XIMPEL React is that in JS XIMPEL this one to one mapping needed to be more explicitly programmed compared to XIMPEL React.
Here is what is more explicit: JS XIMPEL has an in-memory configuration object, where media items were modelled. In XIMPEL React, an XML tag maps one to a React component. Such a component is capable of having properties (and state but that is not important for this argument). A React component plus properties mimics an in-memory model of a media type.
Furthermore, a React component also provides render capabilities dependent on the Component state. It is important to note that in XIMPEL React every media type Component like `Video` and `Image` inherit from a React component called `MediaType` (which does not have the same functionality as MediaType.js in JS XIMPEL), because all media types need to be able to do certain things (e.g. time tracking or knowing whether it should play or not).
While it is a rough comparison: the `MediaType` from XIMPEL React seems to mimic some of the `MediaPlayer` functionality of JS XIMPEL. The specific Media components in XIMPEL React are the same as the specific media types (e.g. Image.js) in JS XIMPEL. This is a rough comparison, because JS XIMPEL is created with a more robust architecture in mind. For example, videos or YouTube videos ends up calling an `ended` method in `MediaType.js` JS XIMPEL, meaning ending media items in JS XIMPEL is centralized. In XIMPEL React a YouTube video cannot end (at the time of writing) and a HTML5 video does not end up calling the `MediaType` component. It first calls the `handleEnd` function on itself (the `Video` component) and then calls the `SubjectRenderer` directly to change to another subject via a `leadsTo` attribute.
Furthermore, at the time of writing, it is not possible for a video that has ended to switch to a next media item if there happens to be one. Currently, it is only possible to switch to a next media item if every media item has an explicit duration. This duration should be shorter in seconds than that it takes for the HTML5 video media item to call `handleEnd` since `handleEnd` determines the next subject via the `leadsTo` attribute. In general, certain media features in XIMPEL React are more localized. Furthermore, while a large part of XIMPEL is implemented, it has not been implemented to a precise specification.

E.2.4 Difficulties encountered with React elements and React components

In general, React Elements provide a whole lot. It replaces the in-memory configuration object and complete trees of React Elements are readily available. There are however a couple of issues.
First of all, it is tough to introspect if a specific React element is currently attached to the DOM. Comparing React elements or components on contents is too difficult because React elements and React components are complex objects. A quick workaround is to only compare properties of React elements or components but the danger of that is that it assumes that the XIMPEL playlist has no XML tag with exactly the same type of attributes with the same type of values, which is a dangerous assumption in itself.
Second of all, React decides when it renders components. Sometimes this has been a problem because in some cases it would have been much easier to manually decide when a component should or should not be rendered. This could be a difficulty of React or expose my lack of professional experience as a React developer.
Third (edit: deprecated see [3] of why I leave this paragraph here), when a subject changes it may still play some of the same media types. It has different media items (e.g. a different path for a video)

[2] I know that both are in JavaScript, but ReactJS makes the difference. Hence, I call one version JS XIMPEL and the React version: XIMPEL React.

[3] I leave this paragraph here in order to show the creative process between programming and writing. Writing things down allows to make it more clear of what I am doing and that in turn improves the program as is demonstrated by this paragraph.

but it still needs to draw upon the same React components in order to render these media items (i.e. the same media type). From the perspective of React, this means that the React components that need to render the same media types should not be removed from the DOM. The difficulty associated with this is that it also means that the state of this React component does not change. And in some cases it can be relatively difficult to find a way to reset the state suited to play the new media item that uses the same media type. This is because a Video component, for example, is not itself aware which subject is playing. In the JS XIMPEL this is not a problem since every media type has a reference to the XIMPEL player via its constructor function. In React it is not that simple. A media type in XIMPEL React could use knowledge about the state of the `SubjectRenderer`. However, the state of the SubjectRenderer is a local state of that specific component. Since there only is one `SubjectRenderer` this could be solved by creating a static state, but that seems like an anti-pattern to do because Facebook (the creators of React) encourage developers to create a unidirectional data flow of state. **Perhaps I could implement the parent state being passed as a prop to all the React element children**. After, stopping with writing and trying to implement a possible solution I was forced to solve the biggest performance bottleneck that I created as technical debt in XIMPEL React. The performance bottleneck was that upon each render, the sub-tree of React elements of one subject would be recreated. Since React elements are complex objects and since the render method can be called for any reason, this is computationally very expensive. While trying to implement the idea of passing the `subjectRenderer` state as props to the complete rendered tree, the maximum call stack depth would be triggered (i.e. this shows how huge of a performance bottleneck it is). By refactoring the invocation of the `createChildren` method to the constructor and to the callback which is triggered upon the moment a subject is changed, passing down the state of the `subjectRenderer` as props to each mediaType is possible. This mimics the functionality of attaching a player object to a media type as has been done in JS XIMPEL. I know that I can improve the architecture of XIMPEL React by having made this change, but I do not fully understand how yet.

E.2.5 Overlays and scoring

Hypermedia applications have two important concepts: (1) media and (2) the fact that it is linked. Overlays allow this linking mechanism to happen within XIMPEL presentations. From an implementation standpoint it is important to understand that while overlays are related to subject, since overlays denote subject changes, they are not related to media types and media items. In that sense subjects within XIMPEL and the `subjectRenderer` specifically within XIMPEL React, serves as a controlling entity between media items and overlays.
In XIMPEL React I decided to tightly couple overlays and scoring. In JS XIMPEL this is not the case, but I could not understand the various use-cases other than clicking an overlay. There were some other use-cases (e.g. when a subject starts) but that is still possible to be modelled with overlays. Furthermore, having smart links is inspired by Ted Nelson who has a strong critique on how links are not smart enough in the current day World-Wide-Web.
The scoring mechanism has been easy to implement. In the playlist scores are made by having attaching values to the ‘score‘ attribute and the ‘scoreId‘ attribute. The ‘score‘ attribute supports values as “+1” or “/5” and will apply the operation, including the value, accordingly. The ‘scoreId‘ attribute allows for giving a specific variable name for the score, so any string that resembles a name would be an appropriate value there which XIMPEL React accepts.
By creating a static score object on the overlay component and putting a method call `handleScore` for everytime an overlay is clicked, the `Overlay` class will track any score that is being created within XIMPEL. These scores can be displayed via additional media types. In the current, implementation the `Message` media type shows a raw JSON output of scores produced within XIMPEL React.
One current implementation drawback is that overlays need to subscribe to repeating media items or subject changes. This is possibly an architectural issue or an issue with React itself. I would not know why it may be an architectural issue, but it does not seem generic enough to let an `Overlay` component listen in on subject changes and repeating videos. The reason why it may be an issue with React itself

is because the state of the overlays need to be resetted if media items repeat.

E.2.6 Data flow within XIMPEL React

The data flow within XIMPEL React tries to follow conventional ReactJS philosophy: try to have a unidirectional data flow as much as possible. However, sometimes this is not possible. If a child gets a state change earlier that a parent also would need to know, then it in some cases becomes difficult to do so. Conventional philosophy suggests to lift state. However, this has not always seemed to be possible, but furthermore it makes code readability worse. Moreover, it goes against the question of whether it is possible to reimplement XIMPEL with ReactJS quickly.
So instead of lifting state I used a trick that JS XIMPEL also used. I used a publish subscribe system. In the current implementation there are six publishers, six subscribers and five topics. This is small enough to oversee. If it gets an order of magnitude bigger, then another system needs to be considered, such as Redux.

E.2.7 Conclusion

F

Appendix Exploration 6: the code needed in YouTube.js for a partial subject change

```
var traverse = function(model, path){
  path.push(model);
  if(model instanceof ximpel.MediaModel){
    for (var i = 0; i < model.overlays.length; i++) {
      var overlay = model.overlays[i];
      for (var j = 0; j < overlay.leadsToList.length; j++) {
        var leadsTo = overlay.leadsToList[j];
        if(customAttributes.stopAtSubjectId === leadsTo.subject){
          insertModel(path.slice());
          return;
        }
      }
    }
    return;
  }
  for (var k = 0; k < model.list.length; k++) {
    traverse(model.list[k], path.slice());
  }
}.bind(this);

var insertModel = function(path){
  var subjectId = path[0].subjectId;
  var sequenceModel = this.player.subjectModels[subjectId].sequenceModel;

  var traverse = function(sequenceModel){
    for (var i = 0; i < sequenceModel.list.length; i++) {
      var model = sequenceModel.list[i];
      if(model instanceof ximpel.ParallelModel){
        var sqModel = new ximpel.SequenceModel();
        sqModel.add(this.model); //create SequenceModel
        model.add(sqModel);          //insert in Parallel Model
        return;
      } else {
        traverse(model.list[i]);
      }
    }
    return;
  }.bind(this);

  traverse(sequenceModel);
}.bind(this);
```

Bibliography

Articles

[9] Dick CA Bulterman. “Specification and support of adaptable networked multimedia”. In: *Multimedia Systems* 1.2 (1993), pp. 68–76.

[10] Vannevar Bush et al. “As we may think”. In: *The atlantic monthly* 176.1 (1945), pp. 101–108.

[12] Augusto Celentano. “Interactive web-based hypermedia coordination”. In: *Multimedia Tools and Applications* 76.4 (2017), pp. 5511–5538.

[13] Beiwen Chen et al. “Basic psychological need satisfaction, need frustration, and need strength across four cultures”. In: *Motivation and Emotion* 39.2 (2015), pp. 216–236.

[17] Open Science Collaboration et al. “Estimating the reproducibility of psychological science”. In: *Science* 349.6251 (2015), aac4716.

[18] Scotty D Craig et al. “Emote aloud during learning with AutoTutor: Applying the Facial Action Coding System to cognitive–affective states during learning”. In: *Cognition and Emotion* 22.5 (2008), pp. 777–788.

[21] Sidney K. D&#39;Mello, Scotty D. Craig, and Art C. Graesser. “Multimethod Assessment of Affective Experience and Expression During Deep Learning”. In: *Int. J. Learn. Technol.* 4.3/4 (Oct. 2009), pp. 165–187. ISSN: 1477-8386. DOI: 10.1504/IJLT.2009.028805. URL: http://dx.doi.org.vu-nl.idm.oclc.org/10.1504/IJLT.2009.028805.

[23] Manfred Del Fabro, Klaus Schoeffmann, and Laszlo Böszörmenyi. “Instant video browsing: a tool for fast non-sequential hierarchical video browsing”. In: *HCI in Work and Learning, Life and Leisure* (2010), pp. 443–446.

[24] Peter J Denning. “Is computer science science?” In: *Communications of the ACM* 48.4 (2005), pp. 27–31.

[36] Venessa Funches. “The consumer anger phenomena: causes and consequences”. In: *Journal of Services Marketing* 25.6 (2011), pp. 420–428.

[37] Ankit Gandhi, Arijit Biswas, and Om Deshmukh. “Topic Transition in Educational Videos Using Visually Salient Words.” In: *International Educational Data Mining Society* (2015).

[41] Frank Halasz et al. “The Dexter hypertext reference model”. In: *Communications of the ACM* 37.2 (1994), pp. 30–39.

[42] Lynda Hardman, Dick CA Bulterman, and Guido Van Rossum. “The Amsterdam hypermedia model: adding time and context to the Dexter model”. In: *Communications of the ACM* 37.2 (1994), pp. 50–62.

[47] Klas Ihme et al. “Frustration in the face of the driver: A simulator study on facial muscle activity during frustrated driving”. In: *Interaction Studies* (2018).

[55] Peter Kuppens et al. “The appraisal basis of anger: Specificity, necessity and sufficiency of components.” In: *Emotion* 3.3 (2003), p. 254.

[57] Chanel J. Larche, Natalia Musielak, and Mike J. Dixon. “The Candy Crush Sweet Tooth: How ‘Near-misses’ in Candy Crush Increase Frustration, and the Urge to Continue Gameplay”. In: *Journal of Gambling Studies* 33.2 (June 2017), pp. 599–615. ISSN: 1573-3602. DOI: 10.1007/s10899-016-9633-7. URL: https://doi.org/10.1007/s10899-016-9633-7.

[58] Fwa Hua Leong. “Fine-Grained Detection of Programming Students Frustration Using Keystrokes, Mouse Clicks and Interaction Logs”. In: *Journal of Social Sciences* 4 (2016), pp. 9–18.

[59] Regan L. Mandryk and M. Stella Atkins. “A Fuzzy Physiological Approach for Continuously Modeling Emotion During Interaction with Play Technologies”. In: *Int. J. Hum.-Comput. Stud.* 65.4 (Apr. 2007), pp. 329–347. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2006.11.011. URL: http://dx.doi.org.vu-nl.idm.oclc.org/10.1016/j.ijhcs.2006.11.011.

[61] Britta Meixner, Stefan John, and Christian Handschigl. “SIVA suite: an open-source framework for hypervideos”. In: *ACM SIGMultimedia Records* 8.1 (2016), pp. 10–14.

[62] Edward Melcer et al. “Games research today: Analyzing the academic landscape 2000-2014”. In: *network* 17 (2015), p. 20.

[63] S. Milborrow, J. Morkel, and F. Nicolls. “The MUCT Landmarked Face Database”. In: *Pattern Recognition Association of South Africa* (2010). http://www.milbo.org/muct.

[69] Andrew Ortony and Terence J Turner. “What’s basic about basic emotions?” In: *Psychological review* 97.3 (1990), p. 315.

[71] JR van Ossenbruggen. “Processing Structured Hypermedia-A matter of style”. In: (2001).

[80] Maaike van Rest et al. “De Sociale Informatieverwerkingstest (SIVT) Voor Jongeren Binnen Gesloten Residentiële Jeugdzorg.” In: *Onderzoek en praktijk. Tijdschrift voor de LVG-zorg* 12 (2014). The language of the article is in Dutch, pp. 31–42.

[92] Klaus Schoeffmann, Marco A Hudelist, and Jochen Huber. “Video interaction tools: A survey of recent work”. In: *ACM Computing Surveys (CSUR)* 48.1 (2015), p. 14.

[93] Rajvi Shah and PJ Narayanan. “Interactive video manipulation using object trajectories and scene backgrounds”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 23.9 (2013), pp. 1565–1576.

[94] Paul Lucian Szasz, Aurora Szentagotai, and Stefan G Hofmann. “The effect of emotion regulation strategies on anger”. In: *Behaviour Research and Therapy* 49.2 (2011), pp. 114–119.

[102] Toshio Yamagishi et al. “Rejection of unfair offers in the ultimatum game is no evidence of strong reciprocity”. In: *Proceedings of the National Academy of Sciences* 109.50 (2012), pp. 20364–20368.

Conference Papers

[11] Antonio José G. Busson et al. “A Hypervideo Model for Learning Objects”. In: *Proceedings of the 28th ACM Conference on Hypertext and Social Media*. HT ’17. Prague, Czech Republic: ACM, 2017, pp. 245–253. ISBN: 978-1-4503-4708-2. DOI: 10.1145/3078714.3078739. URL: http://doi.acm.org.vu-nl.idm.oclc.org/10.1145/3078714.3078739.

[14] Kai-Yin Cheng et al. “SmartPlayer: User-centric Video Fast-forwarding”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’09. YouTube: https://www.youtube.com/watch?v=-VGBOR1ZNw0. Boston, MA, USA: ACM, 2009, pp. 789–798. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518823. URL: http://doi.acm.org.vu-nl.idm.oclc.org/10.1145/1518701.1518823.

[16] Claudiu Cobârzan and Klaus Schoeffmann. “How do users search with basic html5 video players?” In: *International Conference on Multimedia Modeling*. Springer. 2014, pp. 109–120.

[25] Ansgar E Depping et al. “Trust Me: Social Games are Better than Social Icebreakers at Building Trust”. In: *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. ACM. 2016, pp. 116–129.

[27] Pierre Dragicevic et al. “Video Browsing by Direct Manipulation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’08. YouTube: https://www.youtube.com/watch?v=WcIy9O344bI. Florence, Italy: ACM, 2008, pp. 237–246. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357096. URL: http://doi.acm.org.vu-nl.idm.oclc.org/10.1145/1357054.1357096.

[30] Anton Eliëns et al. "Clima Futura@ VU–communicating (unconvenient) science." In: *GAMEON*. 2007, pp. 125–129.

[31] Anton Eliëns et al. "XIMPEL Interactive Video-between narrative (s) and game play." In: *GAMEON*. 2008, pp. 132–136.

[38] Kiel M Gilleade and Alan Dix. "Using frustration in the design of adaptive videogames". In: *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM. 2004, pp. 228–232.

[39] Dan B Goldman et al. "Video object annotation, navigation, and composition". In: *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM. 2008, pp. 3–12.

[40] Joseph F Grafsgaard et al. "Automatically recognizing facial indicators of frustration: a learning-centric analysis". In: *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*. IEEE. 2013, pp. 159–165.

[45] Wolfgang Hürst and Dimitri Darzentas. "HiStory: a hierarchical storyboard interface design for video browsing on mobile devices". In: *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*. ACM. 2012, p. 17.

[51] Daniel Karavolos, Anders Bouwer, and Rafael Bidarra. "Mixed-initiative design of game levels: integrating mission and space into level generation". In: *Proceedings of FDG 2015 - Tenth International Conference on the Foundations of Digital Games*. June 2015. URL: http://graphics.tudelft.nl/Publications-new/2015/KBB15.

[52] Juho Kim et al. "Data-driven interaction techniques for improving navigation of educational videos". In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM. 2014, pp. 563–572.

[53] Don Kimber et al. "Trailblazing: Video playback control by direct object manipulation". In: *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE. 2007, pp. 1015–1018.

[56] Rodrigo Laiola Guimarães et al. "Synchronizing web documents with style". In: *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*. ACM. 2014, pp. 151–158.

[60] Regan L. Mandryk, M. Stella Atkins, and Kori M. Inkpen. "A Continuous and Objective Evaluation of Emotional Experience with Interactive Play Environments". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. Montr&#233;al, Qu&#233;bec, Canada: ACM, 2006, pp. 1027–1036. ISBN: 1-59593-372-7. DOI: 10.1145/1124772.1124926. URL: http://doi.acm.org.vu-nl.idm.oclc.org/10.1145/1124772.1124926.

[66] Lus A. R. Neng and Teresa Chambel. "Get Around 360&Deg; Hypervideo". In: *Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments*. MindTrek '10. Tampere, Finland: ACM, 2010, pp. 119–122. ISBN: 978-1-4503-0011-7. DOI: 10.1145/1930488.1930512. URL: http://doi.acm.org.vu-nl.idm.oclc.org/10.1145/1930488.1930512.

[**67**] Cuong Nguyen, Yuzhen Niu, and Feng Liu. "Direct manipulation video navigation in 3D". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. YouTube: https://www.youtube.com/watch?v=YomeZfCo7P0. ACM. 2013, pp. 1169–1172.

[68] Carl Magnus Olsson, Staffan Björk, and Steve Dahlskog. "The conceptual relationship model: understanding patterns and mechanics in game design". In: DiGRA. 2014.

[**77**] Suporn Pongnumkul et al. "Content-aware dynamic timeline for video browsing". In: *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. YouTube: https://www.youtube.com/watch?v=fqf_w5JE6CA. ACM. 2010, pp. 139–142.

[78] David Portugal et al. "Identification of an Individual's Frustration in the Work Environment Through a Multi-sensor Computer Mouse". In: *Human Aspects of IT for the Aged Population. Healthy and Active Aging*. Ed. by Jia Zhou and Gavriel Salvendy. Cham: Springer International Publishing, 2016, pp. 79–88. ISBN: 978-3-319-39949-2.

[89] Melvin Roest and Sander Bakkes. "Engaging Casual Games That Frustrate You: An Exploration on Understanding Engaging Frustrating Casual Games." In: FDG. 2015.

[91] Jason M Saragih, Simon Lucey, and Jeffrey F Cohn. “Face alignment through subspace constrained mean-shifts”. In: *Computer Vision, 2009 IEEE 12th International Conference on*. Ieee. 2009, pp. 1034–1041.

[101] Kuldeep Yadav et al. “Content-driven Multi-modal Techniques for Non-linear Video Navigation”. In: *Proceedings of the 20th International Conference on Intelligent User Interfaces*. IUI ’15. Atlanta, Georgia, USA: ACM, 2015, pp. 333–344. ISBN: 978-1-4503-3306-1. DOI: `10.1145/2678025.2701408`. URL: `http://doi.acm.org.vu-nl.idm.oclc.org/10.1145/2678025.2701408`.

Books

[26] Joris Dormans et al. *Engineering emergence: applied theory for game design*. Universiteit van Amsterdam [Host], 2012.

[34] Paul Feyerabend. *Against method*. Verso, 1993.

[64] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1967. ISBN: 0-13-165563-9.

Book Chapters

[4] W. Bhikharie and A.P.W. Eliëns. “XIMPEL for Education – inspiring creativity through storytelling and gameplay”. In: *GAMEON 2015*. Eurosis, 2016.

[6] Ian Bogost. “Procedural rhetoric”. In: *Persuasive games: The expressive power of videogames*. MIT Press, 2007. Chap. 1, pp. 1–64.

Chapters in a Collection

[83] Manuel Rodrigues et al. “Keystrokes and clicks: Measuring stress on e-learning students”. In: *Management Intelligent Systems*. Springer, 2013, pp. 119–126.

Master Theses

[8] Stefan Bruins. “The interactive media framework XIMPEL”. MA thesis. the Netherlands: Vrije Universiteit Amsterdam, 2016.

Reports

[81] Heather Anne Richter et al. *A multi-scale timeline slider for stream visualization and control*. Tech. rep. Georgia Institute of Technology, 1999.

Github Repositories

[7] brightcove. *Video.js Overlay*. 2018. URL: `https://github.com/brightcove/videojs-overlay` (visited on 03/28/2018).

[43] Dan Michael O. Heggö. *Microticks*. 2016. URL: `https://github.com/scriptotek/microticks` (visited on 04/30/2018).

[49] Philip Jägenstedt, Michael Smith, and Anne van Kesteren. *HTML Standard FAQ*. 2017. URL: `https://github.com/whatwg/html/blob/master/FAQ.md` (visited on 04/30/2018).

[54] Marek Kubica. *node-xml2js*. 2018. URL: https://github.com/Leonidas-from-XIV/node-xml2js (visited on 04/26/2018).
[65] mrhenry. *Overlay.js*. 2016. URL: https://github.com/mrhenry/overlay-js (visited on 03/28/2018).
[75] Audun Mathias Øygard and Melvin Roest. *Why does the example only have 4 emotions, while the emotionmodel.js has 6?* 2017. URL: https://github.com/auduno/clmtrackr/issues/130 (visited on 10/17/2017).
[82] Morgan Roderick. *PubSubJS*. 2018. URL: https://github.com/mroderick/PubSubJS (visited on 04/24/2018).
[84] Melvin Roest. *XIMPEL Analytics Server*. 2018. URL: https://github.com/melvinroest/ximpel-analytics-server (visited on 05/04/2018).
[85] Melvin Roest. *XIMPEL FSM*. 2018. URL: https://github.com/melvinroest/XIMPEL-FSM (visited on 06/15/2018).
[86] Melvin Roest. *XIMPEL js*. 2018. URL: https://github.com/melvinroest/XIMPEL-JS (visited on 05/04/2018).
[87] Melvin Roest. *XIMPEL React*. 2018. URL: https://github.com/melvinroest/XIMPEL-React (visited on 05/04/2018).
[88] Melvin Roest. *XIMPEL Terminal Media Type Server*. 2018. URL: https://github.com/melvinroest/ximpel-terminal-media-type-server (visited on 05/04/2018).
[90] Melvin Roest and Audun M. Øygard. *Github Issues - I don't get something in the fitting faces article, does an SVM kernel when doing logistic regression really exist? Isn't it just a linear kernel?* 2018. URL: https://github.com/auduno/clmtools/issues/11 (visited on 05/03/2018).

Hip Blog Posts From The Web

[2] Fabrice Bellard. *JSLinux*. 2011. URL: https://bellard.org/jslinux/vm.html?url=https://bellard.org/jslinux/buildroot-x86.cfg (visited on 06/12/2018).
[19] Jason Cranfordteague. *Program or be Programmed: The GeekDad Interview With Douglas Rushkoff*. 2011. URL: https://www.wired.com/2011/07/douglas-rushkoff/ (visited on 06/19/2018).
[76] PhiNotPi et al. *code challenge - Build a working game of Tetris in Conway's Game of Life - Programming Puzzles & Code Golf Stack Exchange*. 2014. URL: https://codegolf.stackexchange.com/questions/11880/build-a-working-game-of-tetris-in-conways-game-of-life (visited on 06/12/2018).
[95] Julian Togelius. *The differences between tinkering and research*. 2016. URL: http://togelius.blogspot.com/2016/04/the-differences-between-tinkering-and.html (visited on 06/23/2018).
[98] Alvin Ward. *24 Unintended Scientific Discoveries*. 2015. URL: http://mentalfloss.com/article/53646/24-important-scientific-discoveries-happened-accident (visited on 06/12/2018).
[100] women-inventors.com. *Ruth Wakefield - Chocolate Chip Cookie Inventor*. 2006. URL: http://www.women-inventors.com/Ruth-Wakefield.asp (visited on 06/12/2018).
[104] Andreas Zwinkau. *Accidentally Turing-Complete*. 2018. URL: http://beza1e1.tuxen.de/articles/accidentally_turing_complete.html (visited on 06/12/2018).

Hypermedia Companies

[28] Eko. *Eko*. 2018. URL: https://helloeko.com/ (visited on 06/19/2018).
[99] Wirewax. *Wirewax - Interactive Video*. 2018. URL: https://www.wirewax.com/ (visited on 06/19/2018).

[103] Zentrick. *Zentrick - Make every video ad count.* 2018. URL: https://www.zentrick.com/ (visited on 06/19/2018).

Other Online Sources

[1] Sophie Alpert. *Discontinuing IE 8 Support in React DOM.* 2018. URL: https://reactjs.org/blog/2016/01/12/discontinuing-ie8-support.html (visited on 04/30/2018).

[3] Tim Berners-Lee. *Tags used in HTML.* 1992. URL: https://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html (visited on 04/09/2018).

[15] Alan Chodos and Jennifer Ouellette. *This Month in Physics History - June 1963: Discovery of the Cosmic Microwave Background.* 2002. URL: https://www-aps-org.vu-nl.idm.oclc.org/publications/apsnews/200207/history.cfm (visited on 06/12/2018).

[20] Joel La Croix and Matt Hristovski. *CPS 621 - SMIL Presentation W011.* 2011. URL: https://www.youtube.com/watch?v=ZVA4UyyhNtI (visited on 11/09/2018).

[22] Inc. DataCamp. *How to run R in the cloud (for teaching).* 2013. URL: https://www.r-bloggers.com/how-to-run-r-in-the-cloud-for-teaching/ (visited on 04/17/2018).

[32] Douglas Engelbart. *AUGMENTING HUMAN INTELLECT: A CONCEPTUAL FRAMEWORK.* 1962. URL: http://www.dougengelbart.org/pubs/augment-3906.html (visited on 04/24/2018).

[33] Douglas Engelbart. *Implemented Hypermedia in the '60s.* 2018. URL: http://dougengelbart.org/firsts/hypertext.html (visited on 04/24/2018).

[35] Mark Leighton Fisher. *Tim Berners-Lee, the World Wide Web, and the Dexter Model. Journal of Mark Leighton Fisher (4252).* 2007. URL: http://use.perl.org/use.perl.org/_Mark%5C%2BLeighton%5C%2BFisher/journal/35016.html (visited on 09/29/2017).

[46] Hugo C. Huurdeman. *Engagement with the Abel Prize via a Touch Table App.* 2017. URL: http://www.ub.uio.no/om/prosjekter/the-visualisation-project/news/abel-prize-app.html (visited on 11/09/2018).

[48] Doug Engelbart Institute. *Doug's Great Demo: 1968.* 1968. URL: http://www.dougengelbart.org/firsts/dougs-1968-demo.html (visited on 06/24/2018).

[50] Jack Jansen. *SMIL, Synchronized Multimedia Integration Language.* 2011. URL: https://www.youtube.com/watch?v=xqups1sSlHI&lc= (visited on 11/09/2018).

[70] Universitetet i Oslo - Universitetsbiblioteket. *Invisible Interactions: Studying In-Library Usage of a Movie-Related Touch Table Application (Explorative Study).* 2017. URL: http://www.ub.uio.no/om/prosjekter/the-visualisation-project/news/invisible-touch-table-interactions.html (visited on 04/30/2018).

[72] Audun M. Øygard. *Fitting Faces.* 2014. URL: https://www.auduno.com/2014/01/05/fitting-faces/ (visited on 03/28/2018).

[73] Audun Mathias Øygard. *PDM Modelviewer.* 2017. URL: https://www.auduno.com/clmtrackr/examples/modelviewer_pca.html (visited on 04/30/2018).

[74] Audun Mathias Øygard. *Twisting Faces.* 2014. URL: https://www.auduno.com/2014/04/29/twisting-faces/ (visited on 04/30/2018).

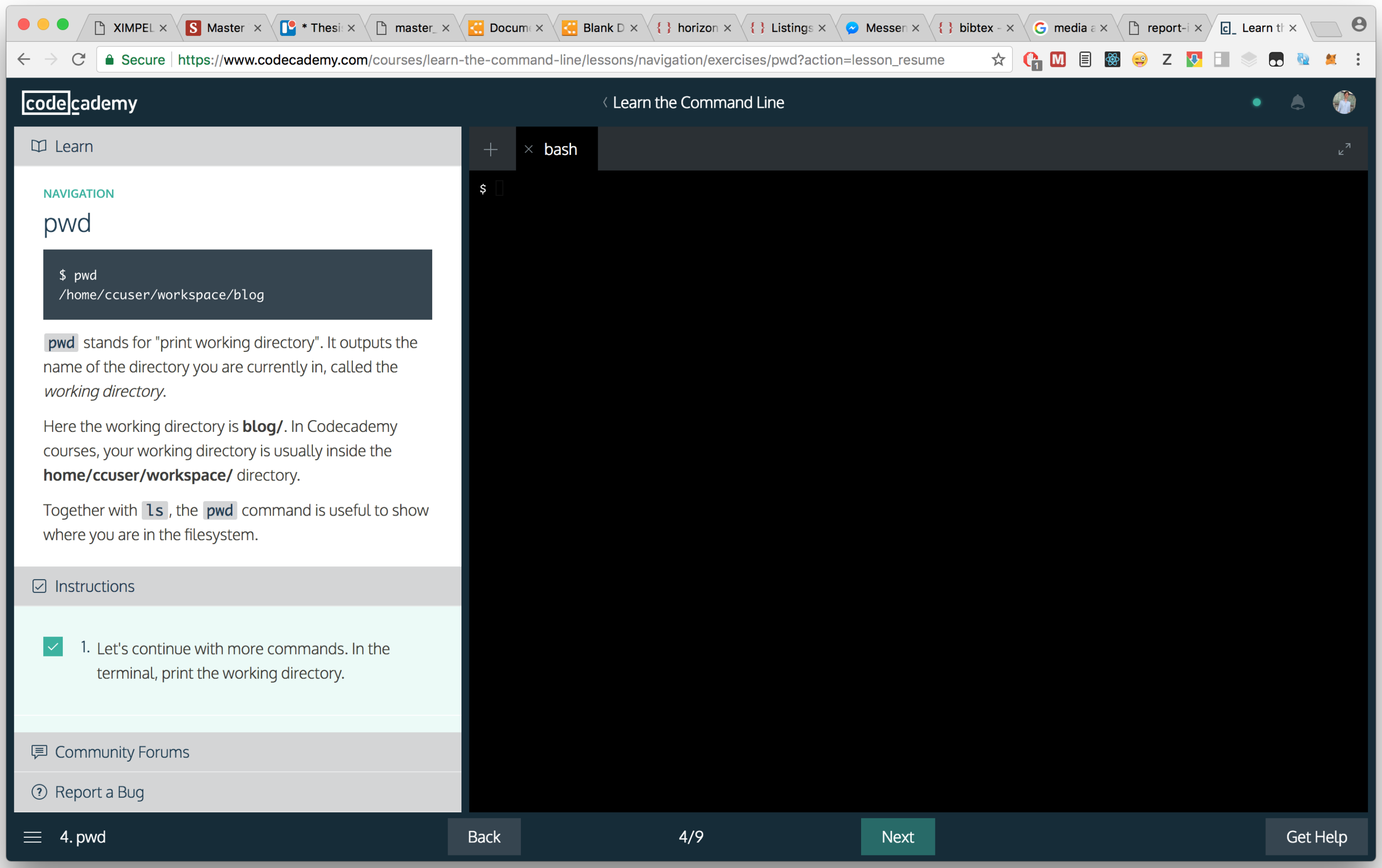
[79] R.e.s. et al. *Two-counter machines are Turing equivalent (with a caveat).* 2007. URL: https://en.wikipedia.org/wiki/Counter_machine#Two-counter_machines_are_Turing_equivalent_(with_a_caveat) (visited on 05/26/2018).

[96] W3C. *SMIL Current Status - W3C.* 2017. URL: https://www.w3.org/standards/techs/smil#w3c_all (visited on 09/29/2017).

[97] W3C and Thierry Michel. *SMIL Current Status - W3C.* 2016. URL: https://www.w3.org/AudioVideo (visited on 09/29/2017).

Misc

[5] Winoe Bhikharie. *XIMPEL Meeting (personal communication)*. Mar. 2018.
[29] A. Eliëns. private communication. 2017.
[44] Bos Herbert and C. Giuffrida. *Binary and Malware Analysis (personal communication during lecture)*. Sept. 2015.



https://www.codecademy.com/courses/learn-the-command-line/lessons/navigation/exercises/pwd?action=lesson_resume

NAVIGATION

pwd

```
$ pwd
/home/ccuser/workspace/blog
```

`pwd` stands for "print working directory". It outputs the name of the directory you are currently in, called the *working directory*.

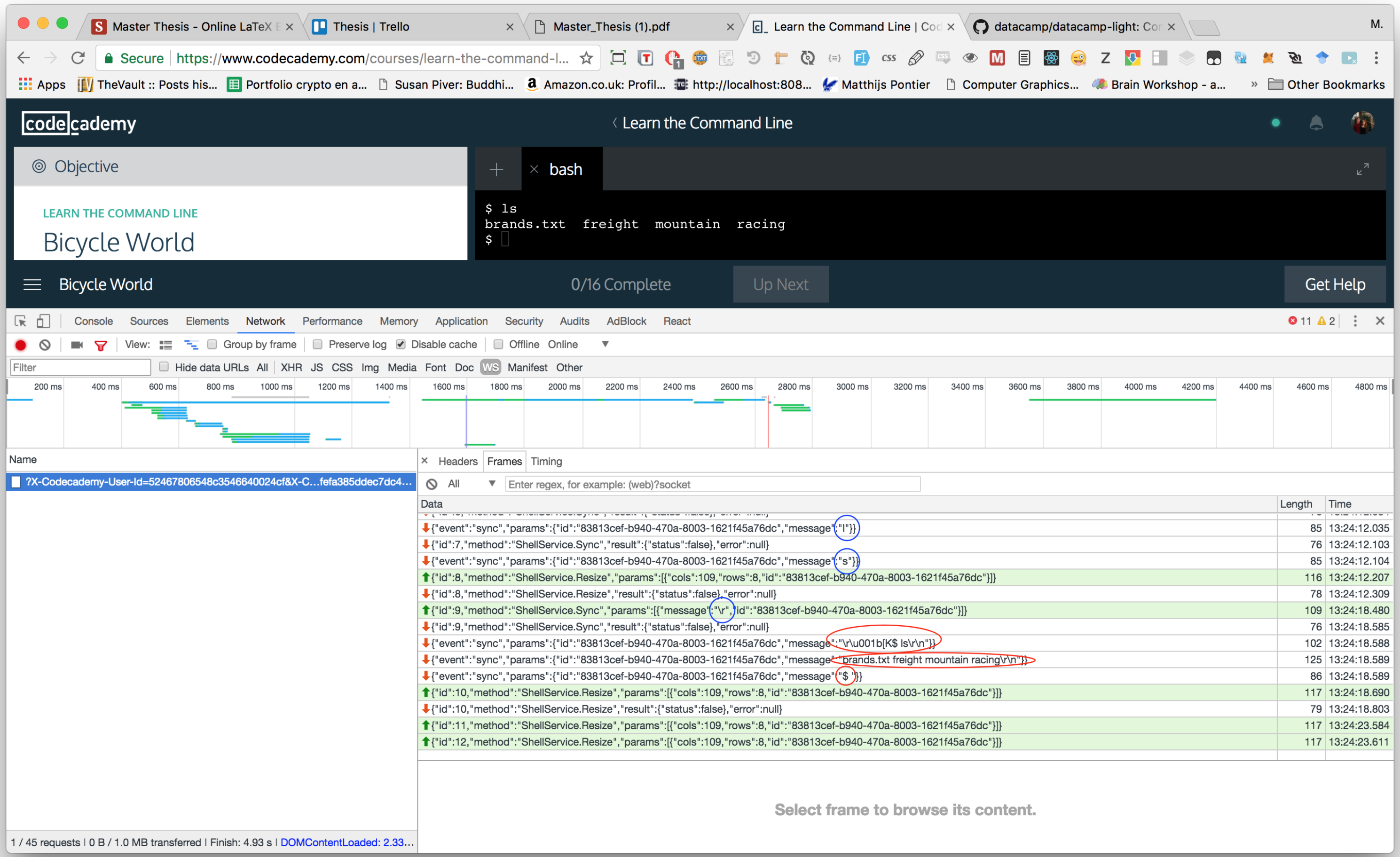
Here the working directory is **blog/**. In Codecademy courses, your working directory is usually inside the **home/ccuser/workspace/** directory.

Together with `ls`, the `pwd` command is useful to show where you are in the filesystem.

✓ 1. Let's continue with more commands. In the terminal, print the working directory.

```
$
```

Figure F.1: A lesson in the Codecademy command-line tutorial. This lesson shows how to use the print working directory command. The lesson has two important panes. The left pane shows the lesson, while the right pane shows a tutorial.



https://www.codecademy.com/courses/learn-the-command-l...

Learn the Command Line

LEARN THE COMMAND LINE

Bicycle World

```
$ ls
brands.txt  freight  mountain  racing
$
```

Bicycle World — 0/16 Complete — Up Next — Get Help

Name: ?X-Codecademy-User-Id=52467806548c3546640024cf&X-C...fefa385ddec7dc4...

```
Data                                                                                                                   Length  Time
↓{"event":"sync","params":{"id":"83813cef-b940-470a-8003-1621f45a76dc","message":"l"}}                                 85  13:24:12.035
↓{"id":7,"method":"ShellService.Sync","result":{"status":false},"error":null}                                          76  13:24:12.103
↓{"event":"sync","params":{"id":"83813cef-b940-470a-8003-1621f45a76dc","message":"s"}}                                 85  13:24:12.104
↑{"id":8,"method":"ShellService.Resize","params":[{"cols":109,"rows":8,"id":"83813cef-b940-470a-8003-1621f45a76dc"}]}  116  13:24:12.207
↓{"id":8,"method":"ShellService.Resize","result":{"status":false},"error":null}                                        78  13:24:12.309
↑{"id":9,"method":"ShellService.Sync","params":[{"message":"\r","id":"83813cef-b940-470a-8003-1621f45a76dc"}]}          109  13:24:18.480
↓{"id":9,"method":"ShellService.Sync","result":{"status":false},"error":null}                                          76  13:24:18.585
↓{"event":"sync","params":{"id":"83813cef-b940-470a-8003-1621f45a76dc","message":"\r\u001b[K$ ls\r\n"}}               102  13:24:18.588
↓{"event":"sync","params":{"id":"83813cef-b940-470a-8003-1621f45a76dc","message":"brands.txt freight mountain racing\r\n"}}  125  13:24:18.589
↓{"event":"sync","params":{"id":"83813cef-b940-470a-8003-1621f45a76dc","message":"$ "}}                                86  13:24:18.589
↑{"id":10,"method":"ShellService.Resize","params":[{"cols":109,"rows":8,"id":"83813cef-b940-470a-8003-1621f45a76dc"}]}  117  13:24:18.690
↓{"id":10,"method":"ShellService.Resize","result":{"status":false},"error":null}                                       79  13:24:18.803
↑{"id":11,"method":"ShellService.Resize","params":[{"cols":109,"rows":8,"id":"83813cef-b940-470a-8003-1621f45a76dc"}]}  117  13:24:23.584
↑{"id":12,"method":"ShellService.Resize","params":[{"cols":109,"rows":8,"id":"83813cef-b940-470a-8003-1621f45a76dc"}]}  117  13:24:23.611
```

Select frame to browse its content.

1 / 45 requests | 0 B / 1.0 MB transferred | Finish: 4.93 s | DOMContentLoaded: 2.33...

Figure F.2: Evidence that CodeCademy uses websockets to communicate with a terminal on some server. The blue circles shows data that is sent from the client to the server. The red circles shows data that is sent from the server to the client.

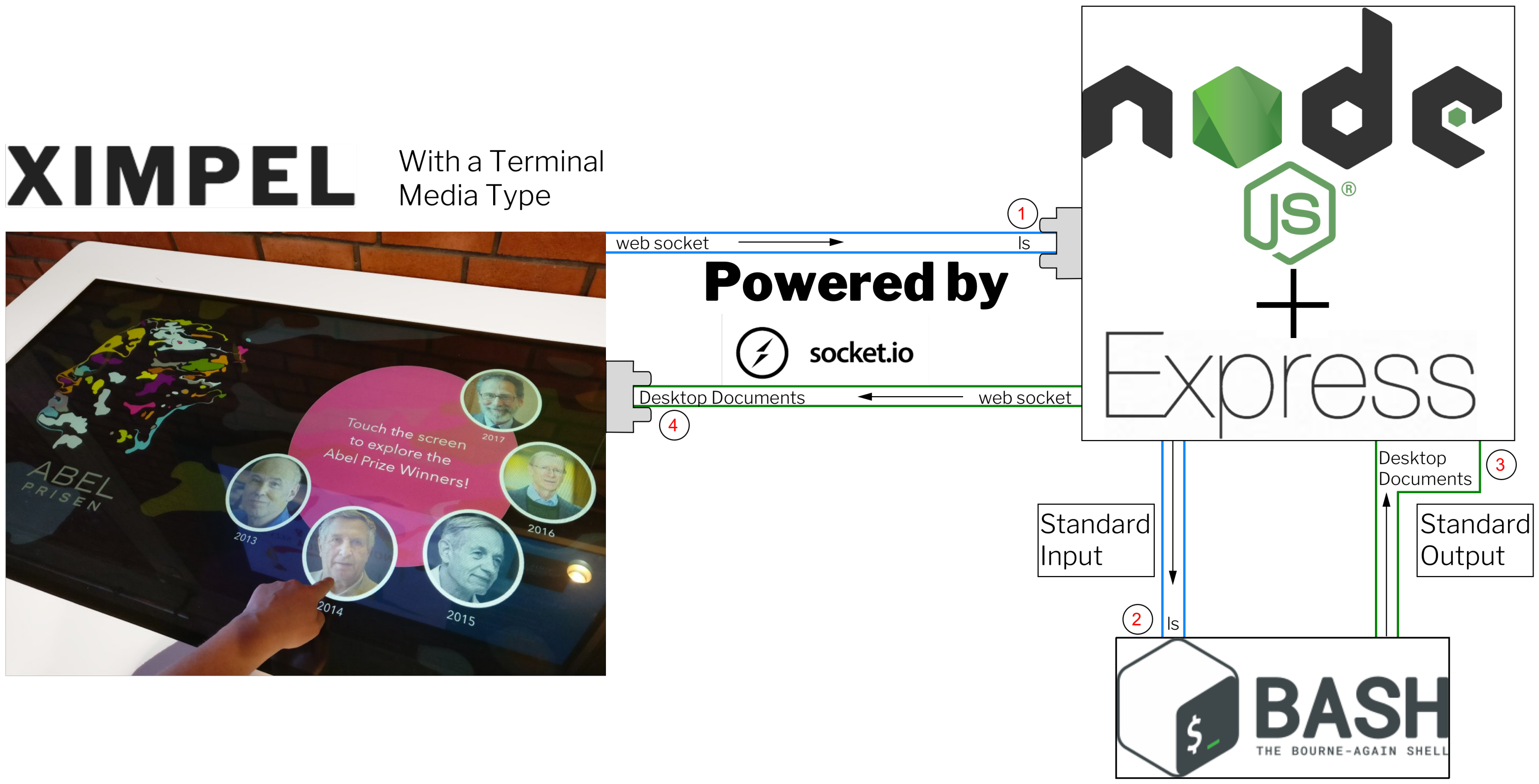


Figure F.3: Communication example of the terminal media type. (1) The client sends a command, for example `ls`. (2) The NodeJS server sends this to a bash shell. (3) The bash shell sends a response to NodeJS, for example *Desktop Documents* and (4) NodeJS sends that back to the client. Note that this is an architecture for prototypes only. The bash shell should be replaced by Docker or Vagrant in order to sandbox bash. Even then, possible commands in bash should probably be limited due to undisclosed sandbox escape vulnerability bugs.

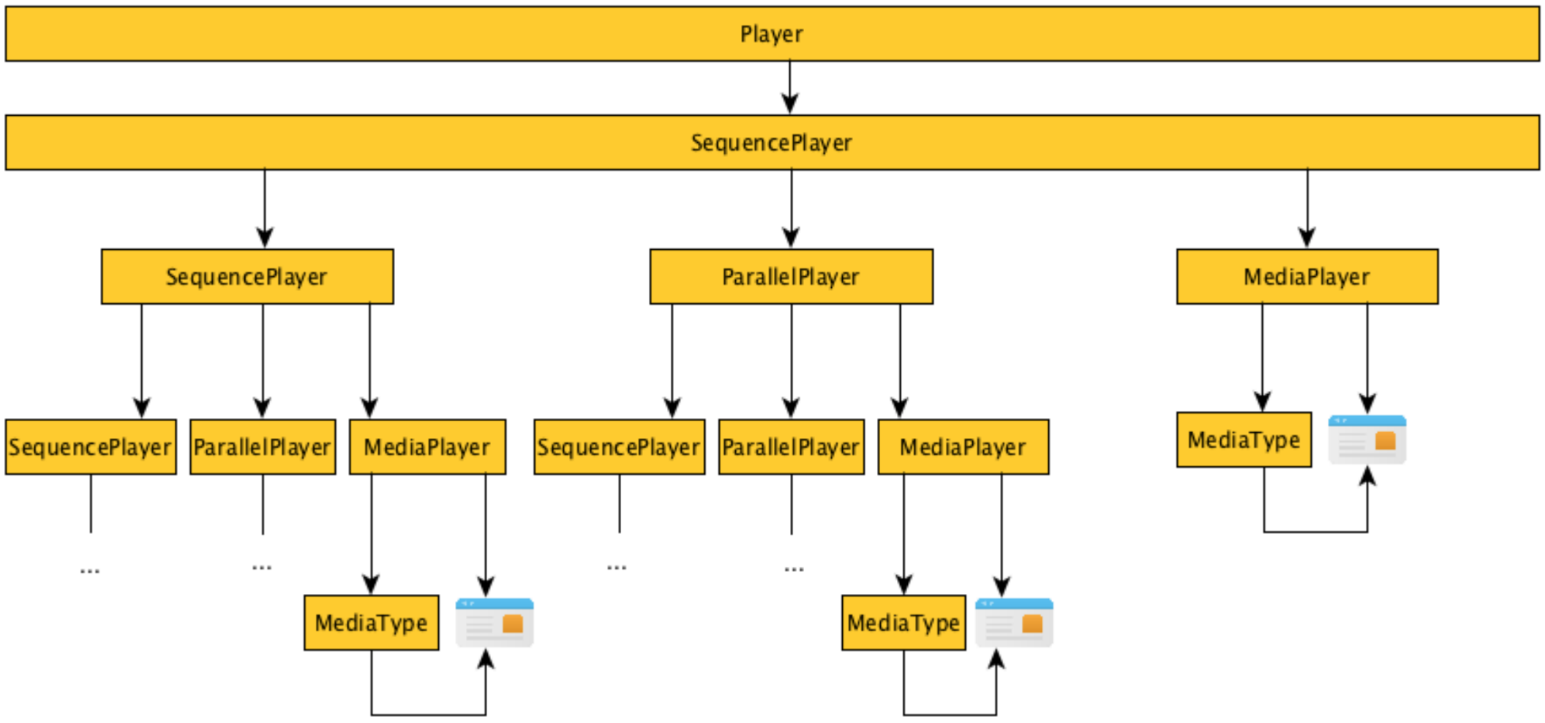


Figure F.4: The original player architecture of XIMPEL. The media type gives rise to media which can be: video, audio, image, text or a custom defined media type. This architecture has been proposed and (partially) implemented by Stefan Bruins [8]. It has been demonstrated by me that nested SequencePlayers have not been implemented and I also explained why this does not need to happen in the beginning of this chapter. The diagram shows the possible valid children a player would be able to play.

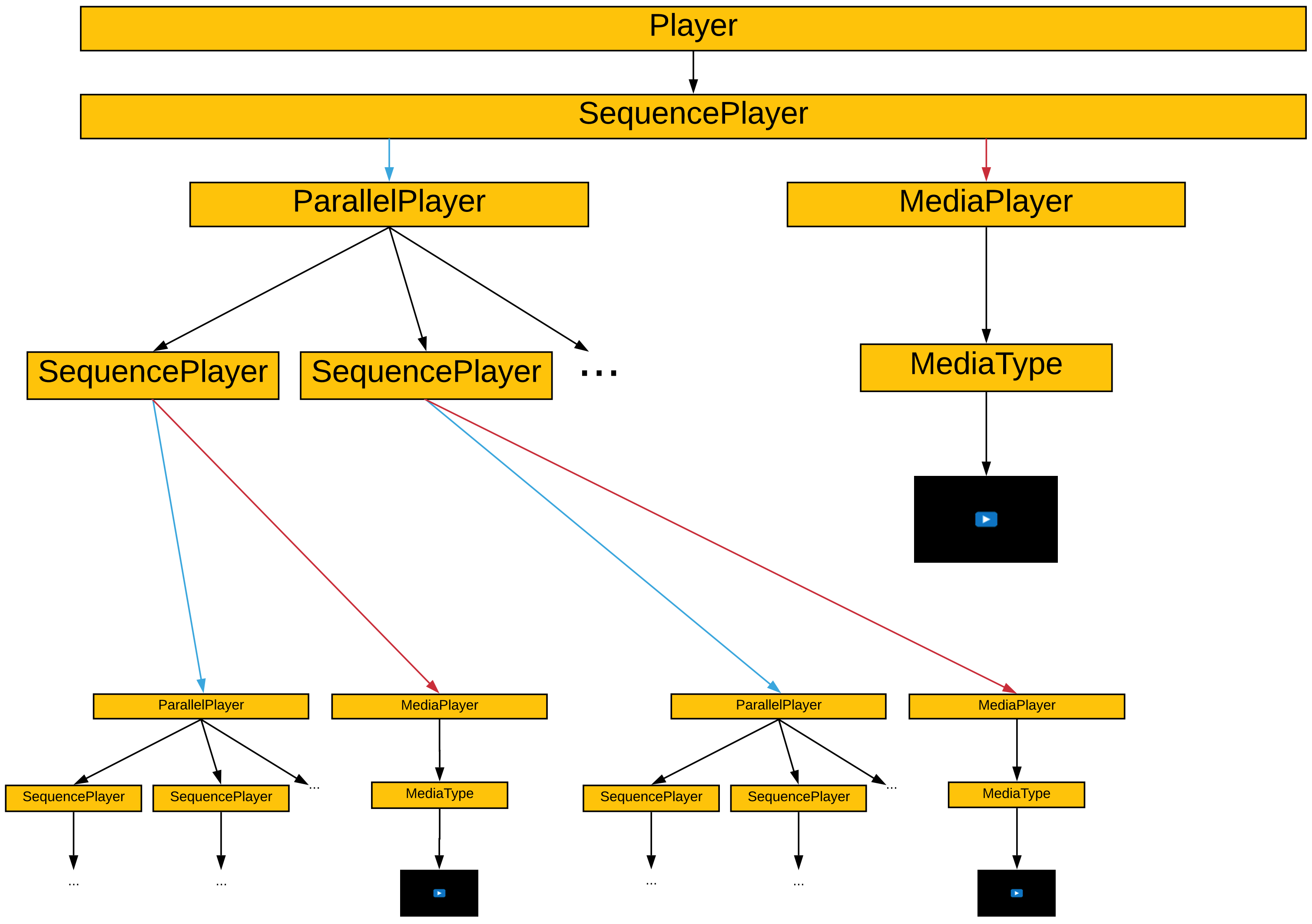
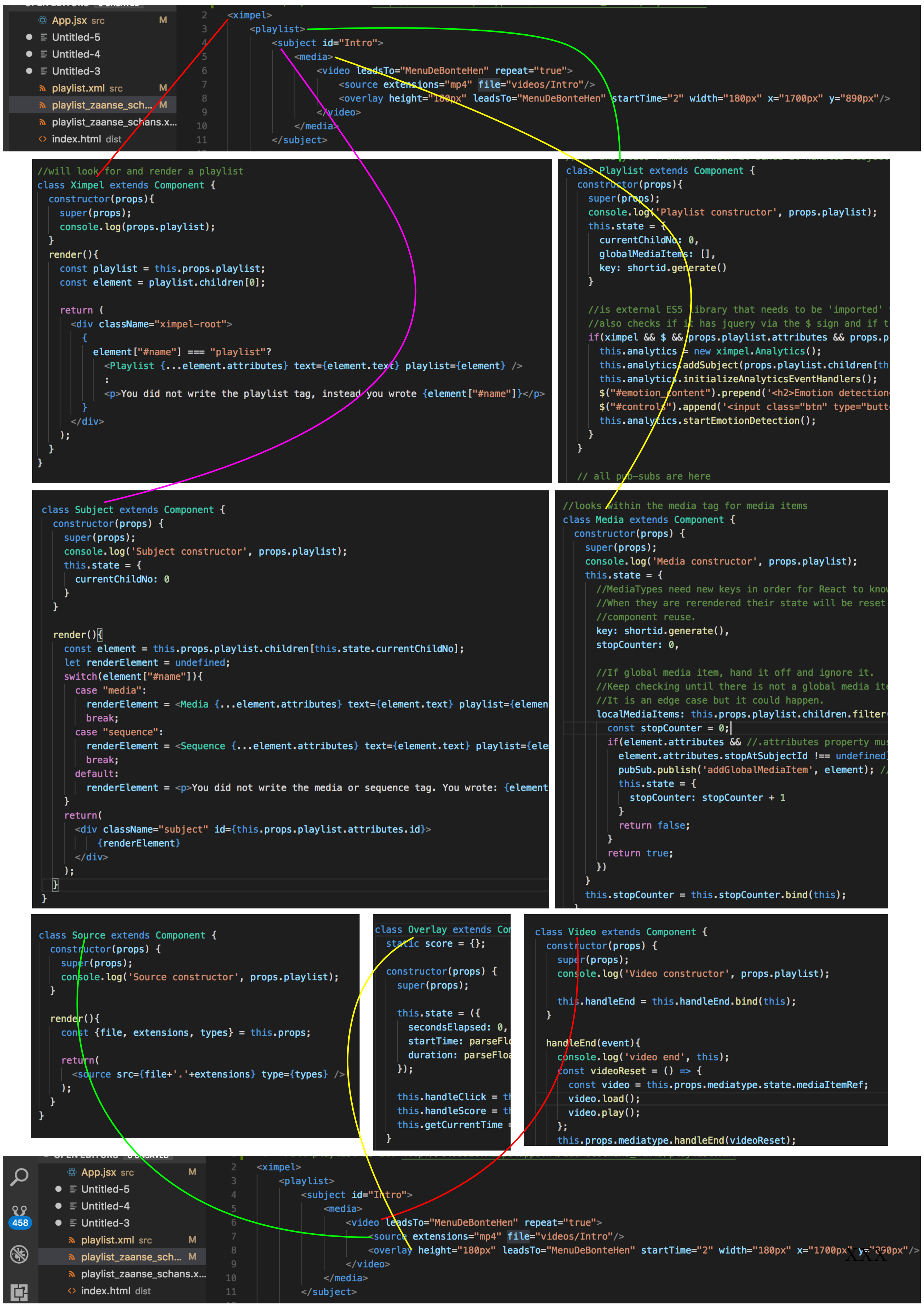


Figure F.5: The current player architecture of XIMPEL. The media type gives rise to media which can be: video, audio, image, text or a custom defined media type. The diagram shows which possible players can be played but also how many players another player can play. A blue or red line indicates that the player of which the line originates needs to choose between one of them. A black line means that a player will be invoked by another player.



```xml
<ximpel>
    <playlist>
        <subject id="Intro">
            <media>
                <video leadsTo="MenuDeBonteHen" repeat="true">
                    <source extensions="mp4" file="videos/Intro"/>
                    <overlay height="180px" leadsTo="MenuDeBonteHen" startTime="2" width="180px" x="1700px" y="890px"/>
                </video>
            </media>
        </subject>
```

```js
//will look for and render a playlist
class Ximpel extends Component {
  constructor(props){
    super(props);
    console.log(props.playlist);
  }
  render(){
    const playlist = this.props.playlist;
    const element = playlist.children[0];

    return (
      <div className="ximpel-root">
        {
          element["#name"] === "playlist"?
            <Playlist {...element.attributes} text={element.text} playlist={element} />
            :
            <p>You did not write the playlist tag, instead you wrote {element["#name"]}</p>
        }
      </div>
    );
  }
}
```

```js
class Playlist extends Component {
  constructor(props){
    super(props);
    console.log('Playlist constructor', props.playlist);
    this.state = {
      currentChildNo: 0,
      globalMediaItems: [],
      key: shortid.generate()
    }

    //is external ES5 library that needs to be 'imported'
    //also checks if it has jquery via the $ sign and if t
    if(ximpel && $ && props.playlist.attributes && props.p
      this.analytics = new ximpel.Analytics();
      this.analytics.addSubject(props.playlist.children[th
      this.analytics.initializeAnalyticsEventHandlers();
      $("#emotion_content").prepend('<h2>Emotion detection
      $("#controls").append('<input class="btn" type="butt
      this.analytics.startEmotionDetection();
    }
  }

  // all pub-subs are here
```

```js
class Subject extends Component {
  constructor(props) {
    super(props);
    console.log('Subject constructor', props.playlist);
    this.state = {
      currentChildNo: 0
    }
  }

  render(){
    const element = this.props.playlist.children[this.state.currentChildNo];
    let renderElement = undefined;
    switch(element["#name"]){
      case "media":
        renderElement = <Media {...element.attributes} text={element.text} playlist={elemen
        break;
      case "sequence":
        renderElement = <Sequence {...element.attributes} text={element.text} playlist={ele
        break;
      default:
        renderElement = <p>You did not write the media or sequence tag. You wrote: {element
    }
    return(
      <div className="subject" id={this.props.playlist.attributes.id}>
        {renderElement}
      </div>
    );
  }
}
```

```js
//looks within the media tag for media items
class Media extends Component {
  constructor(props) {
    super(props);
    console.log('Media constructor', props.playlist);
    this.state = {
      //MediaTypes need new keys in order for React to kno
      //When they are rerendered their state will be reset
      //component reuse.
      key: shortid.generate(),
      stopCounter: 0,

      //If global media item, hand it off and ignore it.
      //Keep checking until there is not a global media ite
      //It is an edge case but it could happen.
      localMediaItems: this.props.playlist.children.filter(
        const stopCounter = 0;
        if(element.attributes && //.attributes property mus
          element.attributes.stopAtSubjectId !== undefined
          pubSub.publish('addGlobalMediaItem', element); /
          this.state = {
            stopCounter: stopCounter + 1
          }
          return false;
        }
        return true;
      })
    }
    this.stopCounter = this.stopCounter.bind(this);
```

```js
class Source extends Component {
  constructor(props) {
    super(props);
    console.log('Source constructor', props.playlist);
  }

  render(){
    const {file, extensions, types} = this.props;

    return(
      <source src={file+'.'+extensions} type={types} />
    );
  }
}
```

```js
class Overlay extends Co
  static score = {};

  constructor(props) {
    super(props);

    this.state = ({
      secondsElapsed: 0,
      startTime: parseFl
      duration: parseFloa
    });

    this.handleClick = t
    this.handleScore = t
    this.getCurrentTime
  }
```

```js
class Video extends Component {
  constructor(props) {
    super(props);
    console.log('Video constructor', props.playlist);

    this.handleEnd = this.handleEnd.bind(this);
  }

  handleEnd(event){
    console.log('video end', this);
    const videoReset = () => {
      const video = this.props.mediatype.state.mediaItemRef;
      video.load();
      video.play();
    };
    this.props.mediatype.handleEnd(videoReset);
```

```xml
<ximpel>
    <playlist>
        <subject id="Intro">
            <media>
                <video leadsTo="MenuDeBonteHen" repeat="true">
                    <source extensions="mp4" file="videos/Intro"/>
                    <overlay height="180px" leadsTo="MenuDeBonteHen" startTime="2" width="180px" x="1700px" y="890px"/>
                </video>
            </media>
        </subject>
```

Figure F.6: An example of how a XIMPEL playlist maps onto React components.

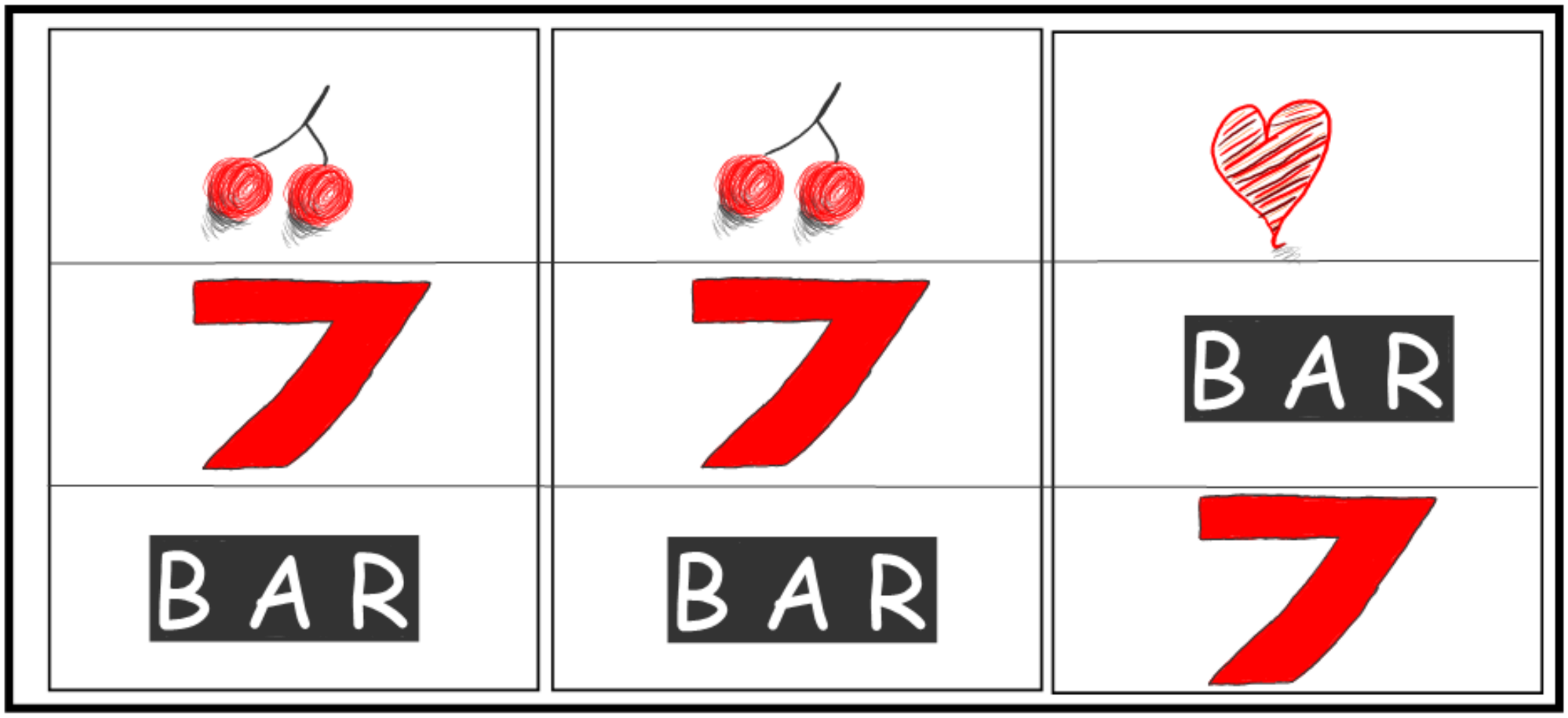


Figure F.7: A visual example of a near-miss. The lucky number 7 on the third slot machine column went one place further than intended.

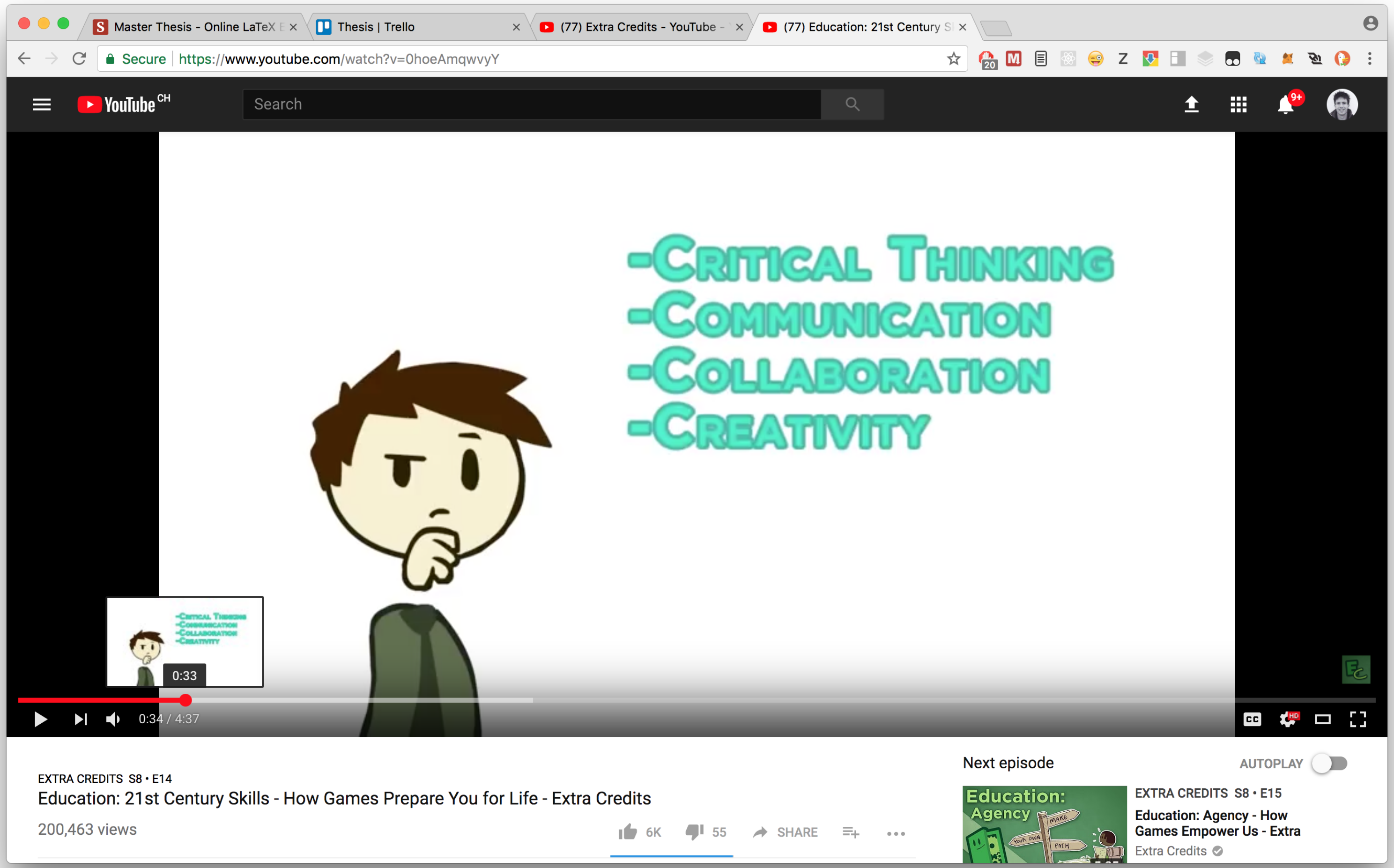

Master Thesis - Online LaTeX E
Thesis | Trello
(77) Extra Credits - YouTube
(77) Education: 21st Century S
Secure https://www.youtube.com/watch?v=0hoeAmqwvyY
YouTube CH
Search
-Critical Thinking
-Communication
-Collaboration
-Creativity
0:33
0:34 / 4:37
EXTRA CREDITS S8 • E14
Education: 21st Century Skills - How Games Prepare You for Life - Extra Credits
200,463 views
6K
55
SHARE
Next episode
AUTOPLAY
EXTRA CREDITS S8 • E15
Education: Agency - How Games Empower Us - Extra
Extra Credits

Figure F.8: An example of time scrubbing with a YouTube video. The red line could be dragged by the mouse to earlier or later times and resume playback there. The video itself is taken from a show called Extra Creditz, which talks about how game-design affects our lives for the better.

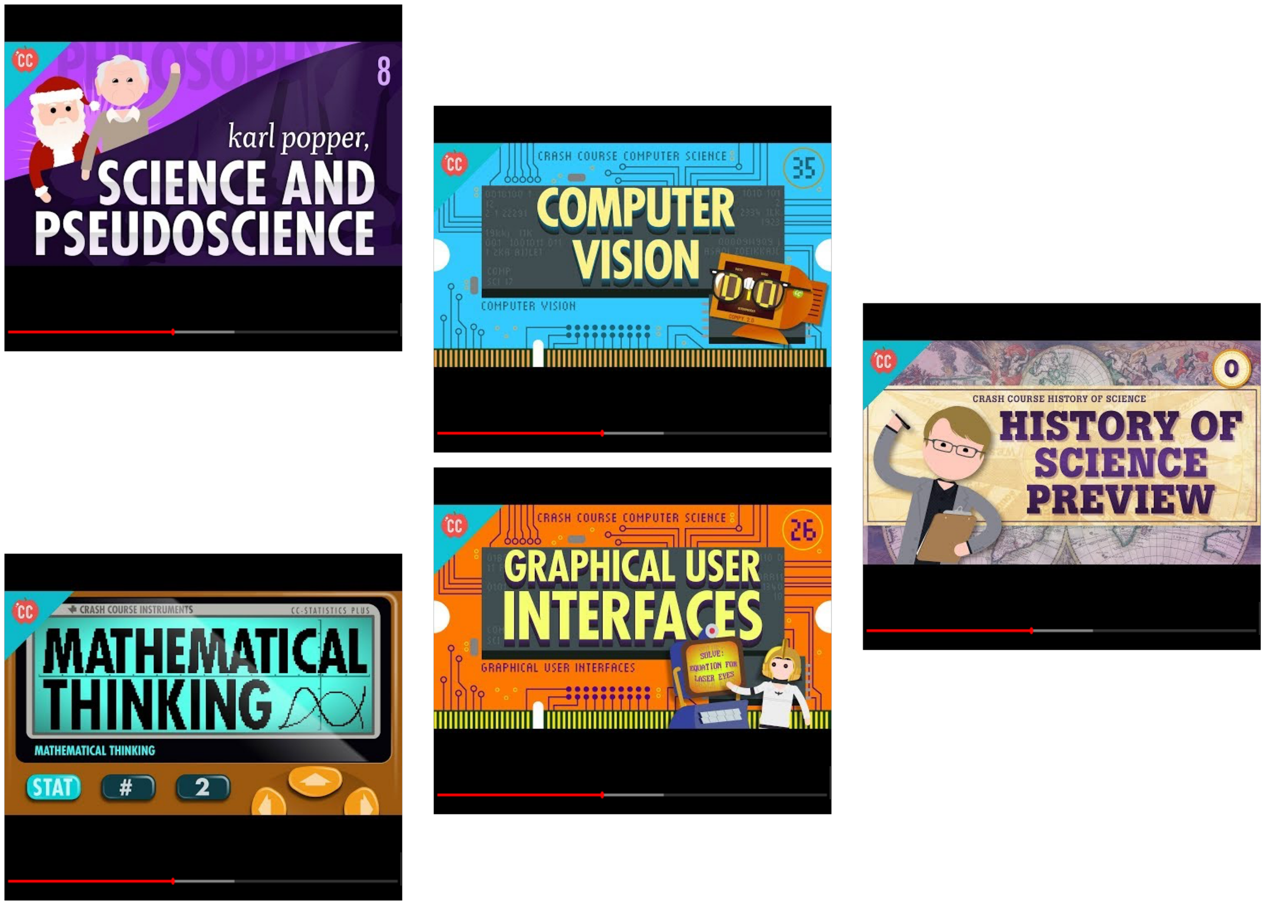

8
karl popper,
SCIENCE AND PSEUDOSCIENCE
CRASH COURSE COMPUTER SCIENCE
35
COMPUTER VISION
CRASH COURSE HISTORY OF SCIENCE
0
HISTORY OF SCIENCE PREVIEW
CRASH COURSE INSTRUMENTS
CC-STATISTICS PLUS
MATHEMATICAL THINKING
STAT
#
2
CRASH COURSE COMPUTER SCIENCE
26
GRAPHICAL USER INTERFACES

Figure F.9: An example of how local scrub bars look like. They are completely independent of each other, and it is unknown to each scrub bar that there are other scrub bars and at what time they are resuming playback. Image credit: thumbnails from the educational show Crash Course (on YouTube).

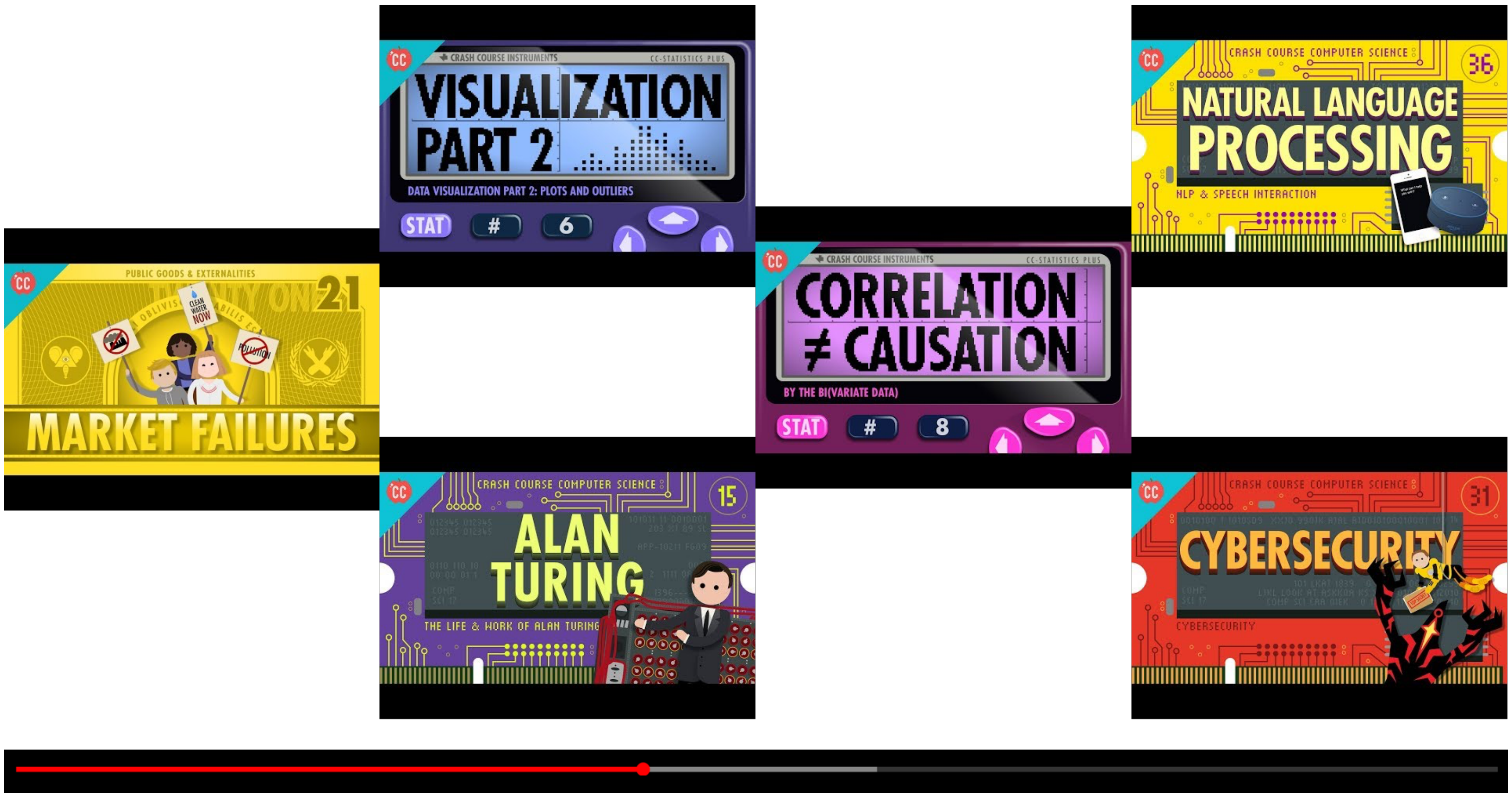

VISUALIZATION PART 2
DATA VISUALIZATION PART 2: PLOTS AND OUTLIERS
NATURAL LANGUAGE PROCESSING
NLP & SPEECH INTERACTION
MARKET FAILURES
PUBLIC GOODS & EXTERNALITIES
CORRELATION ≠ CAUSATION
BY THE BI(VARIATE DATA)
ALAN TURING
THE LIFE & WORK OF ALAN TURING
CYBERSECURITY

Figure F.10: An example of how a global scrub bar looks like. One scrub bar manipulates the time for all media items in the presentation. Image credit: thumbnails from the educational show Crash Course (on YouTube).

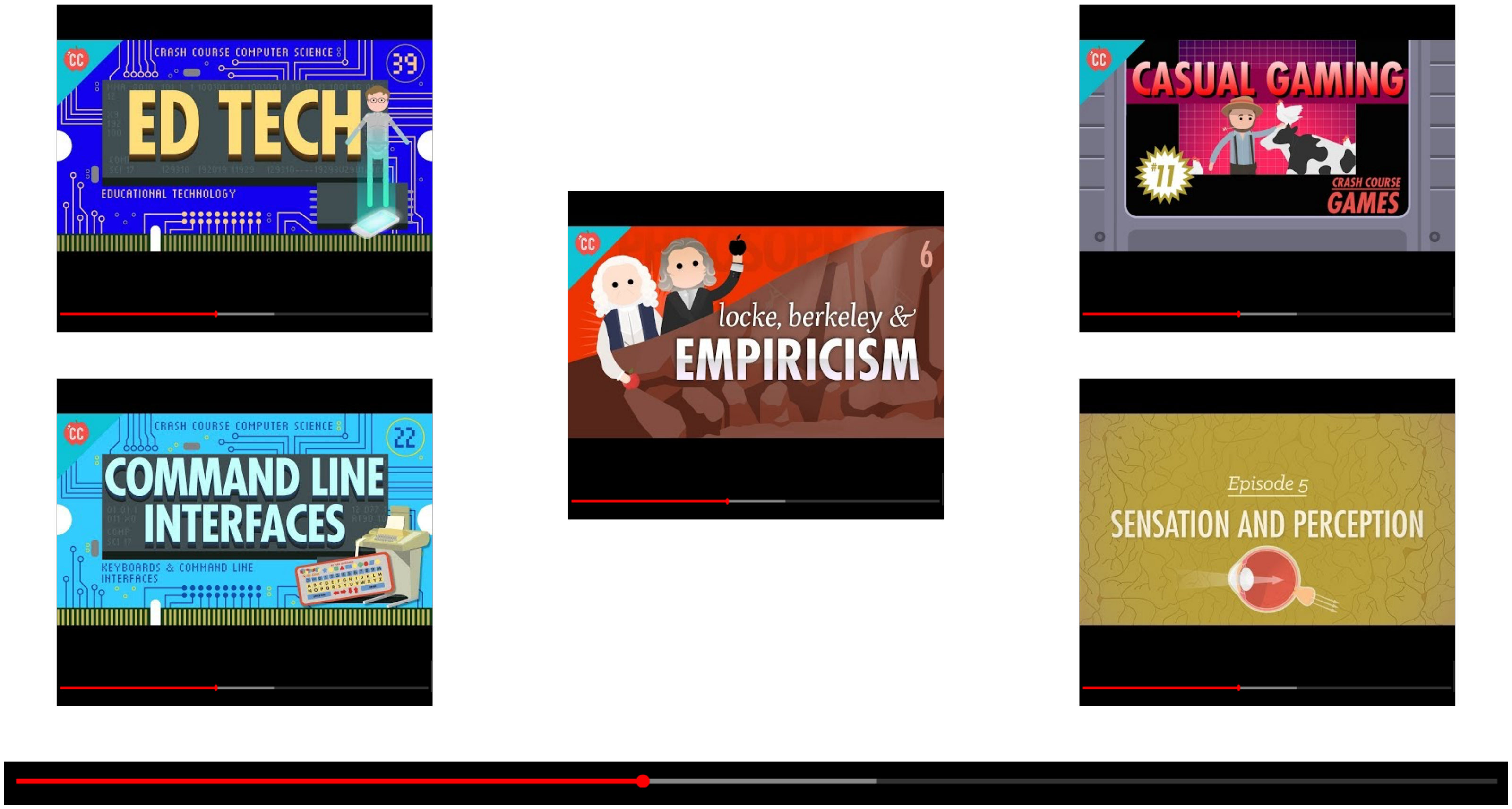


Figure F.11: An example of how a global scrub bar looks like with local scrub bars. The global scrub bar manipulates the time of the local scrub bars but the local scrub bars can also be manipulated individually. Image credit: thumbnails from the educational show Crash Course (on YouTube).

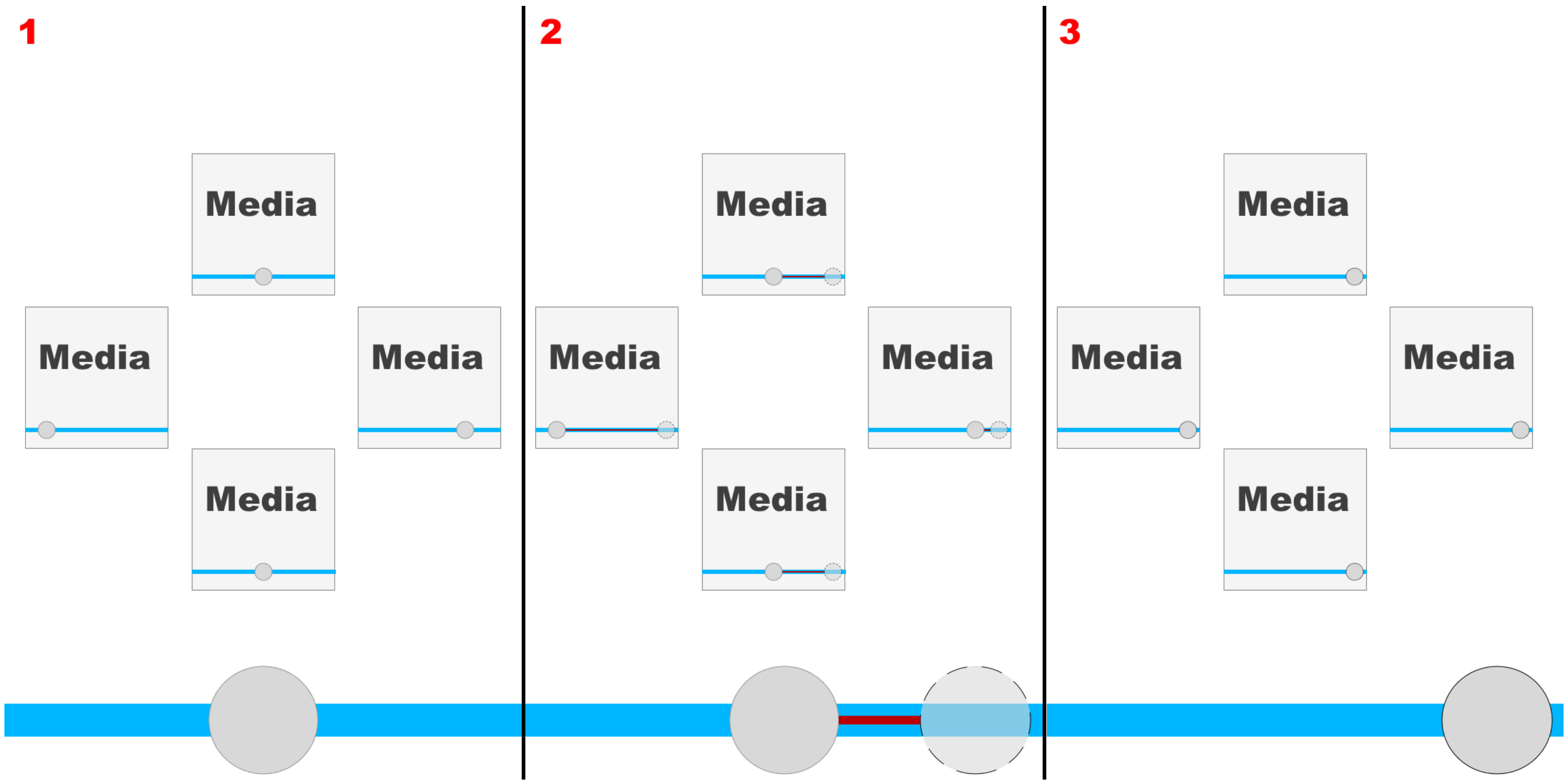


Figure F.12: The global time scrub bar determines where all local scrub bars will resume playback.

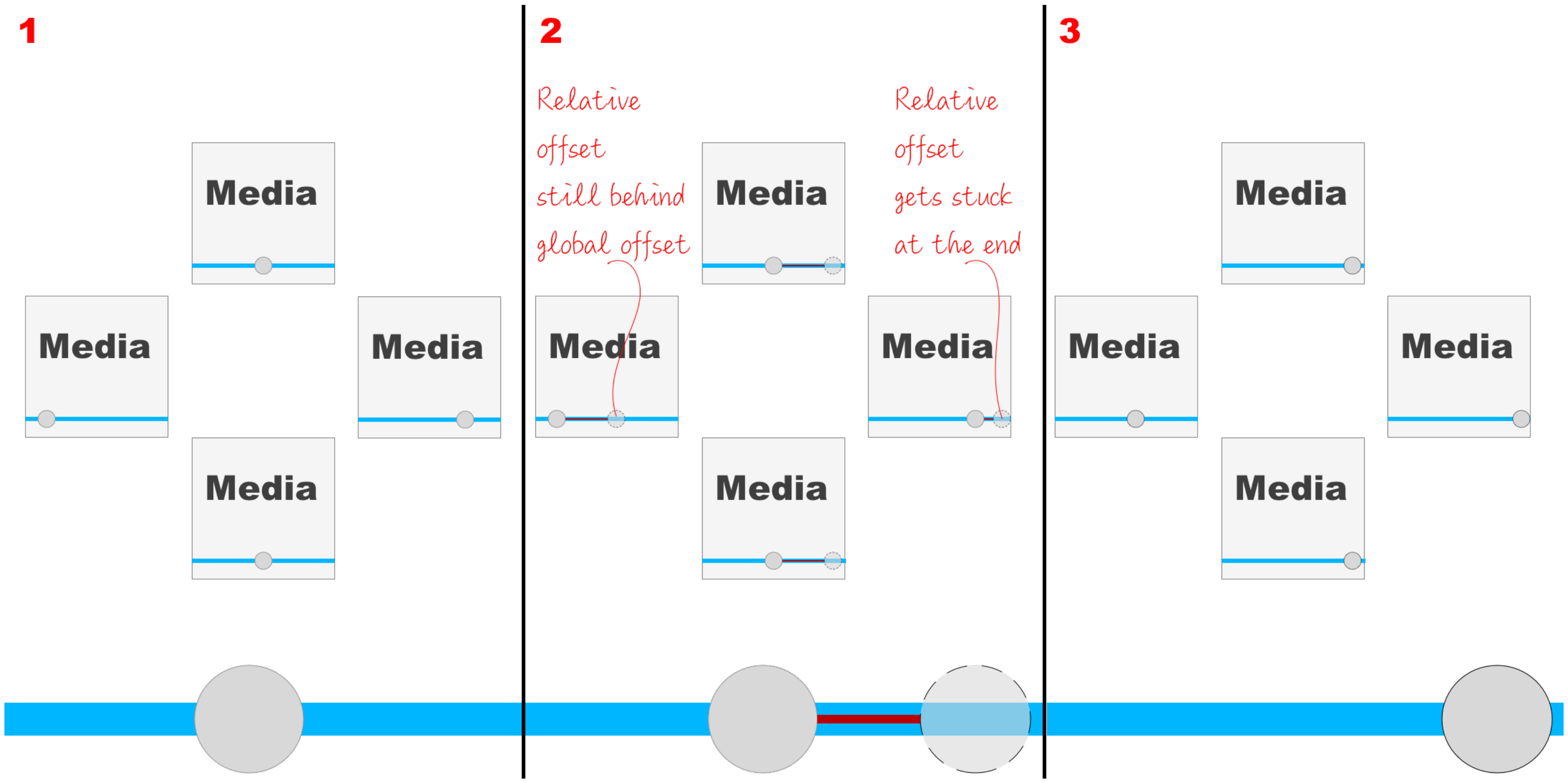

1
2
3
Relative offset still behind global offset
Relative offset gets stuck at the end
Media

Figure F.13: The global time scrub bar determines the percentage offset that local scrub bars will give to their position in time.

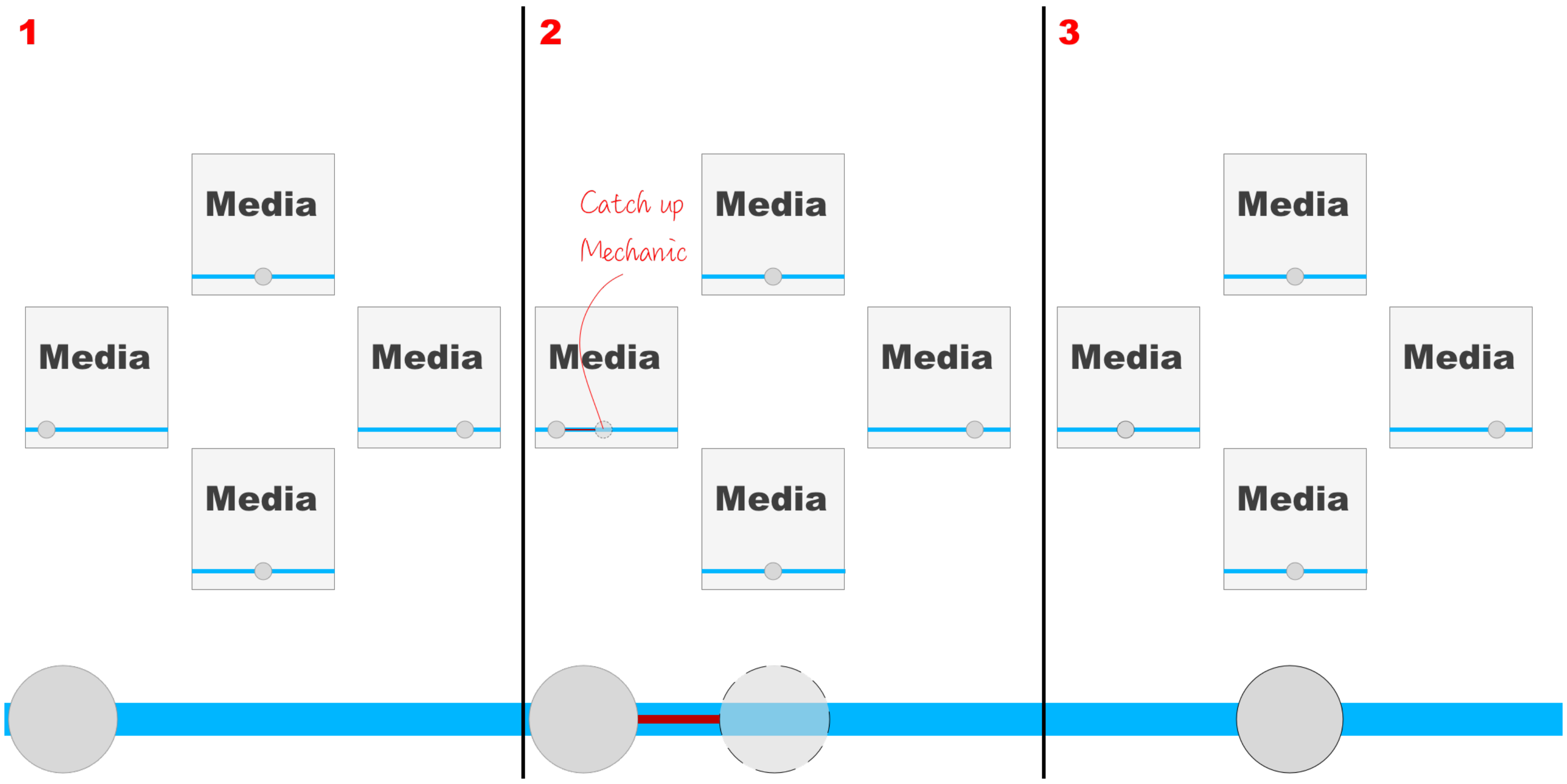


Figure F.14: The local scrub bars catch up with the global time scrub bar, but only if the local scrub bars are behind the global time scrub bar. There is also an inverse situation possible, here the local scrub bars will catch up to the global time scrub bar if they are ahead. This inverse situation is not shown in the figure. In the figure only one media item needs to catch up with its local scrub bar.

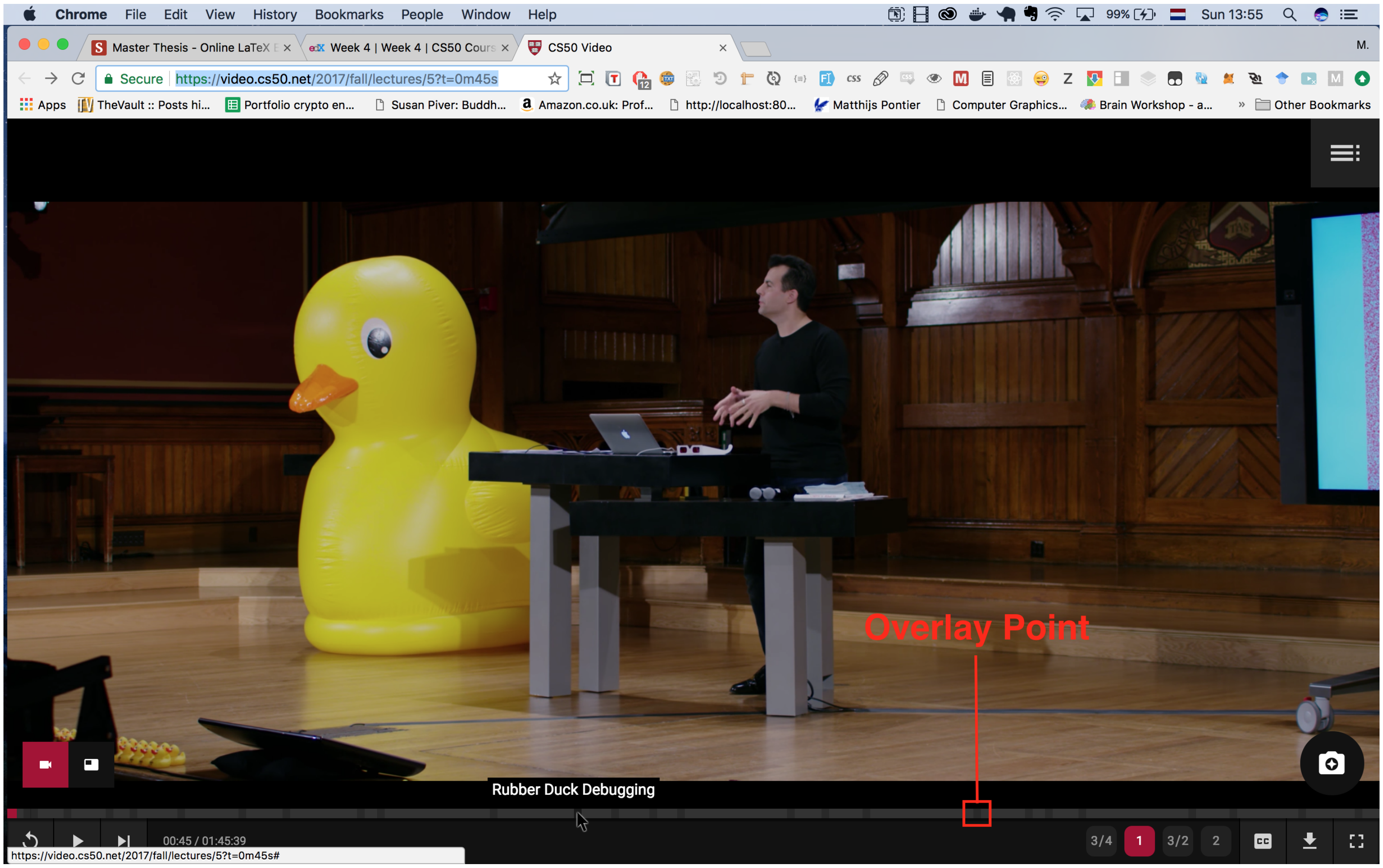


Figure F.15: An example of the CS50 player. In this screenshot the mouse hovered over an overlay point called Rubber Duck Debugging.

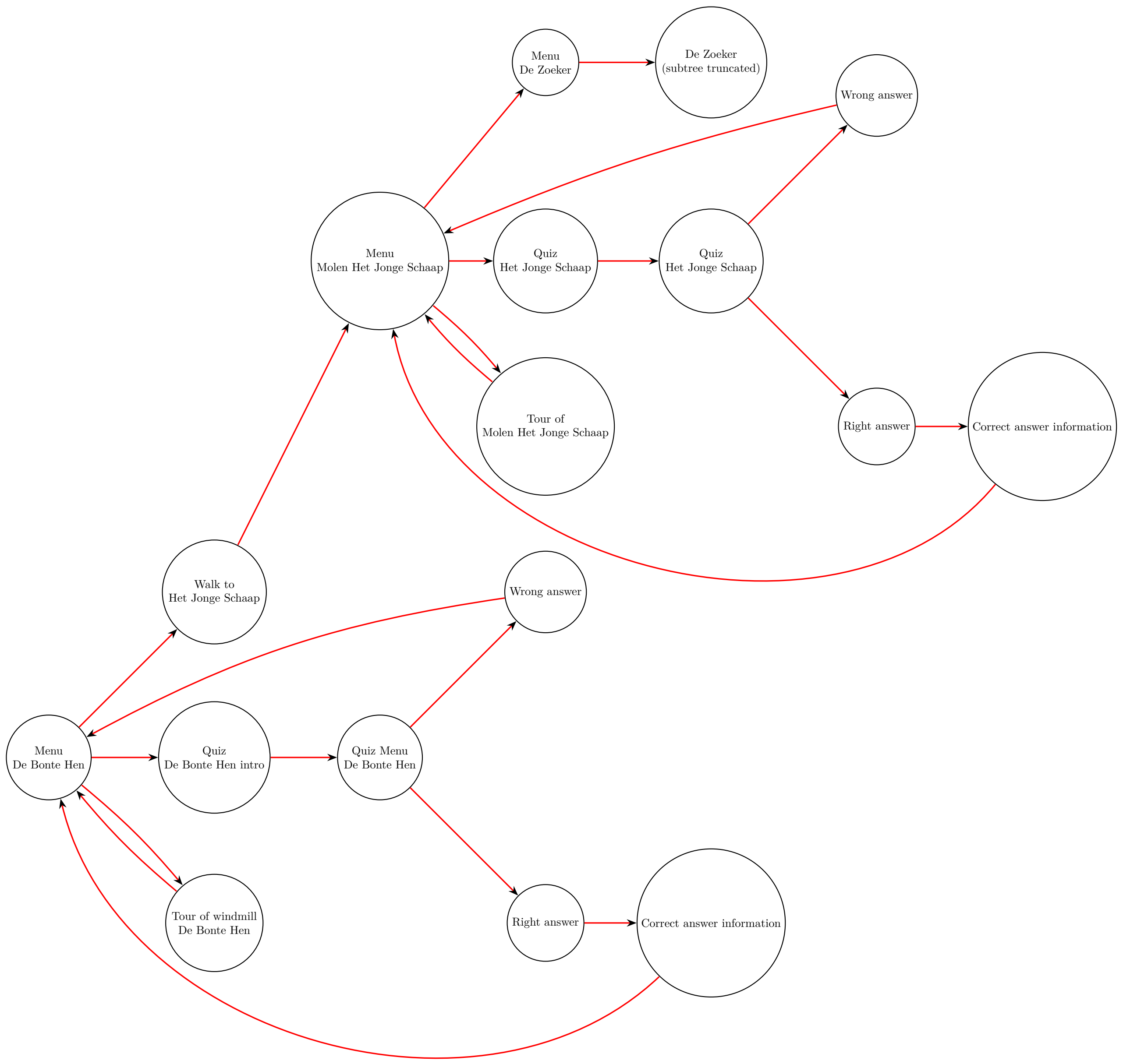


Figure F.16: In this figure a part of the full interaction graph of the XIMPEL presentation of the Zaanse Schans is shown. Users who would like to scrub between subjects could see a graph like this, and click on any part of the edge to indicate how far they want to start within a subject. A graph like this could be shown in the upper right corner by clicking on a button, for example. What is not shown in this image is that overlays could be displayed on any part of the edge as well for additional scrubbing information. As one could imagine, the figure would become too crowded.

Table F.1: Summary XIMPEL JS versus XIMPEL React in XML-parsing and storing it in the in-memory configuration object.

XIMPEL Module	LoC XJS	LoC XR	Time XJS	Time XR
Parser	812	15	N/A	8 hours[a]
In-memory configuration object	366	0	N/A	0[b]

Table Notes
[a] debugging in Webpack is tough, hence 8 hours for 15 lines of code.
[b] 0 lines of code for in-memory configuration object for XIMPEL React, because they are handled by props which is a feature of the library.

Acronyms
LoC: Lines of Code
XJS: XIMPEL JS
XR: XIMPEL React
N/A: Not Available

Table F.2: Comparison between all XIMPEL JS files and all XIMPEL React components.

XIMPEL JS	XIMPEL React Counterpart
Analytics.js	Analytics.js (same library)
MediaTypeRegistration.js	React props and error message handling is designed away
Models.js	React props and `Overlay.score`
Parser.js	Webpack XML parser
PubSub.js	PubSub.js (another library with the same name)
QuestionManager.js	Designed Away
XimpelApp.js	Handled by Webpack compilation[a], playback functionalities are designed away
polyfills.js	Cross-browser compatibility is done by React
ximpel.js	Partially React and partially designed away
Audio.js	`Audio`
Filler.js	`Filler`
Iframe.js	`Iframe`
Image.js	`Image`
MediaType.js	`MediaType`
Message.js	`Message`
Terminal.js	`Terminal`
TextBlock.js	`TextBlock`
Video.js	`Video`
YouTube.js	`YouTube`
MediaPlayer.js	`MediaType`
ParallelPlayer.js	`Media`
Player.js	`Playlist`, `Rule` and `Subject`
SequencePlayer.js	`Sequence`
OverlayView.js	`Overlay`
QuestionView.js	Designed Away
View.js	Not needed because QuestoinView.js is designed away
XimpelAppView.js	Designed Away

[a] It does mean we cannot retrieve the playlist from a server, it needs to be locally included.