



CRAFTING EFFECTIVE USER INTERFACE ANIMATIONS

Understanding the complexity of crafting user
interface animations on web platforms

Yonatan Wolowelsky

Bachelor's thesis

August 2015

Degree Programme in Media

TAMPEREEN AMMATTIKORKEAKOULU

Tampere University of Applied Sciences

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Media

WOLOWELSKY, YONATAN
Crafting effective user interface animations

Bachelor's thesis 53 pages, appendices 8 pages

August 2015

The purpose of this thesis is to cover widely the area of user interface animations in different lights and perspectives, in order to fully understand the complexity interactive producers, such as designer and developers, have to deal with in order to create good, well thought and well designed animated experience.

The thesis is consisted of an introduction, 4 research chapters and final conclusion chapter. The first research part is an overview of user interface animation usage. The second part focuses on the basic animation principles in a user interface context. The third chapter outlines the results and analysis of a user behaviour research composed especially for this thesis, and the fourth part describes the complexity of crafting user interface animation from a technical perspective.

The thesis combines collection of research methods in order to cover a wide range of aspects. While in the first and seconds chapters the main research method was to collect information from variety of sources in order to bring a wide overview to the reader, the third and the fourth chapter are using original research methods such as experimenting and testing, in order to dig deeper and put some of the statement from previous chapters on a practical trial.

Keywords: animations, user interface, user experience, human behaviour, web design, web development.

ABBREVIATIONS AND TERMS

UI	User Interface
UX	User Experience
Bounce Rate	The percentage of visitors to a particular website who navigate away from the site after viewing only one page.
Keyframe	Defines animated values for a specific point in animation's timeline
Bézier curve	Parametric curve frequently used in computer graphics
ms	Milliseconds
MHP	Model Human Processor
GIF	Graphics Interchange Format
CSS	Cascading Style Sheet
SASS	Syntactically Awesome Stylesheets
HTML	HyperText Markup Language
SEO	Search Engine Optimisation
WebKit	Open source web browser rendering engine
JavaScript	Programming language used primarily for web products
JS	JavaScript
jQuery	A popular JavaScript library
Velocity.js	A modern JavaScript library for creating web animations
CPU	Central processing unit

CONTENTS

1. Introduction	6
2. The role of animations in the user experience flow	8
2.1 Use cases of user interface animations	8
2.2 Avoiding overuse and misuse of user interface animations	9
2.3 Storytelling with user interface animations	10
2.4 Animation as user guides	11
2.5 Animation branding	12
3. Classic animation principles in a user interface context	13
3.1 Squash and stretch	13
3.2 Anticipation	14
3.3 Staging	14
3.4 Straight ahead and pose to pose	15
3.5 Follow-through and overlapping action	16
3.6 Slow in and out	18
3.7 Arcs	20
3.8 Secondary action	20
3.9 Timing	22
3.10 Exaggeration	26
3.11 Solid drawing and appeal	27
4. Experiment - The influence of UI animations on user reaction and task fulfilment speed	29
4.1 Experiment overview	29
4.2 Experiment results	33
4.3 Experiments conclusions	36
5. UI animation in a technical context	38
5.1 Overview of web animation methods overtime	38
5.1.1 The dawn of the Internet	38
5.1.2 The Flash and DHTML era	39
5.1.3 The HTML5 era (post-Flash Era)	42
5.2 Comparing contemporary web animations solutions	42
5.2.1 Usage	43
5.2.1.1 CSS animations and transitions	43
5.2.1.2 jQuery.js Animate	45
5.2.1.3 Velocity.js	46
5.2.2 Performance	48
5.2.3 Compatibility	49

6. Discussion and conclusions	52
List of references	54
Appendices	56
Appendix 1. Model Human Processor	56
Appendix 2. Task fulfillment speed experiment specs and details	57
Appendix 3. CPU Usage test specs	61

1. INTRODUCTION

The past couple of years I have been working in several companies as a hybrid front-end developer and user experience designer. This gap between product design and product implementation is an absorbing position for me. I was lucky enough to get many opportunities to learn how the product's cycle-process works, from conceptualizing and design, through prototyping, implementation, testing and back again to design.

One of the biggest challenges in my work is designing and implementing user interface animations. This task could be fascinating and frustrating at the same time. Simply because like most of the sub-topics in user experience design, it is involved enormous amount of time dedicated to trial and error.

The complexity of crafting a good user interface animation is not really about the technical tools or the best practices methods, though they can help a lot to produce higher quality products and save some precious time. The biggest objection when dealing with animations in a user interface context, is to make them make sense. Making sense in animation means that the animation is not necessarily stunning and visually attractive, though it is highly important that it will be one. Making sense means that the animation crafted is valuable to the user.

Animation helps the user to understand the product, it can help user experience designer to tweak the story and make the storyline clearer to the user. Animation can draw user attention to the important parts of the product or guide him through the product's user journey. There are many parameters that needs to be considered, both from visual and conceptual perspectives, when planning the animation flow and placement and even during the tedious implementation process.

This article consisted of 4 major research chapters, an introduction chapter and conclusion chapter.

The first research part is an overview of user interface animation usage (and misuse) in the wider user experience context - usages, flow, goals along with storytelling and branding.

The second part goes deeper into the principles and which have to be considered while designing an animation and the user interaction with it. As when one thinks of the verb “to craft”, the first thing comes to mind is physical, handmade crafting, the best way to go through these crafting principles, would be by referencing the old school classic animation principle defined by Disney animators in their book “Illusion of Life”.

The third part is an online-experiment I have created and moderated with 207 participants from 28 different countries. This experiment is a practical test on a few statement which were brought up by different sources in the first chapters.

Last but not least, the third part is dealing with the technical perspective. For me it was really important to bring some technical context to the thesis work, as I believe design and development should not be separated from each other. This chapter overviews the history of animation in web platforms and compares the most common ones used today in terms of performance, developer-usability and compatibility.

The purpose of this thesis is to cover widely the area of user interface animations from various lights, in order to fully understand the complexity developers and designers have to deal with while crafting good, well thought and well designed animated experience. My goal was to bring a healthy mix of theory and practicality to this thesis, as I believe this is the core idea of the Interactive Media Program.

2. THE ROLES OF ANIMATIONS IN THE USER EXPERIENCE FLOW

Based on my experience, product managers love animations. They think animations are “cool” and generate traction and traffic to their product. Often they would encourage the front-end team to create as much animation as possible. Some of them would use a real user studies as a practical argument, and some of them would just claim that animations will “improve the user experience”, without really understanding what does it mean.

This is not untrue. However, animation should be thought through, well-planned, and carefully crafted. This chapter describes the roles of animations in the general user experience flow, while looking at animation from perspective of assisting to the general UX flow, to the storytelling and to the branding.

The inspiration for this chapter came from a presentation made by Mariusz Cieřła in Berlin 2014, called “Making animation make sense”. (M. Cieřła, Berlin. 2014)

2.1 Use cases of user interface animations

Before we start describing usages and examples for user interface animations, we need to understand the basic role animations fulfill in user experience design.

First of all, animations are helpers. They help us, the users, to understand each feature in our interactive product. In order to do so, animations use real-life references to describe a virtual process, action, or pieces of information.

A simple example would be a drop-down animation. When a user triggers a drop-down element by clicking on a button, the drop-down, which was hidden prior the user action, is now appearing on the screen. In real life, a new visible object might fall, move, slide or scale, but no where in our physical world objects simply appear out of the blue. That is precisely the gap animation fills between what the computer does - showing/hiding the drop-down element, and what users understand - where is the drop-down coming from and what is it related to.

Nowadays, animation are used widely in many types of devices and screens. The mobile phenomenon, which kicked off around 2007 with Apple’s first-generation iPhone,

boosted up the usage and awareness of the importance of motion in user experience design.

Mobile phones, or essentially any touch device that can be literally carried and used with one hand only, are a great example for the role of animations and the gap they fill.

Since mobile devices can be rotated, shaken or even thrown away, and furthermore as these devices are using direct touch interface and not a remote interface (such a mouse or keyboard), they feel more like “real” objects rather than virtual data floating in a parallel universe.

Because of their touch interface, mobile devices which provide the user an interface that lacks animations, feels rather dull, gloomy and unreal. A user who uses a mobile device that is not using basic transitions on its interface, could be compared to situation when a child is playing with a broken toy car. It might be working, but it is really not fun.

2.2 Avoiding overuse and misuse of user interface animations

Animation is a great method for connecting the user to the product and a great user experience practice to make the users feel comfortable using the product. However, animations can easily become an overkill. Misusing, overusing or overdoing of animations in user interfaces can easily lead to user frustration. From marketing perspective, these user emotions could easily be translated into analytical numbers like high bounce rate or low conversion rate.

Before crafting user interface animations, we need to think about the meaning and the goal of our the animation. According to the research group Nielsen Norman Group’s article, there are three main goals for using animation in a virtual interface (A. Bedford, 2014):

- **Attracts users attention.** Animations attract user's attention to changes in a state of an object or for a new information appearance. As animation are doing a great job with drawing user’s attention, when misusing animation, we can easily draw

user attention to something that is not important or far worse, to distract the user from his original task.

- **Shows continuity in transition.** As mentioned previously, animation are very handy with filling the gap between two states of an object. This type of animations are also called transitions, which are ideally leading the user from state 1 into state 2 smoothly and without further interference. This role can be easily ruined by an overdone animation, when a transition contains more than two keyframes, too many animation attributes, or involving variety of unrelated objects in the overall animation.
- **Indicates relations between objects.** Animations help users to visualize a map of relations between objects. Going back to the drop-down example from the previous chapter, the animated object scales towards the the bottom of the screen and changes its position slightly from the top to the bottom. This visual illusion is manipulating the user to think the drop-down was hiding behind a parent button, and only when the user triggers it, it will be sliding out. This way, we are essentially illustrating the relationship between the parent button and the drop-down child.

Yet again, poor animation can easily confuse the user about the hierarchy of objects in our product. Not using the right animation attributes, for example sliding the drop-down from bottom to top, or from the edge of the screen into a different unrelated position, can mislead the user and distract him from a smooth usage flow.

Crafting a good user interface animation is crafting animation with meaning. The top goal of animation is to provide users a smooth flow and experience, to raise their attention level and to keep them focused on the task they are doing. In other words - animation should be designed to lead and not necessarily to impress.

2.3 Storytelling with user interface animations

Every product is a story. It does not matter if the product's goal is to sell clothes from reindeer fur, to bring attention to Amnesty International's activities in developed

countries, or to promote a political figure's campaign - the story will be always the base of your user experience concept, flow and implementation.

When referring to digital products, the story is usually the process the user go through while using the product. The beginning of the story would be the first time the user interacts with the product, however the ending may vary and dependent on the user character and personal desires. The user experience designer role is to tell this story to the user and guide him through the possible continuations and endings.

Animations help the users to follow the product storyline - what solution the product brings to the user, and how the user can use it the best way - and are a great method to guide and navigate users through the designed user flow, in order to bring them eventually to the story's happy ending.

2.4 Animations as user guides

Often when we see inexperienced users, such as children or elderly people, we can see a variety of facial expressions that symbolised confusion or surprise. From user experience perspective, this usually means that something in the flow is broken or not smooth enough, and therefore requires fixing. When these facial expression become visible, it often implies that the user stops his original task for rethinking the situation and considering what to do next.

In many cases as such, animation can be a great intersection to connect different interaction, actions, events or just static objects, in order to avoid user confusion. This brings us back to the principle of showing continuity - where transition fills the gap between two states of an object - but not only that. Animations attract user's attention to an alert, tell our users that they might improve the way they approach their task or warn them about the next to come.

2.5 Animation branding

A user interface animator must not forget that above all, animations are visual elements. Animations are part of the product's visual style as much as the color scheme, the fonts, the elements sizing rules or the grid system.

Animations, just like every other visual element, define brand. If an animation team decides to go with jumpy-squashy-styled animation, they should know that by doing so, they define the product's brand not less than choosing a childish handwriting font. In contrast, if the animation team decides to implement a subtle transition with not much of a shape or size changes, they should keep in mind that they have just branded their product as much as they would have chose a monochrome color scheme.

Looking at animation from branding perspective also means that all of the animations in the product should be consistent and following the same visual language and style. Each animation preset that is being crafted, should be compared and tested in the global product's branding scope. All the animations in the product, and in the brand sub-products in general, should live with each other in harmony together in order to create a composition that is stunning, beautiful and also makes sense, both from visual perspective and from user-flow perspective.

3 CLASSIC ANIMATION PRINCIPLES IN A USER INTERFACE CONTEXT

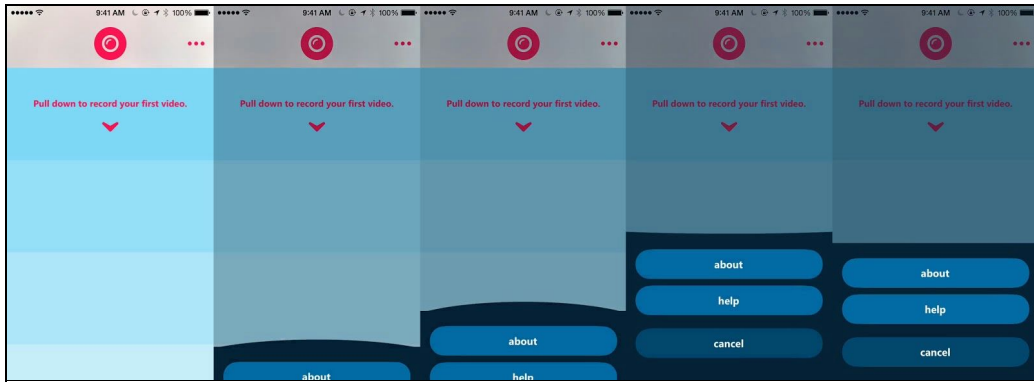
Even though user interface animation needs are very minimal compared to classic animations, such as full length Walt Disney animation feature films, there is still a lot that could be learnt from the old-school animation principles and applied in a modern context of user interface.

In 1981, Disney animators Ollie Johnston and Frank Thomas wrote a book called “Disney Animation: Illusion of Life” (*O. Johnston, F. Thomas. 1981*). What was later referred by many as the “Bible of Animation”, included a set of 12 basic animation principles that even today, after 32 years, are surprisingly relevant for both classic and digital animators. The following chapter will briefly go through some of these principles and try to evaluate how to practice them in a user interface context.

3.1 Squash and stretch

“People and objects have an inherent mass. When an object moves, the quality of the movement often indicates the rigidity of the object.” (*R. Hinman, 2012*)

In real life, some object materials like solid wood or cement, would feel harder and less flexible than softer materials like rubber or foam. By defining the amount of squashing and stretching in the animation, we are essentially defining its character and personality. We tell our user whether the animated object is jumpy and playful, hard and solid or even if it is soft and dreamy. This attribute, and the principle behind it, has a huge impact on the product general style and the brand behind it.



PICTURE 1: “Skype Qik” mobile app screenshot. The mobile menu is squashing and stretching, giving the user a playful feeling. (capptivate.co, 2015)

3.2 Anticipation

When designing an animation, we can distinct the process into three different phases: Preparation, action and result. The anticipation principle referring to the preparation phase, when we give the user a clue about what is going to happen next. (R. Hinman, 2012)

Triggering emotions and feelings is an essential part of user experience design, it makes interactions become live and humanizing the product’s content. When anticipation in animation is done in a right way, we make the user feel thrilled and excited, even for a few milliseconds, and by doing so we achieve a great challenge of triggering these emotions while people are using the product.

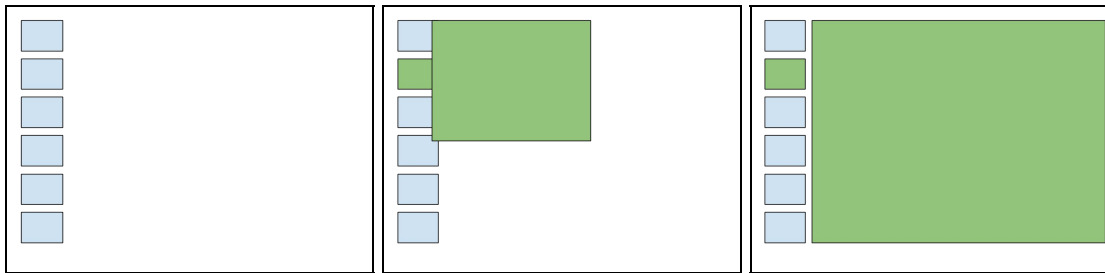
3.3 Staging

In classic animation, the principle of staging used to define the context of the surrounding of the scene. Poor staging effects the the animation’s storytelling directly and could lead to viewers misunderstanding the situation, the story or the scene.

When arguing about the role of animation staging in user interfaces, some would say that the screen itself is the animation’s staging - as nowadays the user can interact with the product on a variety of different devices and screen sizes (mobile, desktops, tablet, TV) and in many ways (touch screen, mouse, keyboard). Those parameters should be taking into account when designing an animation stage, but not only. On every device and

screen size, it is important to give the a context for what is happening and why is it happening.

For example, when user clicks on an image from a list of images in order to open it, the state of the image changes. After the click, the image gets bigger and includes more details, sometimes with a caption text. When staging this simple interaction, the designer must think of the list of images as the anchor point for the big image item, in order to help the user to understand the relativity between the two elements.



PICTURE 2: When choosing an item from a list, the list is used as an anchor point for the animated object in order illustrate relativity. (Wolowelsky, 2015)

3.4 Straight ahead and pose to pose

Straight ahead and pose to pose are two different strategies referred to way an animator would draw the animation.

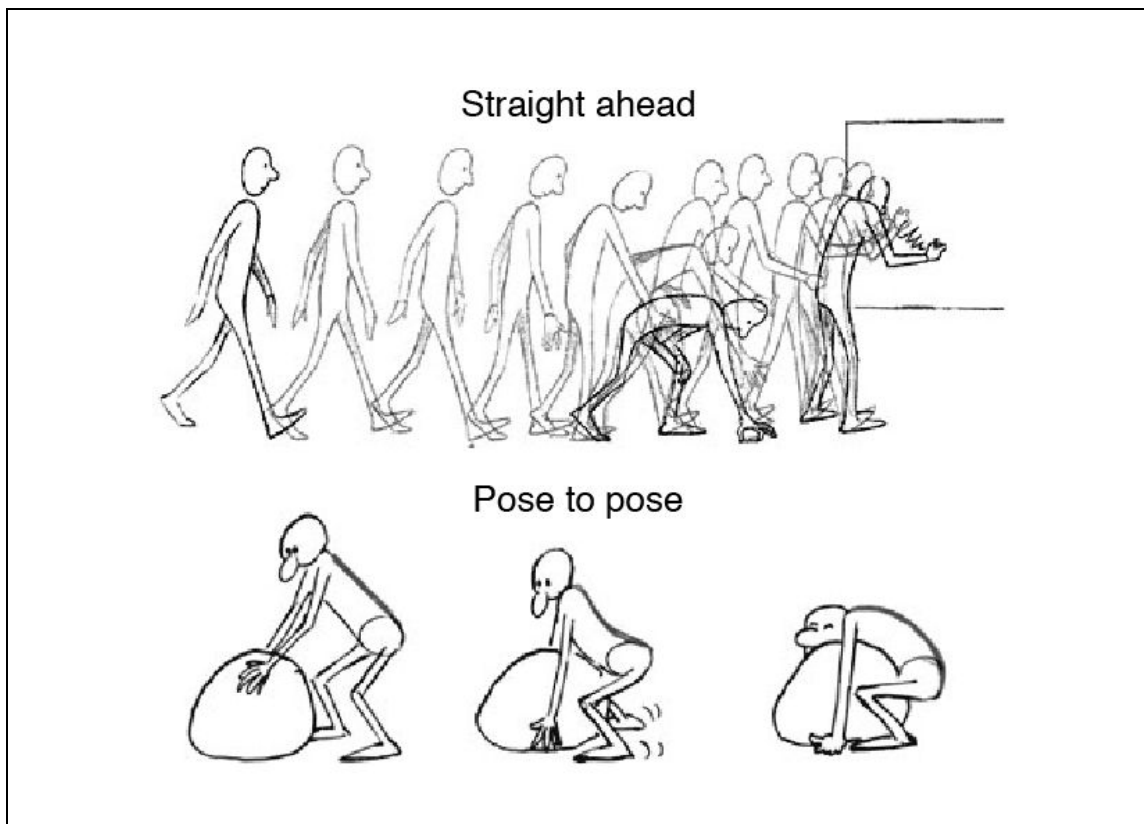
In the “Straight ahead” approach, the illustrator would draw every single frame in the animation, while in “Pose-to-pose” the animation drawings process would involved only a few main keyframes, and the computer, or a clean up artist, would figure out the rest. This two concepts could be interpreted in the world of user interface design to differentiate the need of a simple two-keyframed transition, and a complex multi-keyframes animation.

The vast majority of user interface animations are using the “Pose-to-pose” approach, as most of these animation are in fact transition - a simple animation which contains only two keyframes that usually crafted in order to help the user understand an interactive event or to illustrate a change of state in a visual element.

Using the “Straight ahead” approach in a user interface context would normally mean crafting an animation which contains more than two keyframe, and therefor is no longer a

transition but a proper animation. A good example for taking this complex approach would be in game design, when the user interface animation is sometimes mixed up with the game and characters animation, and therefore requires a deeper thinking into the game's and character movement.

The question raised while taking the “Straight ahead” approach would be, if these animations are indeed user interface animations or are they decorative animations inside a user interface layout.



PICTURE 3: An Examples of classic “Straight ahead” animation with multiple keyframes and a “Pose-to-pose” animation with only 3 main keyframes. (R. Hinman, 2012)

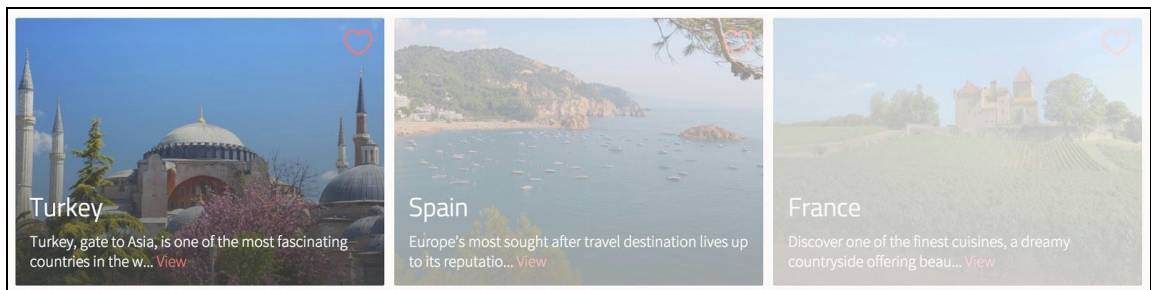
3.5 Follow-through and overlapping action

“While anticipation has to do with the preparation of an action, follow-through involves the end of an action. Follow-through and overlapping are more commonly used in character animation for body movement, for example when arms swaying as the character drops them or long hair falling.” (R. Hinman, 2012)

In physical world, actions and movements rarely come to a sudden and a full stop, but rather would end gradually. “Follow-through captures how parts of an object continue to move even after other parts of the object have stopped moving.” (R. Hinman, 2012)

Overlapping action, on the other hand, is the animation principle that describes how parts of an object moves at different speeds and rates. Capturing the movement, as well as the slight variations in timing and speed of these parts, makes the objects feel a lot more natural. As an action never come to an absolute stop before another action has began, the overlapping action maintains a flow between self-contained phrases of the movement.

Follow-through and overlapping action can help user interface designers define and communicate the relationship between user interface elements. A good practice to illustrate these principles would be to apply a “one-by-one” transition. For example, when a grid layout of boxes appears on the screen, the boxes could be animated one by one, with a tiny delay between them, in order to show relativity and bring continuous feeling to the whole transition.



PICTURE 4: Example of “one-by-one” animation I have created while working at everyglobe.com. One by one, the image containers are fading in on the grid, illustrating relationship and order while bringing natural and continuions feeling to the general user experience flow. (Wolowelsky, 2015)

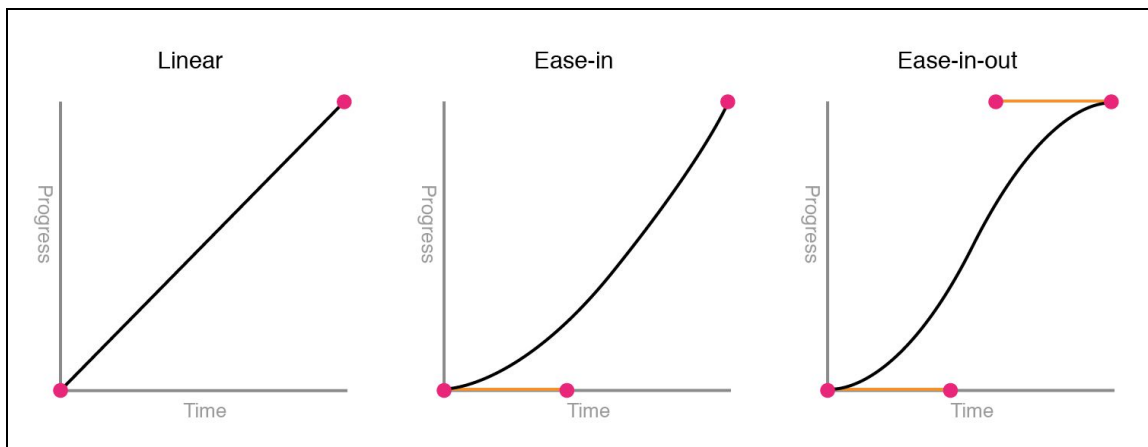
3.6 Slow-out and slow-in

In the real life, things rarely move at a linear speed. When a car starts moving, its speed is always starting from zero kilometers per hour, and gradually speeding up. Same applies when the car stops, it never simply stops. Even in an unexpected break, the car's speed always changes gradually.

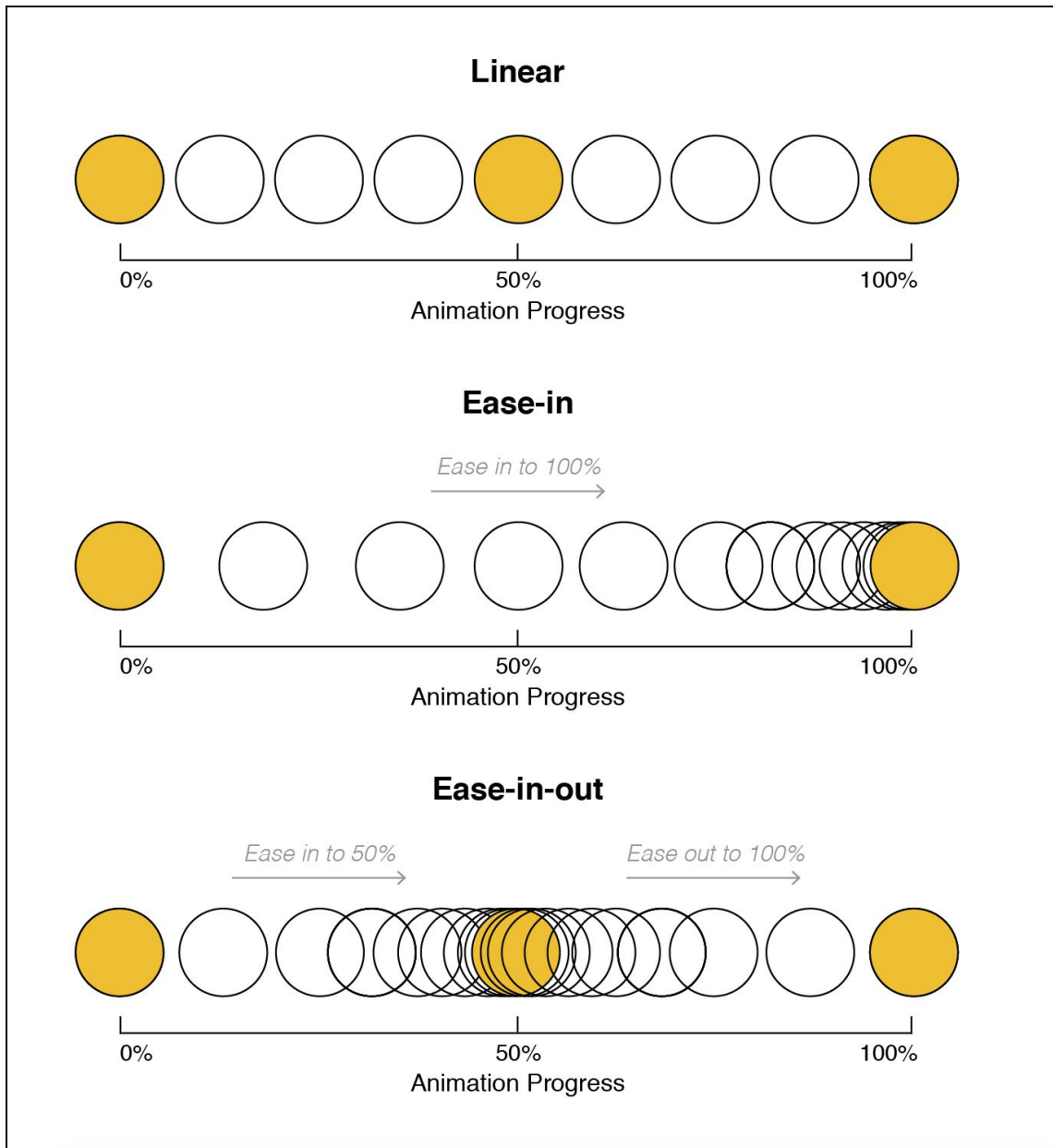
As said already before, animations use real life references to describe a virtual process. In order to be realistic in animations, designer must have to take into considerations the laws of physics. A good practice to apply these realistic-looking UI animation, designers and developers could apply timing function to their animations.

Timing function attribute specifies the speed curve of an animation. The animation speed curve is a Bézier curve, which is often used in vector graphics as well, and formed by four points plotted on a graph. (Greig, 2014)

By declaring timing function, developers guide the machine about the ratio of the visual attribute progress in comparison to time.



PICTURE 5: Bézier curves used to form 3 timing functions presets - “Linear”, “Ease-in” and “Ease-in-out”. (Wolowelsky, 2015)



PICTURE 6: “Onion skinning” graph of the animation progress, illustrating the 3 timing functions presets described in picture 5 - “Linear”, “Ease-in” and “Ease-in-out”. (Wolowelsky, 2015)

The graphs in picture 5 and picture 6 represent three common timing functions. For example, when analyzing the “Ease-in-out” graph, we can see that the animation starts slowly, gradually speeds up in the middle, and at the end it slows down again while being symmetrical in speed to the animation’s beginning.

In contrast to the “Linear” timing function, when the relation between progress and time doesn’t change, the “Ease-in-out” timing function, together with other timing function

presets such as “Ease-in”, or “Ease-out”, give the animation the feeling that it is not just simply moving, but it is also alive and real.

3.7 Arcs

“Objects don’t move through space randomly. Instead, they move along relatively predictable paths that are influenced by forces such as thrust, wind resistance and gravity.” (R. Hinman, 2012)

The arcs animation principle refers to each object’s trajectory. Similar to “Follow through” and “Slow-in slow-out” principles, arcs could refer to the laws of physics which dictate the object path and movement in space.

While these paths are mainly virtual and unseen by the user eye, trajectories patterns exist and based the objects’ character or material. Mechanical objects, such as vehicles - cars, bicycles or trains - would move along straight trajectories, while organic objects, such as plants, people and animals, would move along arched trajectories. This principle refer to the rule that animated object should reflect these characteristics for better realism and naturalism.

Just like “Squash and stretch” principle, the Arcs principle defines the style and character of an animated element. User interface designers must consider whether the animated element should reflect organic or mechanical qualities. If the former, then the arc animation principle suggests that the element should move along an arched trajectory. If the latter, then the object would move along a straight path.

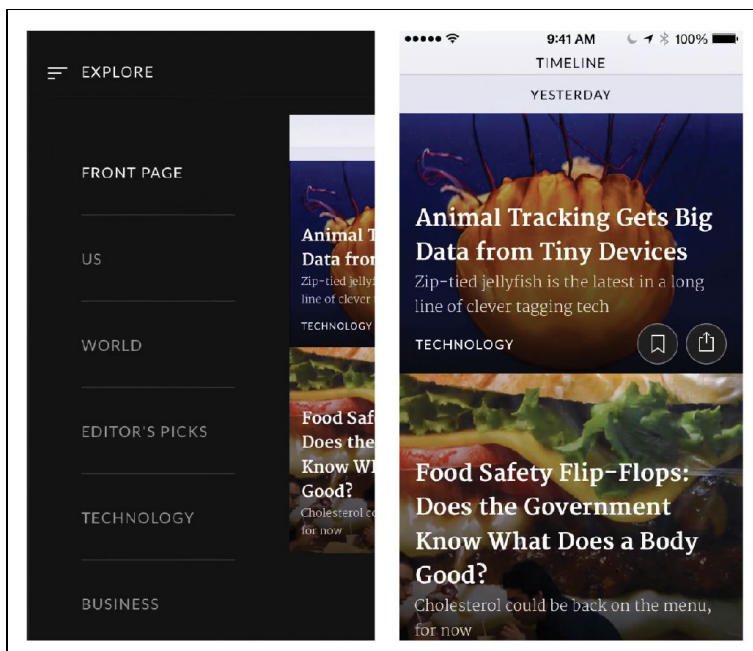
3.8 Secondary action

In English language people use to say the devil is in the details, in some other languages they say the opposite - “God is the details”. Same applies for the Secondary action principle, which argues that each primary movement or action should have a secondary action as a result of the first one.

An example for secondary action in real life would be a squirrel that running across a lawn and then leaping into a tree. The movement of the squirrel’s body and legs (considered the primary action) would be animated to emphasize the animal’s light, nimble, spry gait. The agile, undulating movement of the squirrel’s tail (the secondary action) would be a separate and slightly different type of movement that supports the primary action.

The secondary action is not a different entity, but an integral part of the major movement. When creating a secondary action, user interface designers must consider it as a supportive and secondary visual element to the primary action, and therefore it should not draws the user's attention or distract him from the primary action. In contrast, the secondary action should emphasis the primary action.

In “Timeline” iPhone application (picture 7), the transition that occurs when user clicks on the menu button, activating the application’s main menu, is an example of a secondary action in user interfaces. The primary action is the menu swingin into the view, while the secondary action is the previous content view receding down and to the right of the screen. Both actions occur simultaneously, but the secondary action of the content view supports the primary action of the main menu.



PICTURE 7: Timeline iPhone application. The primary action is menu that swings from the left, the secondary action is the content view which scales down and moves right. (cappivate.co, 2015)

3.9 Timing

This is perhaps the most important principle to master as a motion designer, user interface designer and generally in life - timing is everything. “In the world of animation, timing refers to the number of drawings or frames of a given action, which translates to the speed of the action on film.” (R. Hinman, 2012)

Similar to “Squash and stretch” and “Arcs”, timing defines mood, style and characteristic of an object. However, in UX design timing is the key for user’s frustration when objects are moving too slow, or to confusion when they move too fast.

One of the greater issues when thinking of user interface animations, and animations in general, is what would be the right animation-duration for the human brain to perceive the visuals and understand the action, but at the same time, would not be too slow and would not lead to confusion or boredom.

It would be safe to say that when it comes to animation timing, it is all about balancing. “Rapid change in the interface is difficult to see, perceive and to understand. Slow, on the contrary, slows down the movement of the user’s service. To help the user understand the animation and the position of objects, designers need to calculate the optimal timing. Developers and designer of interactive products have to spend an enormous amount of time testing and determining the optimal timing of which is clear to millions of users.” (Hasan, 2015)

In order to find out the balance in timing, it is necessary to understand how the human brain perceive and process data, or more importantly, how fast it does it.

The Model Human Processor is an abstracted model for understanding users cognitive speed and abilities, developed by Card, Moran, and Newell as a way to summarize decades of psychology research in an engineering model. “It is a high-level look at the cognitive abilities of a human being - really high level, like 30,000 feet. MHP is an abstraction, of course. But it is an abstraction that actually gives us numerical parameters describing how we behave.” (Card, Moran, Newell, 1983)

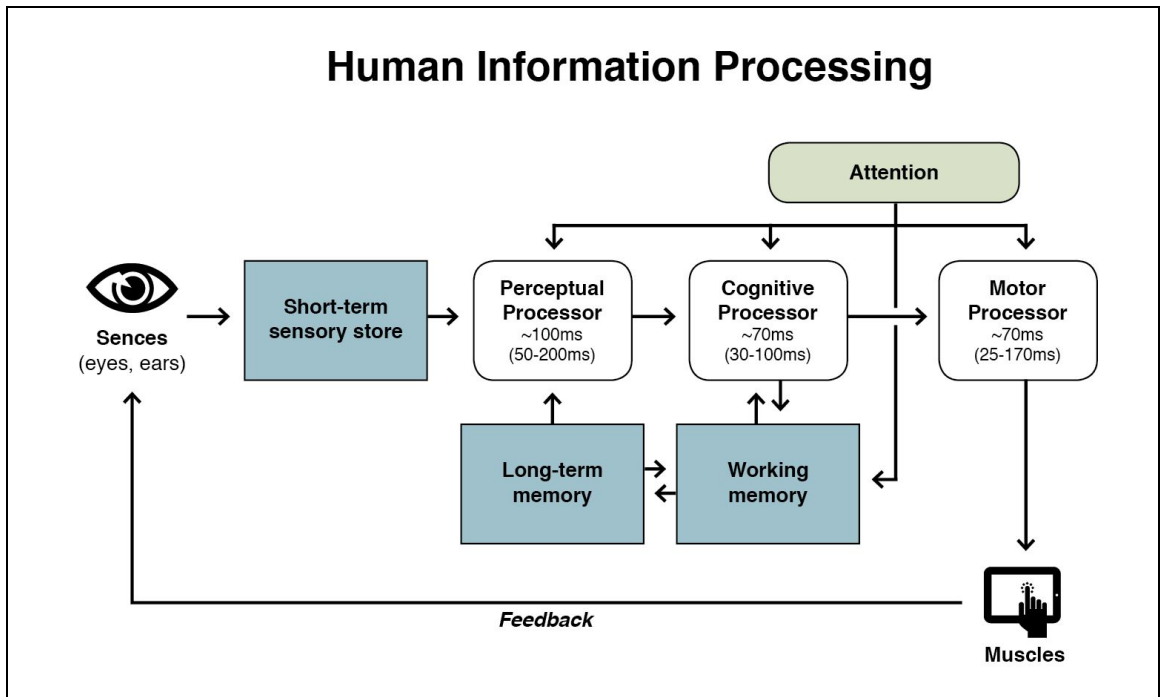


FIGURE 1: Simplified human information processing flowchart, based on the Model Human Processor. The original MHP chart can be found in the Appendices. (Wolowelsky, 2015)

The Model Human Processor does not mean to reflect the anatomy of the nervous system. There is probably not a single area in the human brain corresponding to the perceptual processor, but it is useful abstraction for analyzing human-machine behaviour. For example when looking at the chart above, which is simplified version (and more related to this thesis topic) of the original MHP chart, we can get an approximate estimation guidance for ideal animation length.

In order to make this rough calculation, we must understand the 3 processors as described by the MHP authors. The perceptual processor takes the stored sensory input and attempts to recognize symbols in it: letters, words, phonemes, icons. It is aided in this recognition by the long-term memory, which stores the symbols you know how to recognize.

The cognitive processor takes the symbols recognized by the perceptual processor and makes comparisons and decisions. It might also store and fetch symbols in working memory. The cognitive processor does most of the work that we think of as “thinking”.

The motor processor receives an action from the cognitive processor and instructs the muscles to execute it. There is an implicit feedback loop here: the effect of the action can be observed by your senses, and used to correct the motion in a continuous process.

According to the MHP, when crafting animation timing we should consider that on average it takes a human 240 milliseconds to visually perceive a visual event or action, which could be concluded by summarizing the perceptual processor time (~100 ms), the cognitive processor time (~70 ms) and the motor processor time (~70 ms). However, there is still a huge gap between different people and their processing cycle time.

“Like all parameters in the MHP, the cycle times shown above are derived from a survey of psychological studies. Each parameter is specified with a typical value and a range of reported values. For example, the typical cycle time for perceptual processor, is 100 milliseconds, but studies have reported a range between 50 and 200 milliseconds. The reason for the range is not only variance in individual humans; it is also varies with conditions. For example, the perceptual processor is faster (shorter cycle time) for more intense stimuli, and slower for weak stimuli. People cannot read as fast in the dark. Similarly, a cognitive processor actually works faster under load. Considering how fast human mind works when people driving or playing a video game, relative to sitting quietly and reading. The cognitive processor is also faster on practiced tasks.” (Miller, 2011)

Another aspect of timing in user interface animation, is user preferences and user dropout when animation is too fast or too slow. An interesting research made by a joint group of technical researches from the Technical Research Centre of Finland (VTT) and the Nokia Research Center in Oulu, was examined a few animation-timing cases and rated how users responded to them.

The Finnish researchers from VTT and Nokia used a “fade-in fade-out” transitions between random images taken from Internet image services (e.g. Flickr), and asked the participants to rate which transition perceived faster. The researchers tested two different timing attributes.

The first timing attribute was how quickly the new image (after the first “fade-out”) should be presented to the user: “early”, “equal” or “late”. In equal timing, the fade in

and fade out had the same length. Late timing changed the image when 75 percent of the time was elapsed, and early timing at 25 percent.

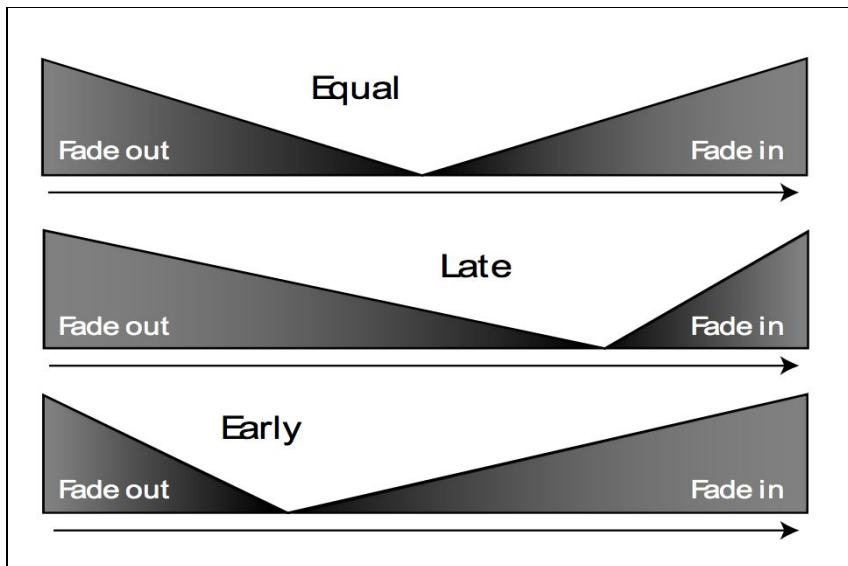


FIGURE 2: Early, equal and late timing, as described in the study “Animated UI Transitions and Perception of Time”. (Huhtala, Sarjanoja, Mäntyjärvi, Isomursu and Häkkinä, 2010)

The second timing attribute which was being monitored by the researchers was which animation speed would be favoured and faster-perceived by users: “slow” or “fast”.

	Early	Equal	Late
Fast	94%	63%	38%
Slow	67%	29%	0.08%

TABLE 1: Proportions of favored transition parameters. Statistically significant results are bolded. (Huhtala, Sarjanoja, Mäntyjärvi, Isomursu and Häkkinä, 2010)

The table above shows the proportions of user preferences when the duration of the corresponding pair was either prolonged or shortened against its counterparts. This unveils the anticipated result that the fast transitions were also perceived faster than the slower ones. The results reveal that the participants perceived the transitions with early timing (for example, that the latter screenshot was brought to the screen early) as faster ones. (Huhtala, Sarjanoja, Mäntyjärvi, Isomursu and Häkkinä, 2010)

These interesting research findings bring us to even more interesting conclusion - which might be in contrast to what some might think about human behaviour - users, also

known as people, are not stupid. While crafting user interface animation, it is important that new content will be brought up rather earlier than later despite of the effects of transition or overall duration.

Furthermore, animation might be used as time-killer or a time-placeholder (for loading heavy content, for example). Nevertheless, it is important to remember that according to the research finding, human beings are pretty good in noticing that content appearing late feels slower, which leads often to frustration and disturbances of attention, despite of the animation.

In other words, the research suggest that people would might process information faster if it would be provided to them faster.

3.10 Exaggeration

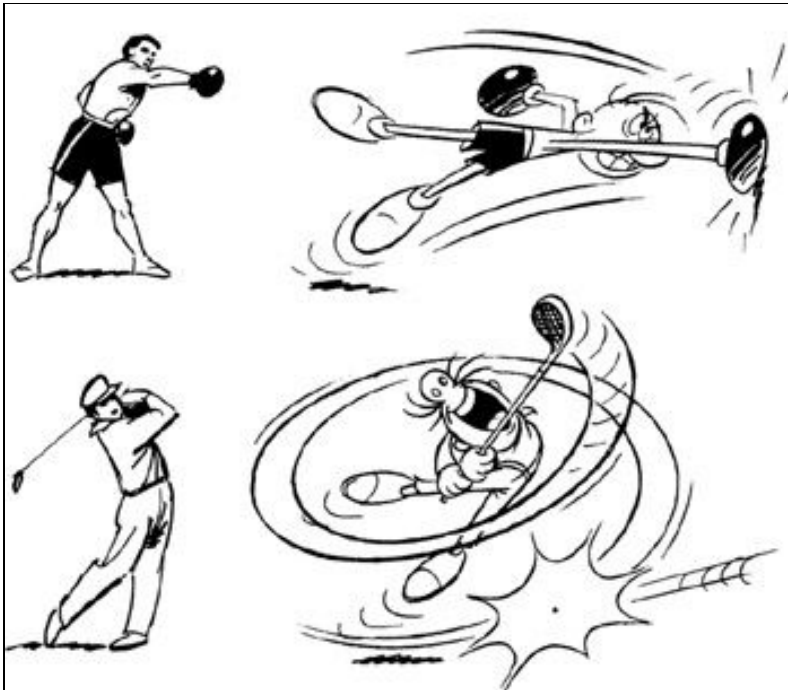
A big part of designing a successful user experience for an interactive product is the fun element. Users love to use products, whether they are mobile app, banking softwares or games, which involved fun. Fun is motivating humans to fulfill tasks faster and more efficiently.

Exaggeration is also an important part in storytelling. For example, one could tell a story in a grey and clinical way, telling only the facts without decorating his words with exaggerating adjective. From the other hand, a more interesting way to tell the same story would be by using many adjective, small side-details and jokes while exaggerating everything to make the story sounds more fun and appealing. As animations are helpers for the storytelling process, the same rule apply for UI animation.

The exaggeration principle is about going away from other animations principles, such as “Arcs” and “Slow-out and slow-in”, which binding animation to reality and laws of physics, and bring more fun and dynamic feeling to the animation.

Yet again, the amount of exaggeration used (or its absence) defines very much the characteristic and style of the animation. Animation which uses too much of exaggerated attributes could be defined as cartoonish and childish, which might not be perceived well

by users of investment-banking software, for example, but would be an enormous success among children using an iPad toy-app.



PICTURE 8: Examples of exaggeration in classic illustration. A subtle unexaggerated illustrations (on the left) in contrast to the exaggerated versions (on the right). (R. Hinman, 2012)

Nevertheless, exaggeration should be used wisely and subtly. The classic definition of exaggeration, employed by Disney, is “to remain true to reality but to present a wilder form”. When applying this principle to an action or movement in a user interface context, developers should exercise a high level of careful constraints. Applying an exaggerated level of exaggeration could easily lead to confusion or annoyance of the user.

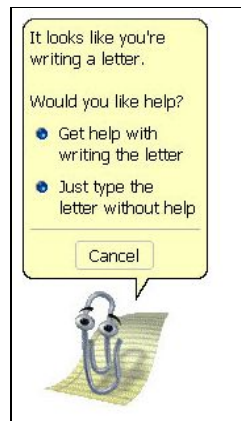
3.11 Solid drawing and appeal

The last two animation principles, as described by Disney’s animators, Johnston and Thomas, are very specific to character animation. “Solid drawing is about honoring the rules of three-dimensional space and giving objects and characters appropriate dimensionality through volume and weight. Solid drawing requires animators to understand the basics of three-dimensional shapes: anatomy, weight, balance, light and shadow. The appeal of an animated character is similar to the charisma of a live actor. A character who is appealing is not necessarily sympathetic — because villains or monsters

can also be appealing. The important thing is that the viewer feels the character is real and interesting.” (R. Hinman, 2012)

In order to describe how these rules applied in the user interface world, we would have to use some abstract and vague terms. Most UI animation will not likely be as complex as intricate figure animations, but the basic principle would still apply. Unless a developer want to give the user a mechanical feeling, he should consider the solid drawing principles regarding animated visuals style and animation behaviour. For example, animation should be believable a not robotic.

The appeal, or charisma, of each character shall be unique, as Disney has always shown, anything can have character: a teapot, a tree, even spoons. Regarding user interfaces, we should consider how the overall animation will contribute to the design and make the overall experience more satisfying. For example, the old Microsoft Office “Clippy” animation would be an example of how to add an unappealing and even annoying animated element which affects the overall software’s user experience. (Waterhouse, 2011)



PICTURE 9: Screenshot from Microsoft office “clippy” assistant. An example of overusing the appeal principle and creating in fact an annoying and unappealing experience. (Microsoft Office Screenshot)

4. EXPERIMENT - THE INFLUENCE OF UI ANIMATIONS ON USER REACTION AND TASK FULFILLMENT SPEED

4.1 Experiment overview

Inspired by the research and experiment compiled and written by Finnish researchers Huhtala, Sarjanoja, Mäntyjärvi, Isomursu and Häkkilä regarding animation timing, I decided to compose an animation user experiment of myself, in order to test a few statement brought up by variety of sources previously in this thesis work.

The key difference between the following experiment to the Finnish study is the data collection approach. While the Finnish study was focusing on users preferences, which could be measured primarily by asking participants for their personal opinions, the following research does not involve questioning at all. Instead of collecting participants opinion, this experiment is collecting participants actions.

Furthermore, while the previous study was comparing different types of animations and animation timing, the following test also includes cases where no animation was applied to the appearing object at all, in order to test how is it influencing user reaction and task fulfillment.

The main reason for compiling this experiment is to check the influence of animation on user reaction and task fulfillment speed. This Experiment is not comparing only animated elements against not animated object, but also comparing different animation styles and timing presets.

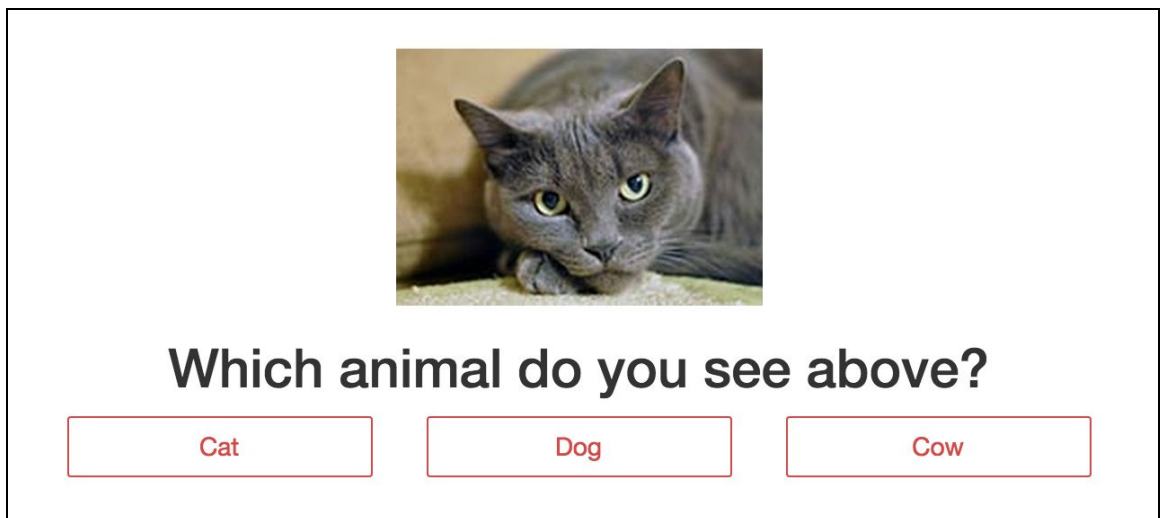
The secondary purpose of this experiment is observe how people interact with interactive products in terms of reaction time and information processing, in order to find out how long is the “dead zone” in our brain dedicated to basic information processing. Determining the length this so-called “dead zone”, is important for UX designers and animators in order to craft smooth UI animations which allow users to perform UI tasks easily, as during this time, the user’s brain is busy with processing and it would not be recommended to bother him with new tasks until he completely processed the initial information.

It would be quite interesting to put into a practical test a few statements brought earlier in the Timing chapter (3.9) regarding the Model Human Processor's processing times.

The experiment's approach is that in order to measure reliable data, we should not tell users what the study is actually about. Therefore we shall collect the data about how they behave while they are not prepared, so they would not change their mind based on personal visual preference or prejudices.

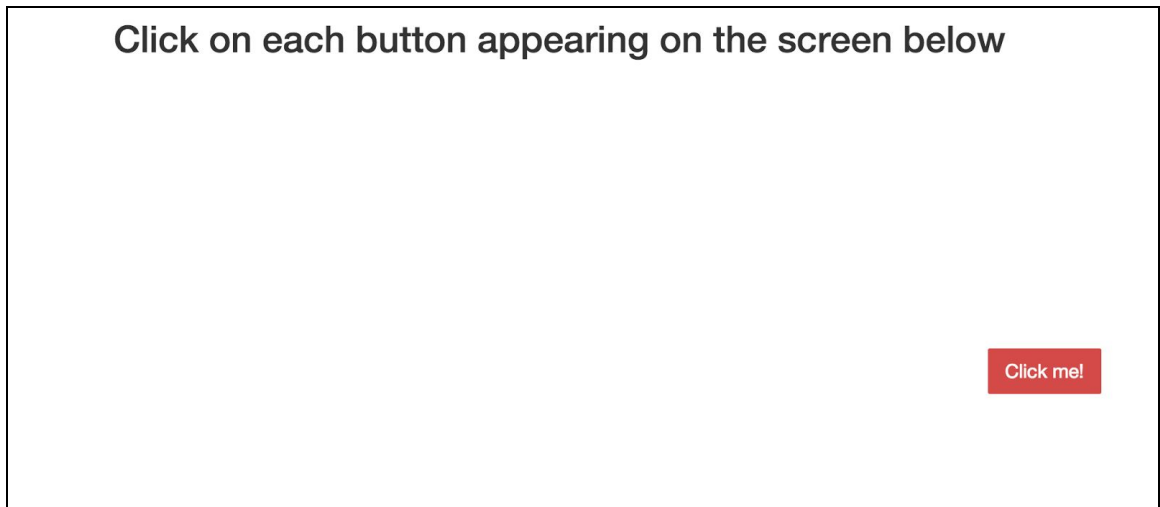
In my experiment, I have asked the participants to fulfill 3 different tasks. The participants have been told that this is a "human behaviour" task, and in fact did not know what is the specific purpose of the test or which data is being collected.

In the first task, an image of an animal appeared on the screen with different and random transition or without any transition at all, and the user has been asked to click on the button which says the correct animal's name.



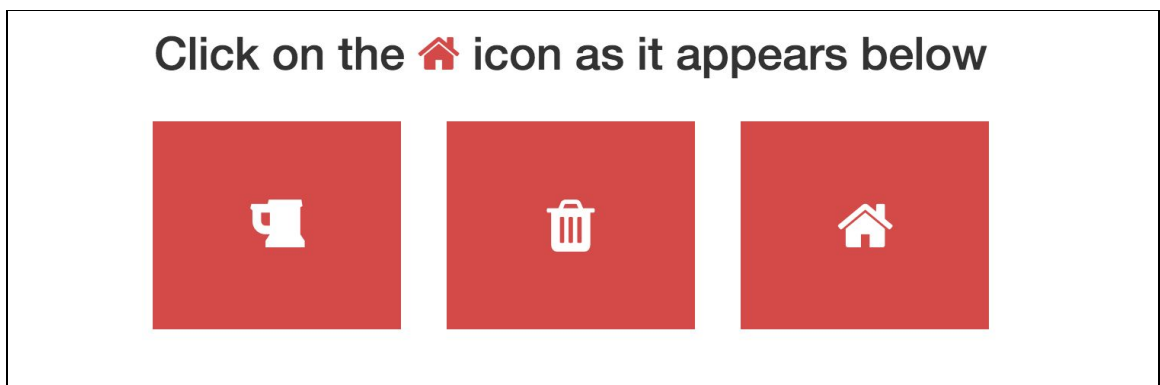
PICTURE 10: Screenshot from the "animals test". (Wolowelsky, 2015)

In the second task, buttons appeared on a random location on the screen, yet again with different and random transition or without any transition at all, and the user has been asked to click on each button as it appears on the screen.



PICTURE 11: Screenshot from the “buttons test”. (Wolowelsky, 2015)

In the third task, a series of 3 icons has been shown on the screen, and the users has been asked to click on the “home” icon. Just like the other test mentioned above, this test included different variations of transition, but also a “one by one” animation preset - an animation preset which shows each icon box with a tiny delay (100 ms) after the previous one. This animation is of course the most time consuming as it requires delaying the appearance of the last two icons.



PICTURE 12: Screenshot from the “icons test”. (Wolowelsky, 2015)

All of the tasks in the experiment has been repeated for 22 times per user, and involved moving the mouse to a certain point and clicking on a button. In order to measure the user initial reaction to the new appearing object, I have recorded the time spent from the initial image appearance until the time when the user moved his mouse for the first time. In order to track when the user finished the task, I have recorded the time spent from the initial image appearance to the time when users clicked on the right button.

The amount of unanimated test elements and animated test elements have been shown equally to the user. For each animated test element, a random animation timing property has been applied:

- **Slow:** 200 milliseconds duration.
- **Middle:** 150 milliseconds duration.
- **Fast:** 100 milliseconds duration.

In addition, a random animation style preset was applied:

- **Fade In:** Element's opacity grows from 0% to 100%.
- **Fade In Down:** (in the "Icons test" - **Fade In Left**) Element's opacity grows from 0% to 100% and element was moving from top to bottom. (in **Fade In Left** - from left to right)
- **Fade In Zoom In:** Element's opacity grows from 0% to 100% and zoomed in from 80% to 100%.

The experiment is based on over 12,000 test samples, fulfilled by 207 participants from 28 different countries which have taken part in the experiment. The experiment's full details and specs could be found in the appendices.

4.2 Experiment results

Before going through the results and data, it is important to mention the feedback participants gave after taking the tests. Participants were encouraged to comment, ask and give feedback via email or in the social media threads where the experiment was advertised (Reddit, Facebook, Twitter).

Based on the feedback and communication with participants who took the tests, the two most common triggered emotion was fun and curiosity. It was important for the test accuracy that people will feel comfortable to engage with the tasks, in order to simulate a “real-life” user interface, where people go because they want to and not because they are forced to.



PICTURE 13: Collection of comments to the experiment on Reddit, Facebook and email message. (Wolowelsky, 2015)

The results themselves were a bit more difficult to parse, and would require a closer analysis of the data, while considering many parameters.

In a glance look at the bars charts below, we can clearly see that in all three tests the not-animated elements got a faster reaction (red bars) and fulfilled quicker (yellow bars) than the animated elements - 5.82 milliseconds faster on the animal test, 33.34 milliseconds faster on the buttons test and 62.41 milliseconds faster on the icons test (or 12.34 milliseconds, when excluding the “One by one” transition).

However, when taking into consideration the time it took the animations to be completed (gray-shaded areas - 100 ms, 150 ms or 200 ms), we would get much quicker response from the time the animation was completed until the time first reaction and the task fulfillment time was indicated.

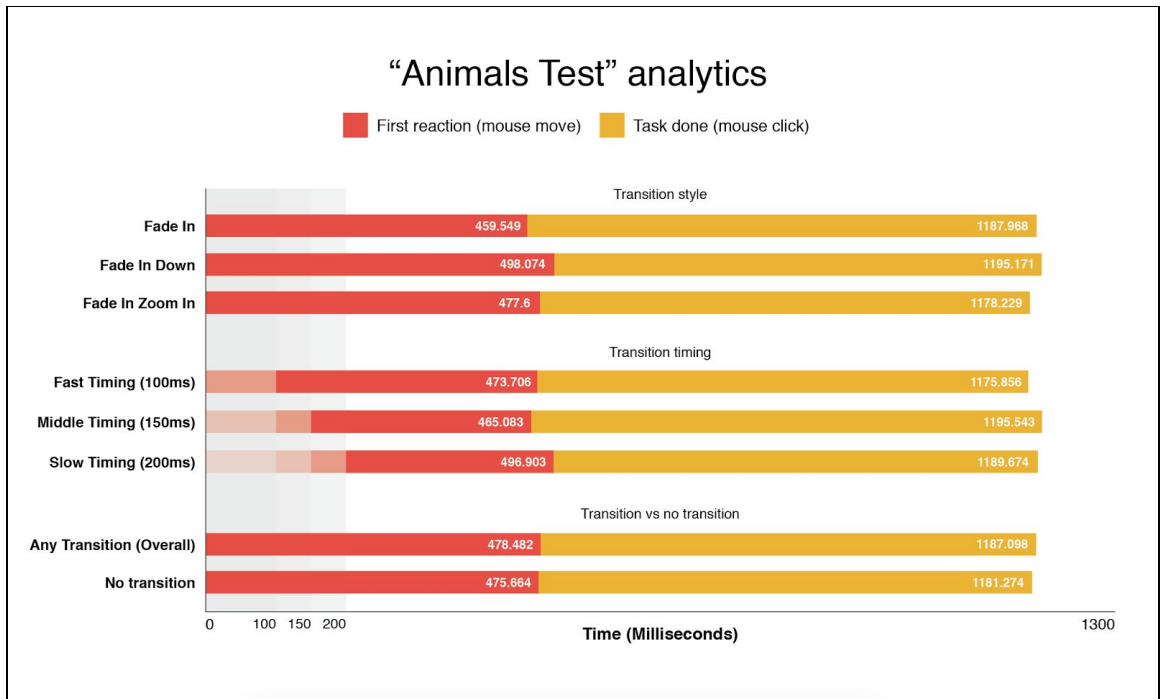


FIGURE 3: “Animals test” analytics. Showing how long it took, in average, to get user reaction and task fulfillment while using applying different transition presets. (Wolowelsky, 2015)

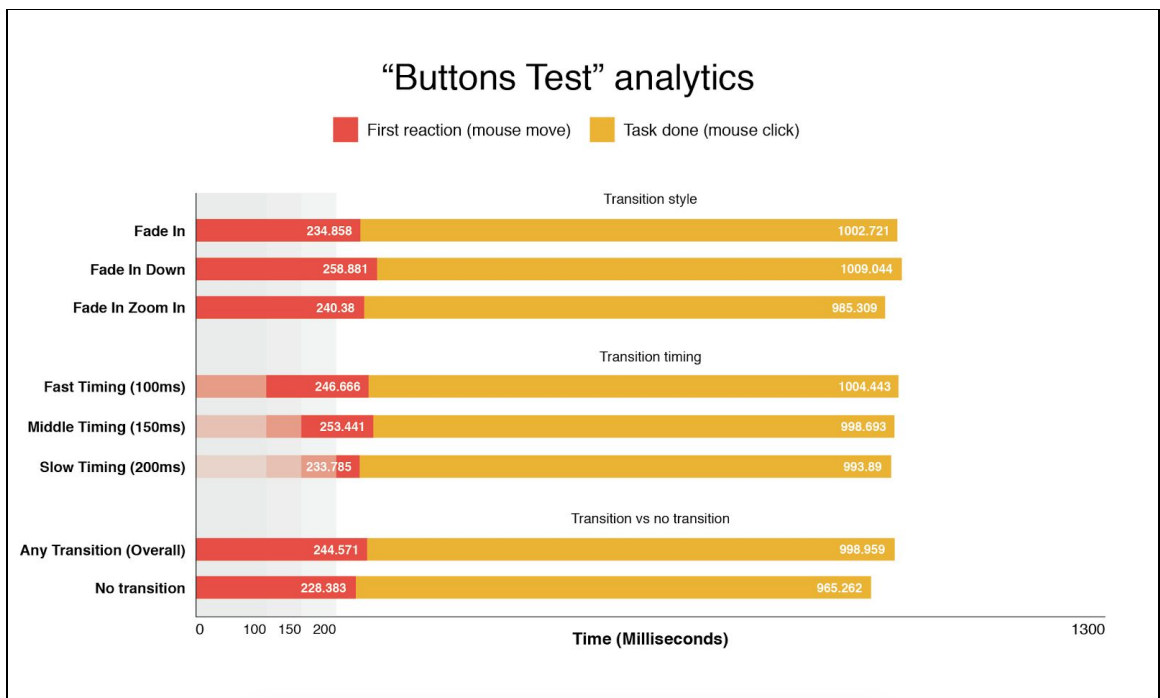


FIGURE 4: “Buttons test” analytics. Showing how long it took, in average, to get user reaction and task fulfillment while using applying different transition presets. (Wolowelsky, 2015)

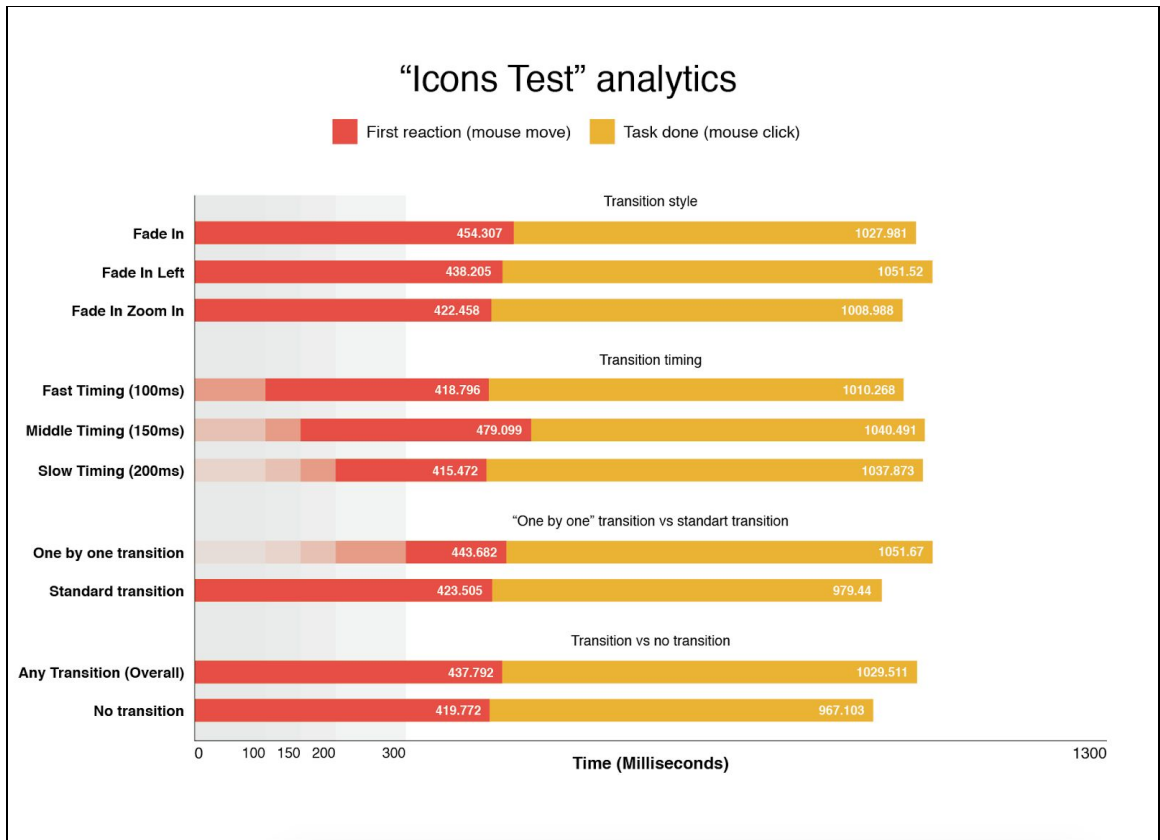


FIGURE 5: “Icons test” analytics. Showing how long it took, in average, to get user reaction and task fulfillment while using applying different transition presets. (Wolowelsky, 2015)

Unlike the “animals test” and the “buttons test”, the “icons test” has also the “One by one” transition, which is a worth considering parameter when analysing the graphs. If one would like to compare the results on “Any transitions” to “No transition”, in all three graphs, we must notice that the “one by one” transition is significantly longer (at least 300ms) than the standard transition. Therefore, it would be wise to look at the “One by one” transition as a separate comparable parameter and compare the “No transition” bar with the “Standard transition” bar instead of the “Any transition” bar.

Looking at the overall results, it seems that while the different timing presets did not significantly affected the reaction speed, participants performed faster results when the “Fade in zoom in” style-preset was applied.

4.3 Experiments conclusions

The most obvious conclusion from the experiment would be that animation slightly slows down users. This is an important conclusion to remember during the animation crafting process. However, I would personally argue that in the wider context this statement is not unequivocal, as the measuring system performed in this experiment was very straight forward and did not considered other significant user experience parameters such as user emotions and preferences, and did not behave as part of a wider interface. Nevertheless, there are still very interesting conclusion that could be made from this experiment regarding how the human brain works, and more specifically for the right usage of UI animations.

While the “animals tests” and the “icons test” represents UI tasks where the user actually have to think in order to fulfill correctly, with three different optional answers, the “buttons test” represent a reflexive task where the user doesn't have to think, with only one possible answer.

Interesting fact is, that the reaction speed for both of the thinking-related test is significantly slower, with about 450 milliseconds until the initial user reaction, than the reflexive “button test” - with about 235 milliseconds. This fact could imply that users usually think, or at least finish basic information processing, before they go to action and do.

Furthermore, when comparing the reflexive “buttons test” reaction figure (~235 ms) to the Model Human Processor, which was mentioned in the timing chapter (3.9), we can see that the model which was illustrated back in 1983 was not too far off. As a reminder, according to the MHP, the human brain processes information with three processor - Perceptual (~100ms), Cognitive (~70ms) and Motor (~70ms) - which almost magically adds up to 240 milliseconds.

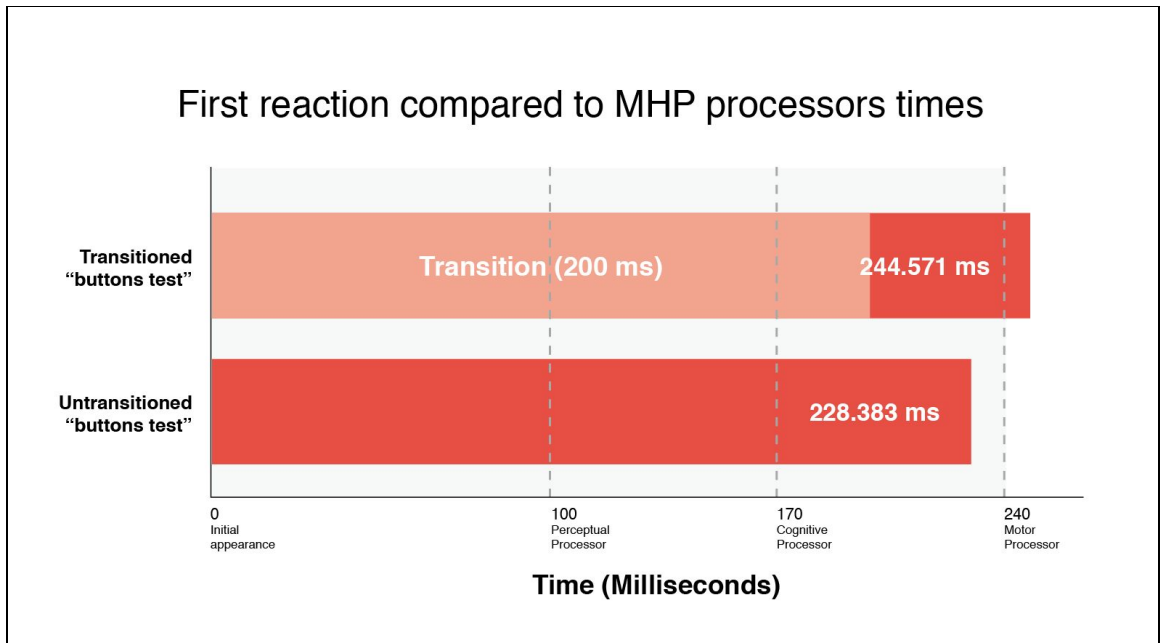


FIGURE 6: First reaction compared to MHP Processor times. The transition length occupies most of the “dead zone” - when the brain is busy with basic information processing. (Wolowelsky, 2015)

Another interesting conclusion regarding how people perceive and process information is that according to the results, the human brain does not wait for the transition to end in order to process the information, but start processing slightly after the first animation frame, when the element is already visible enough. Therefore, we could conclude that even though users may react slightly faster to non-animated elements, transitions neatly fill the mandatory gap between the initial element appearance time to the MHP processors finishing time. This conclusion is also supported by the facts brought by the experiments results, as we could not observe any significant effects on the results when applying the three different timing presets, all of them under the 240 milliseconds MHP delay.

5. UI ANIMATION IN A TECHNICAL CONTEXT

This chapter is about the technical aspect of UI animations in web platforms. It includes an historical overview and a deeper survey of the best practices, advantages and disadvantages of variety of contemporary UI animation implementation methods in web platforms.

5.1 Overview of web animation methods overtime

Executing a highly functional web application with rich visual interface is becoming more and more common. Nowadays it is even possible to develop a full mobile applications with web-based techniques and platforms solely, which look and feel exactly like native applications.

Along the rapid development and usage of web based app, and considering the accelerated importance the UI/UX field get in the past years, many solutions has been developed in order to help developers to quickly create stunning and well-performed UI animations and transitions.

If we will go back in time and try to review web animations in the past, we will find a long, complex and very incompatible history of plug-ins and workaround in order artificially integrate animations into the web.

5.1.1 The dawn of the Internet

At the very beginning of the internet, back at the early 90s, the only way to render and display animations on a web page was by embedding a GIF image (Graphics Interchange Format) into an existing web page. Back then, the web visual interface was limited anyway to text with basic formatting features, so a moving image could have been considered as an innovative feature.

Though the GIF animations looked fancy enough to the average internet consumer of 1994, this solution had a few major issues. The first major issue is the file size of a GIF image - since each frame is an image of itself, an average few-seconds GIF image was a

big hassle to load for the fresh internet users, considering the slow internet connection which portrayed the 90s, alongside with noisy modems and high telephone bill.

The second issue with GIF was the quality. GIF animations supports maximum of 256 colors for each frame, which could have been sufficient enough in the early 90s, but with the fast development of graphic cards which could easily render complex 3D graphics, the need for high quality animation engine for the web raised quickly.

Nowadays GIF images are still widely used, though mainly for animated content and not for UI components, for example for showing short video-snippets on social media sites.

5.1.2 The Flash and DHTML era

In order to provide a high quality and well-compressed animations components for the web, two different disciplines has been developed during the mid-late 90s.

One discipline, which also known as Dynamic Hypertext Protocol (DHTML), was in favour of combining several web standards that create interactive and visually appealing websites, while still using fully web-based programming, formatting and markup languages which could be rendered solely by the web browser without any additional plug ins.

DHTML standards evolved initially when only two competitors ruled the web-browser market: Microsoft with Internet Explorer, and Netscape Communications with Netscape Navigator. The competitions between these two companies radically affected DHTML development and usage, as each company had its own interpretation of what DHTML means and how it should be implemented. Each competitor decides for itself which feature should be included (and which should be excluded) and even forced their users to install extra plug-ins for specific features, in order to purposely prevent the competitor's browser to run these features (Teague, 2001). This situation, when each company develops its own browser while completely ignoring the recommended web standards, created a confusion among web developers and internet users as well.

Many web developers felt that DHTML could have been ideal, as in theory it renders all the visual effects on the client side and the user gets an out-of-the-box experience without

any the need for extra technical installation. However, because of unenforced standards and very wild competition, DHTML considered buggier, more incompatible and not developer-friendly as initially designed.

The second discipline for implementation rich interactive components on the web, was by embedding a completely different piece of software into a web page, just like embedding images. This software, known today as Adobe Flash, was initially developed by FutureWave and acquired later by Macromedia for further development, called Shockwave Flash.

The advantages of Flash was clear, and therefore many developers decided to use it as their primary web design/development tool. In contrast to the DHTML code-only approach, Flash was relatively easy to use with a visual UI. It also has a healthy combination of design and development in one framework, which means that a web developer could design the visuals of a website and add the site's functionality and interactivity solely in Flash, without the need to integrate external platforms or languages like Javascript or CSS. Most importantly, Flash was a sole framework and relatively cross-platform, which means that it did not matter on which browser the user used the site, as long as he has the Shockwave/Flash plugin installed, he will get exactly the same result as designed by the web design team.

Regarding animations, the introduction of Flash let web developers to apply animation principles in full for the first time. While before the Flash era animations were mainly heavy pixel-based components with rough connection to UI, with Flash developer could easily create vector and relatively lightweight UI animation, use important animation principles like Timing, Slow-in-and-out or Arcs and apply them neatly to interactive user interfaces.

Flash was very popular during the late 90's and the the first years of the new millennium. However, since Flash does not fit to the criteria of any web standard, and in fact it is an external software, it had a few minor disadvantages from the beginning, these disadvantages became more and more major with time, as the web usage evolved.

The first and obvious disadvantage of flash was the technical barrier a user has to go through when installing the plugin on his computer. This applies mostly to elderly users,

young children, or users who use company-owned computers without the sufficient permission for installing extra software.

Another major disadvantage of flash was its lack of search engine optimisation (SEO). With the gaining popularity of search engines, the importantancy of SEO raised as well. Since flash is an external software per-se, embedded in a website, most search engine crawlers, such as Google, could not or had major difficulties to crawl flash-based websites.

The last nail in the coffin of Flash was nailed by Apple’s former CEO and founder Steve Jobs. In April 2014, Jobs published an open letter called "Thoughts on Flash" explaining why Apple would never allow Flash on their mobile devices (iPhone, iPod and iPad). He cited the rapid energy consumption, poor performance on mobile devices, abysmal security, lack of touch support, and desire to avoid "a third party layer of software coming between the platform and the developer", but above everything, Jobs claimed that Flash is not a open platform but “100% proprietary” and “controlled entirely by Adobe and available only from Adobe.” (Jobs, 2010)

From the moment Apple, which in the last quarter of 2011 domained 45.3% of the smartphones market-share in the US (Invotex Group, 2012), decided to drop the support for the Flash plugin, and along with the uprising of mobile devices as major medium to consume web content worldwide, the usage of Flash for producing websites and web application decreased dramatically.

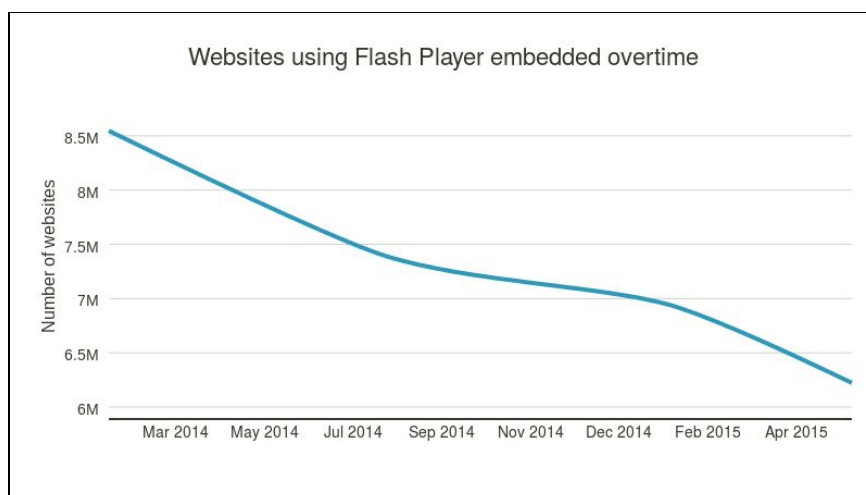


FIGURE 7: Websites using Flash Player Embed overtime (BuiltWith Trends, 2015)

5.1.3 The post-Flash era (the HTML5 Era)

Steve Jobs have been criticized for hypocrisy by many after his “Thoughts on Flash” letter when claiming Flash is not an open framework. Nevertheless, Jobs has opened the gate for a new era of web development - where web users can consume lightweight, rich and interactive content which is search-engine optimised, without the requirement of installing any extra software or plugin and follows the latest web standards. Though Apple could not be considered as an open-source company, and for some it is actually represents the exact opposite, it did supported the development of an open-sourced web rendering engine called WebKit.

The WebKit engine was the basic ground for the development in 2007 of many extended CSS modules (also known as CSS3), such as CSS 2D and 3D transformation, CSS filters and most relevant to this thesis - CSS animations and transitions. (Hyatt, 2007)

The big innovation behind the animation feature which has been added to the existing CSS, was the ability to create lightweight and easy to develop web animations on the browser. Of course, rendering animations on the client side without Flash was possible during the DHTML era, but it could have been done only using JavaScript by programmatically removing and re-rendering the animated element with a different style attribute value (such as position, color or size).

5.2 Comparing contemporary web animations solutions

The difference between JavaScript programmatic animations to CSS animations could be roughly compared to the difference between the processes of crafting frame-by-frame classic animations and computer generated animation. CSS animation is an out-of-the-box solution which allows developer to write less code and does not requires the user's machine to render relatively-complicated and CPU-consuming scripts. However, CSS animations can still be quite limited in terms of functionality, complexity and compatibility.

In this chapter I will outline the common solutions for web-animations in the post-Flash era and the differences between them. I will be focusing on 3 solutions: CSS animations, jQuery.js Animate, and Velocity.js.

In order to equally compare these practices, I will write and run the same type of animation with each of the platforms and analyze the results in terms of performance, compatibility, functionality and developer-usage friendliness.

The animation which will be tested is called “Tween right” - moving a text element from 0 to 100 pixels forward (right) on the X axis.

5.2.1 Usage

5.2.1.1 CSS Animations and transitions

As reviewed earlier, CSS animations and transitions are originally WebKit features, nowadays available at all major modern browsers. The usage of CSS animations and transitions is declarable and not programmatic, as seen in the example below.

```
#element {
  animation-name: myAnimation;
  animation-duration: 1s;
  animation-timing-function: linear;
  animation-iteration-count: 1;
  // Or the shorter, more common inline usage:
  animation: myAnimation 1s 1 linear;
}

@keyframes myAnimation {
  from { left: 0; }
  to { left: 100px; }
}
```

CODE SNIPPET 1: CSS animation code, as used to perform “tween right” animation. (Wolowelsky, 2015)

The declarable usage of CSS animation is very easy to use and can be easily read by any developers or designers with a little bit of CSS-syntax understanding. There is no need to

program anything or to run any additional scripts. This is a native-browser code which can be rendered by any modern browser.

The only issue regarding this usage is its compatibility. When a developer declaring this animation at its current state, he might be risking compatibility issues for many old browser users worldwide. In order to make sure this tiny block of code works on every browsers which support the CSS animation feature, developers must add a prefix to each CSS attribute they declare. This prefix is a reminder from the old days when web standards were in chaos, and each browser developed its own version and interpretation for advanced CSS features.

Browser development politics aside, this situation makes these neat and declarable practice into a longer and more time consuming code, as seen in the example below.

```
#element {
  animation: myAnimation 1s 1 linear; // Modern browsers
  -WebKit-animation: myAnimation 1s 1 linear; // Older WebKit browsers (chrome,
safari)
  -moz-animation: myAnimation 1s 1 linear;
}

@keyframes myAnimation {
  from { left: 0; }
  to { left: 100px; }
}

-WebKit-@keyframes myAnimation {
  from { left: 0; }
  to { left: 100px; }
}

-moz-@keyframes myAnimation {
  from { left: 0; }
  to { left: 100px; }
}
```

CODE SNIPPET 2: Full CSS Animation compatible code, as used to perform “tween right” animation. (Wolowelsky, 2015)

Fortunately, there is a solution which allows developers write the same amount of code as in the first example and be compatible as in the second one, by using SASS. SASS

(Syntactically Awesome Stylesheets) is a CSS compiler that extending the functionality of the native CSS and compiles it into a native CSS code, which the browser can read as it was pure CSS. By using SASS, we can declare a pre-made mixing which takes all the prefixes into account. By using SASS, we can write clean, neat and short code in order to perform a fully compatible CSS animation, as seen in the example below.

```
#element {
  @include animation(myAnimation 1s 1 linear);
}
@include keyframes (myAnimation) {
  from { left: 0; }
  to { left: 100px; }
}
```

CODE SNIPPET 3: CSS animation code using SASS compiler extension, as used to perform fully compatible “tween right” animation. (Wolowelsky, 2015)

5.2.1.2 jQuery.js Animate

Using JavaScript, developers can programmatically create a “frame by frame” animation. It does not mean that they have to craft each frame like a classic animators, but does mean that they need to tell the machine to do that by writing a script, in contrast to the declarable usage of CSS animations.

In JavaScript, there are two methods that can be used to write a frame-by-frame animations. The first method, and the most compatible one is by using the old “setInterval” and “setTimeout” native functions, just like the following example.

```
var left = 0
function frame() {
  left++ // Update parameters
  document.getElementById('element').style.left = left + 'px' // Update element's
  style
  if (left == 100) // Check finish condition
    clearInterval(id)
}
var id = setInterval(frame, 10) // Draw every 10ms
```

CODE SNIPPET 4: Raw JavaScript code, as used to perform “tween right” animation, using old “setInterval” native function. (Wolowelsky, 2015)

This concept is quite simple - every 10 millisecond perform an action, which in this case would be to move the animated element 1 pixel to the forward. This code usage is quite time consuming to develop when we have multiple element with multiple animated attribute to animate. Therefore, a quite early solution was created in order to abstract this process into the famous JavaScript library jQuery.js, using its jQuery.animate function. The simplicity of the usage can be illustrated in the following example.

```
$('#element').animate({
  left: 100 // Visual attribute and value
},1000); // Total duration
```

SNIPPET 5: jQuery.js code, as used to perform “tween right” animation. (Wolowelsky, 2015)

As matter of fact, the result of both examples would be identical, but the development time used to craft this simple animation would decrease dramatically.

5.2.1.4 Velocity.js

The second method creating JavaScript animation uses a different native method called “requestAnimationFrame”. This method is not supported by some older browsers, and therefore it is less compatible. However, “requestAnimationFrame” seems to perform much better than the old “setTimeout” and “setInterval” functions in terms of CPU usages and animation smoothness. Because its compatibility issues, the raw code using “requestAnimationFrame” function might be a bit longer, in order to cover all browser usage scenarios, as seen in the following example.

```
var requestAnimFrame = function() { // Lookup the right requestAnimationFrame call
  return (
    window.requestAnimationFrame    ||
    window.WebKitRequestAnimationFrame || // WebKit browsers
    window.mozRequestAnimationFrame // Mozilla browsers
  );
}();
var left = 0;
```

```

var element = document.getElementById('element');
function frame(timestamp) {
  left++; // Update parameters
  element.style.left = left + 'px' // Update element's styl
  if (left < 100) { // Stop animation when left attribute is 100
    requestAnimationFrame(frame);
  }
}

requestAnimationFrame(frame);

```

CODE SNIPPET 6: Raw Javascript code, as used to perform “tween right” animation, using modern requestAnimationFrame native function. (Wolowelsky, 2015)

To abstract this code, Julian Shapiro, a web developer and web animation specialist, created Velocity.js. Velocity is a web animation engine, which performs as a JavaScript library and abstracts the concept of “requestAnimationFrame” into a simple and easy use usage.

```

Velocity(document.getElementById('element'), {
  left: 100 // Visual attribute and value
}, {
  duration: 1000 // Overall Duration
});

```

CODE SNIPPET 7: Velocity.js code, as used to perform “tween right” animation. (Wolowelsky, 2015)

2.2.2 Performance

A good way to test client-side performance, is by checking how much CPU (Central processing unit) the process (in this case - a web page) uses from the client’s machine. The way to measure CPU usage is by CPU time - the amount of time the CPU is actually executing instructions. There are several computer softwares which calculates the CPU time for each process, and returns the amount of CPU usage (in percentages) per process from the overall user’s CPU.

To run the following test I used Google Chrome’s “Task Manager”, a native Chrome’s feature for advanced browser-tabs management, in order to determine each animation’s scenario CPU usage. The full test specs could be read in details in the Appendices.

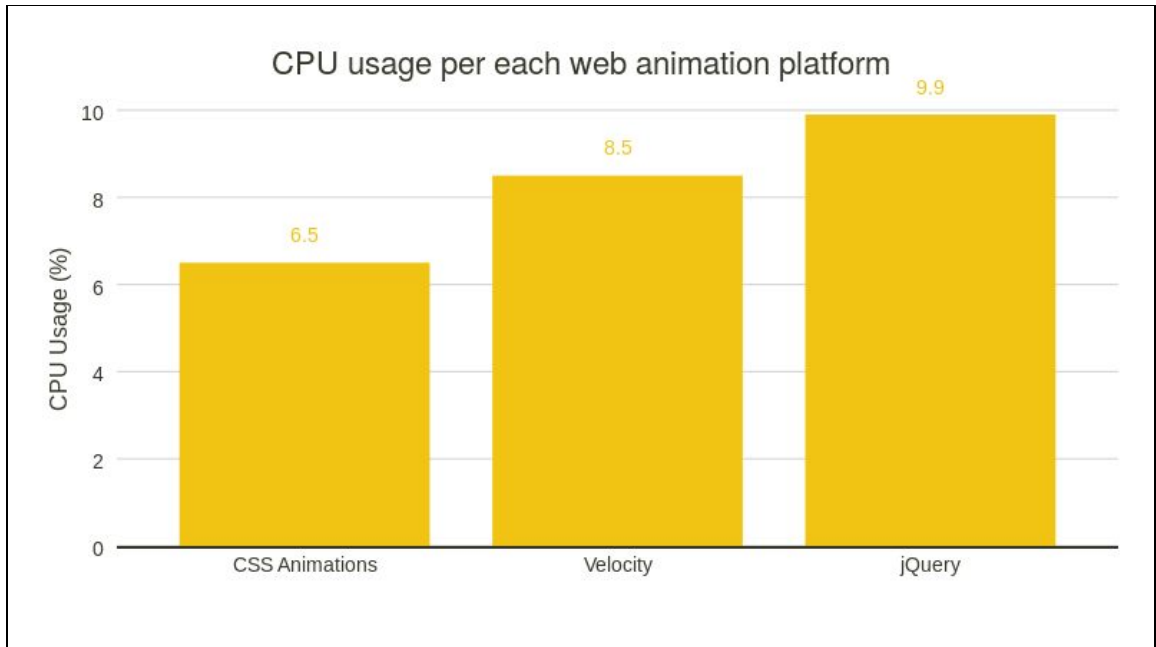


FIGURE 8: Comparing different web animation frameworks CPU usages (lower is better). Based on “Tween right” animation example.

According to the test I ran with these relatively-simple animation example, we can clearly see that pure CSS animations are significantly less CPU-consumptive than JavaScript-based frameworks. Furthermore, Velocity seems to be much faster than jQuery, as it uses “requestAnimationFrame” function by default.

2.2.3 Compatibility

There is a big discussion about compatibility in the web development world. The main problem with web-compatibility is that every user is using a different devices specs to interact with web products. For desktop users, there are 3 major computer operating systems (MacOS, Windows and Linux) and 5 major browsers (Internet Explorer, Chrome, FireFox, Safari and Opera). The same applies for mobile - there are 3 major operating systems (iOS, Android and Windows Phone) and over 5 major browsers. Each

operating system and each browser have different versions which are still in use worldwide, and each version has its own support range for web features.

One of the biggest discussion is whether web developers should make an extra effort to make website fully-compatible for legacy browsers, such as Internet Explorer versions 8 or 9, which unfortunately are still in use by some users.

In this chapter I will present data I have collected about the different web animation platforms and their compatibility range. The following table, shows simply which browser and browsers' version supports each platform.

Browser / Framework	CSS Animations	jQuery 1.x (JavaScript - setTimeout)	Velocity.js (Javascript - requestAnimationFrame)
IE	10 and up	6 and up	8.0*** and up
Firefox	16 and up	All versions	All versions
Chrome	4* and up	All versions	All versions
Safari	4* and up	5.1 and up	All versions
Opera	12.1 and up	12.1 and up	12.1 and up
iOS Safari	All versions	6.1 and up	All versions
Opera Mini	No support	Limited Support	Limited Support
Opera Mobile	12.1 and up	All versions	All versions
Android Browser	2.0** and up	2.3 and up	2.3*** and up
Chrome for Android	All versions	All versions	All versions
BlackBerry Browser	7.0 and up	No Data	No Data
Symbian	“Anna” and up	No Data	No Data
Firefox Mobile	All versions	All versions	All versions

* Requires “-WebKit-” prefix.

** Old android browsers can animate only one property at a time.

*** JavaScripts' RequestAnimationFrame is supported only from Internet Explorer 10 and Android browsers 4.4. When user loads Velocity.js on such an old versions, it will use a setTimeout Callback, therefore performance will suffer.

TABLE 2: Web animation platforms compatibility by browsers.

Sources: caniuse.com, mobilehtml5.com, w3school.com.

The next step in my compatibility research was to find out, for each web animation platform, how many users worldwide would be missing out the animation or would be unable to see it or interact with it. I crossed out data from worldwide browsers usage with the information from the last table, the results are as following.

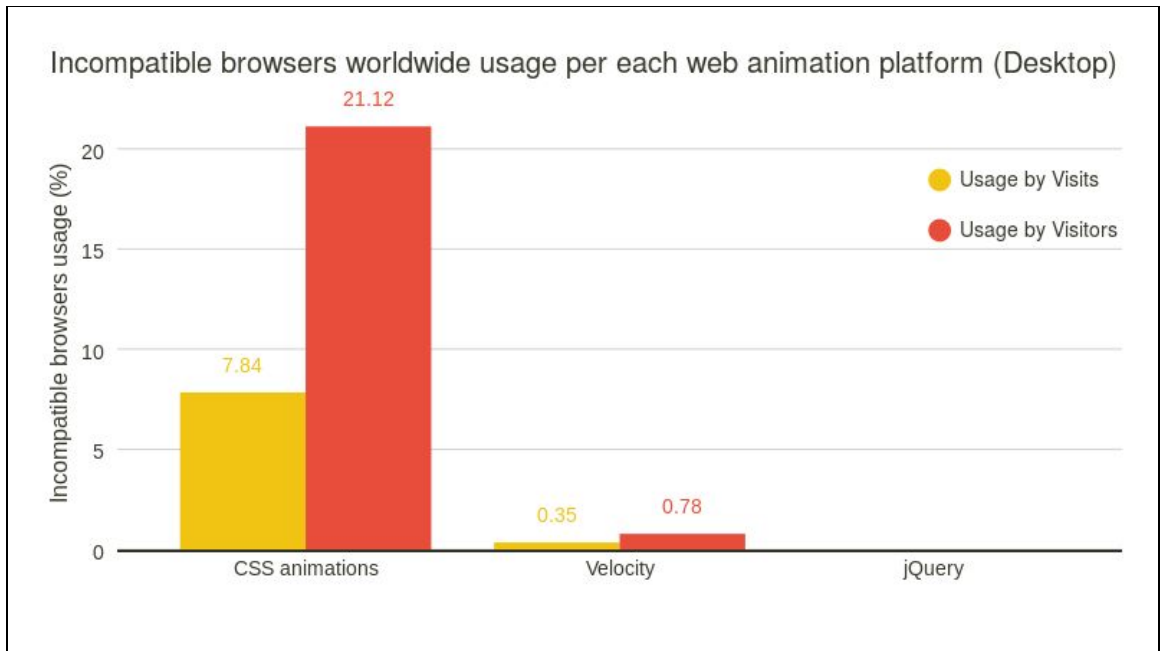


FIGURE 9: Comparing web animations frameworks incompatibility per desktop browsers usage, based on the June 2015 browsers' market share worldwide. (Lower the better)

Sources: Netmarketshare.com (Red bar), statcounter.com (Yellow bar) and table 2 June 2015.

The difference between the red bars the yellow bars is the sources of data. I used to different market-share trackers - NetMarketShare and Statcounter. The key difference between these two source, which explain the dramatic difference, is that NetMarketShare is tracking the amount of unique visitors, while Statcounter is tracking the amount of visits (Buckler, 2015).

To add even more fuel to the fire of web-compatibility debate, we can learn from this chart that by using CSS animations, over 21% or all desktop users (the actual people behind the screens) would not be able to use, interact or see the animation. However, If we are looking at the amount of overall usage, regardless the amount of people, the incompatibility rate drops to less than 8%.

For mobile users, it seems the incompatibility rate is much lower. As seen by analyzing the data from NetMarketShare, only 6.87% or users worldwide (unique visitors) will

have difficulties consuming web animations produced with CSS animations, while Velocity.js consumers should get full experience without major visibility or usability issues.

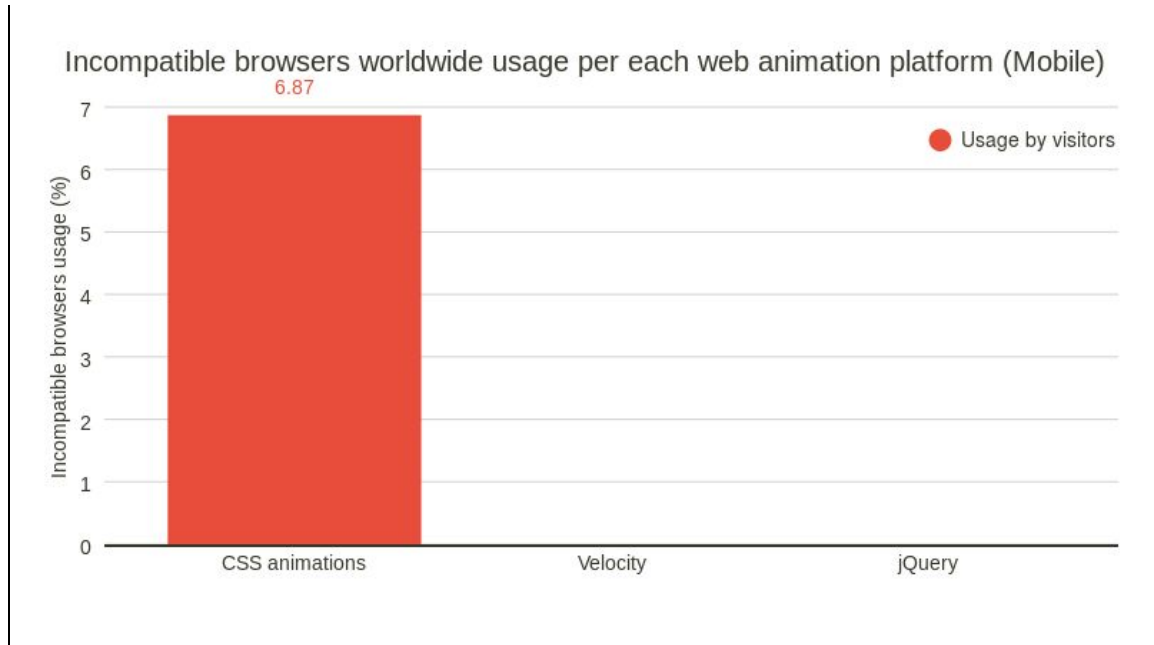


FIGURE 10: Comparing web animations frameworks incompatibility per mobile browsers usage, based on the June 2015 browsers' market share worldwide. (Lower the better)

Source: Netmarketshare.com and table 2, June 2015

6. DISCUSSION AND CONCLUSIONS

The aim of this thesis was to cover a wide range of aspects regarding the design and implementation process of user interface animations. During the research and writing process of the thesis, I have realised that this subject is even wider than I thought initially. I have learned that besides the obvious technical side which related to web development, and the animation-theory part which related to design and classic animation principles, there is a lot of human interaction and human information processing theory which could be studied intensely.

A deeper reading through the existing materials about the psychology of UI animations, brought up more questions in my head regarding this subject. Answering all of these questions would require a lot more pages, time for investigation and would be quite off the original thesis topic. Nevertheless, I have decided to add to my thesis one human information processing research, which focuses on one human behaviour question - does animation help us humans to process information faster.

The research part was the most enjoyable and interesting part for me. First of all, because I could not find anywhere figures which answer this question directly using the methods I have used, and therefore I felt this research brings something new to the field of UX research. The second reason which makes this research interesting was the fact I had to design, implement and alter my experiment solely by myself. This process was a good combination of web development, design, research, gathering feedback from early participants and altering the whole experiment.

As I believe the process is sometimes more important than the results, I felt like I have learnt a lot more while creating this experiment rather than gathering the results, analyzing them and translating them into graphic charts.

Nevertheless, the results themselves were quite interesting as well. As mentioned in the experiment conclusion chapter, the results proved the truthness of some of the theories and statements which has been brought earlier in this thesis, but nevertheless marked some other statements with a big question mark. Does animations make us react or fulfil tasks faster? According to the results - absolutely not, they are even slow us down. Is this tiny delay in reaction worth the extra experience users would get from the transition?

According to many quoted during this thesis, even though have not been exposed to the experiment results yet, I would assume they would say yes.

Animation nowadays are integral part of user interfaces. Partly because of the huge evolvement of mobile devices in past few years, but also because today it is easier than even to create neat and well performed web animations, which fits to modern web standards and does not require the user to download memory consuming plugins such as Adobe Flash.

Nevertheless, many interactive and web producers, such as designers, developers and product managers, still think of UI animations as a “cool” feature but not as a requirement which should be carefully thought through and crafted with care. Therefore, we can see around the web some impressive website which use animation in order to increase their “Wow effect”, but yet lack the basic design thinking regarding what people actually get from the animations as users. Sometimes overuse of animation can lead to performance issues or just user frustration and drop out.

It is important for designers and developers to understand the basic rules of animation - the disney principles, the technical limitation and possibilities and to always remember that in the very end - animations slow us down as humans, and could easily become a disadvantage and a source of disturbance.

LIST OF REFERENCES

- Animation for Attention and Comprehension, Aurora Bedford. (September, 2014) Link: <http://www.nngroup.com/articles/animation-usability/>
- The Illusion of Life: Disney, Thomas, Frank; Ollie Johnston (1981)
- Making Animation Make Sense, Presentation by Mariusz Cieřła. (November 2014) Link: <https://speakerdeck.com/dotmariusz/making-animations-make-sense>
- The Mobile Frontier, Rachel Hinman. 2012.
- The guide to CSS animations principles and examples, Tom Waterhouse. (September 2011) Link: <http://www.smashingmagazine.com/2011/09/14/the-guide-to-css-animation-principles-and-examples/>
- Understanding CSS Timing Functions, Stephen Greig. (April 2014) Link: <http://www.smashingmagazine.com/2014/04/15/understanding-css-timing-functions>
- What is the Optimal Duration of the Animation Interface, Tanvir Hasan. (June 2015) Link: <http://blog.templatemonster.com/2015/06/09/optimal-duration-of-the-animation-interface>
- The Psychology of Human-Computer Interaction. Allen Newell, Stuart Card, and Thomas P. Moran. (1983)
- User interface design and implementation, Robert Miller. (Spring 2011. Massachusetts Institute of Technology: MIT OpenCourseWare)
- Animated UI Transitions and Perception of Time – a User Study on Animated Effects on a Mobile Screen, Jussi Huhtala, Ari-Heikki Sarjanoja, Jani Mäntyjärvi, Minna Isomursu, Jonna Häkkilä. (2010, Nokia Research Center and VTT Technical Research Centre of Finland) LINK: <http://dmrussell.net/CHI2010/docs/p1339.pdf>
- DHTML and CSS for the World Wide Web, Jason Cranford Teague. (2001)
- Thoughts of Flash, Steve Jobs. (April 2010) Link: <http://www.apple.com/hotnews/thoughts-on-flash>
- Summary of Apple's Damages Calculations, Invotex Group. (August 2012) Link: <https://cases.justia.com/federal/district-courts/california/candce/5:2011cv01846/239768/1597/2.pdf>
- Websites Using Shockwave Flash Embedded. Link: <http://trends.builtwith.com/framework/Shockwave-Flash-Embed>

Surfin` Safari, Dave Hyatt. (October 2007) Link:
<https://www.WebKit.org/blog/138/css-animation/>

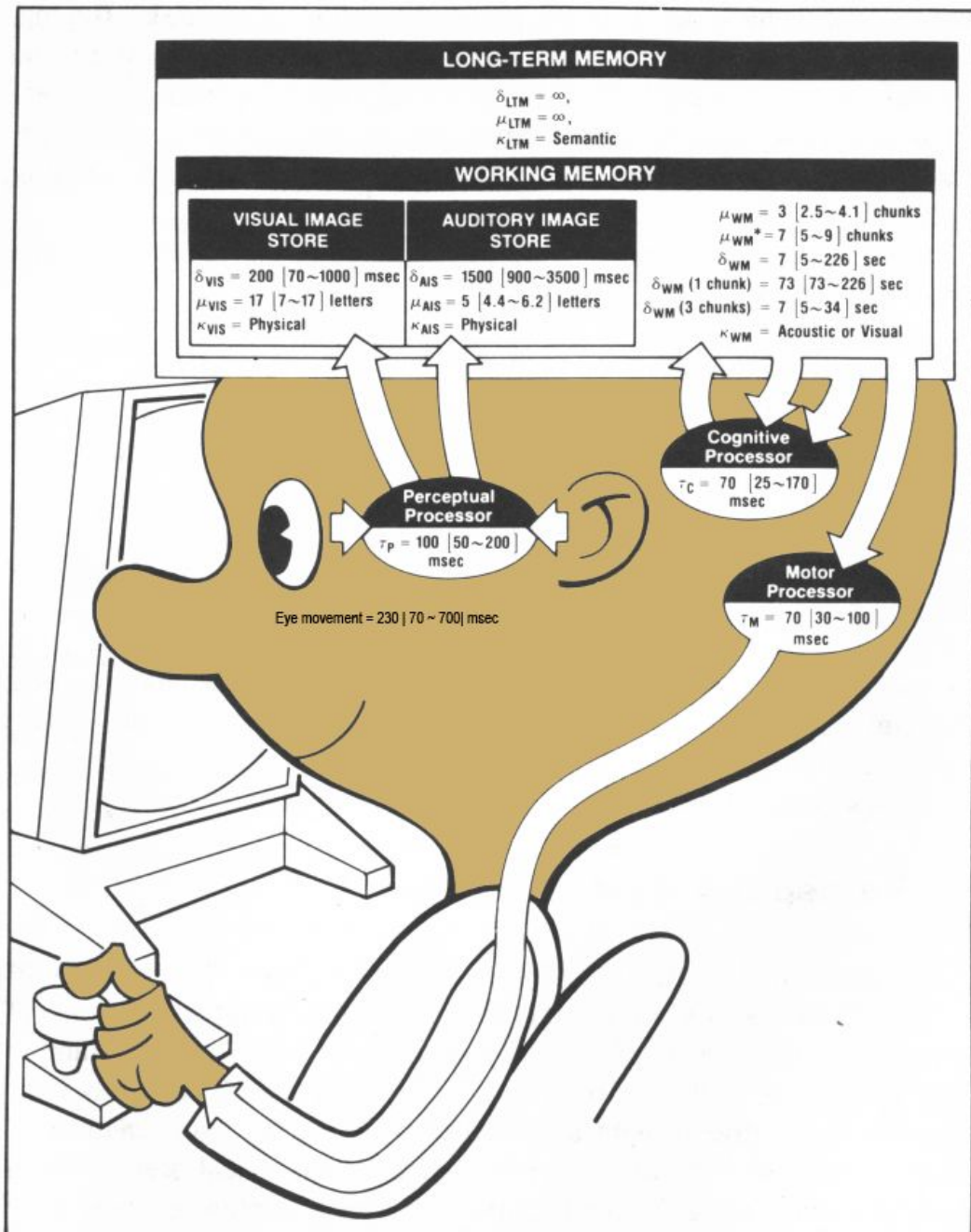
Getting the message across: Ten principles for web animation, George R S Weir Steven Heeps. (University of Strathclyde) Link:
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.185.135&rep=rep1&type=pdf>

StatCounter vs NetMarketShare, Craig Buckler. (April 2015). Link:
<http://www.sitepoint.com/browser-trends-april-2015-statcounter-vs-netmarketshare/>

APPENDICES

Appendix 1: Model Human Processor

In chapter 3.9 (“Timing”), figure 1 (“Human Information Processing”) is based on the following figure from the book “The Psychology of Human-Computer Interaction” by Allen Newell, Stuart Card, and Thomas P. Moran.



Appendix 2: Task fulfillment speed experiment specs and details

A2.1 Participants demographics

A2.1.1 Participants Locations

Participants location has been collected by tracking their IP address and parsing it into a rough geolocation. Please note that this method indicates the participant's physical location only, and does not determine or imply their birthplace or their cultural influences over the years.

Country	Participants	Country	Participants
United States	62	Hong Kong	1
Finland	25	New Zealand	1
United Kingdom	21	Spain	1
Germany	18	Japan	1
Israel	18	Argentina	1
Canada	15	Russia	1
Australia	9	Kenya	1
Netherlands	6	Switzerland	1
Czech Republic	5	Estonia	1
Italy	3	Belgium	1
France	3	Austria	1
Greece	2	Mexico	1
Romania	2	Sweden	1
Latvia	2	Unknown	1
Vietnam	1		

A2.1.2 Participants Age

In the beginning of the test, participants have been asked which age group they belong to.

Age Group	Participants	Age Group	Participants
Under 20	18	40 to 50	5
20 to 30	59	50-60	5
30 to 40	12	Unknown	108

A2.2 Tests briefing

A2.2.1 Welcome page briefing

“Hello! Thanks a lot for participating in this experiment. I promise it will take no longer than 5 minutes of your time! First of all, I would like to know which age group you belong to. This is of course anonymous information for research and analytics reasons only. I will not tell anyone how old you are!

[Under 20](#) [20 to 30](#) [30 to 40](#) [40 to 50](#) [50 to 60](#)

In the next steps I am about to ask you to fulfill a couple of tasks. I must warn you that these tasks you are about to do might look a bit odd. However, they'll be easy-peasy.

Try to fulfill them while you have your full attention at the screen and sitting comfortably in front of a desktop or laptop machine. Mobile devices are unfortunately not supported.

In the first task, images of animals will appear on the screen. Please click on the button which says the animal name right as the images appear without hesitation.


Ready? [Let's Start](#)”

A2.2.2 “Buttons test” briefing

“Awesome! In the next task, buttons will appear on a random location on the screen. Please click on each button right as it appears without hesitation. Please try be fully focused on the test and avoid any external distraction.

Ready? [Let’s Start](#)”

A2.2.3 “Icons test” briefing

“Hooray! In the next task, 3 icons will be shown on the screen. I would ask you to click on the home () icon right as the icons appear without hesitation. Please try be fully focused on the test and avoid any external distraction.

Ready? [Let’s Start](#)”

A2.3 Technical implementation and distribution

The test was distributed as a JavaScript web-application, customly built by the author of this thesis work with Meteor.js framework. The implementation consisted a combination of JavaScript, HTML and CSS for the frontend, CSS animations for the transitions, Node.js as a web server and a MongoDB database. The app was deployed to Meteor servers and served at the following domain: yonatan-thesis.meteor.com.

The experiment was distributed worldwide via email, social media, forums and online groups, primarily on Facebook, Reddit and Twitter.

A2.4 Statistical exclusions and corrections

The tests was tweaked a few times at the first few days after it was already up and running, based primarily on the first 50 participants feedback and the results. The results of these 50 trial participants were excluded from the final analysis and statistics.

For each user, the first test sample of each test type (“animals”, “buttons” and “icons”) was considered as a trial test sample. While assuming that the trial test would naturally take longer to understand and complete, it was excluded from the final analysis and statistics.

For the “animals test” and the “icons test”, wrong answers (i.e clicking and the cat button while a cow image appears) were excluded from the final analysis and statistics.

Test samples which took more than 3500 milliseconds (3.5 seconds) to fulfill were considered as idle and were excluded from the final analysis and statistics.

Appendix 3: CPU usage test specs

Machine used for testing:

MacBook Pro. 2.5GHz quad-core Intel Core i7 processor (Turbo Boost up to 3.7GHz) with 6MB shared L3 cache.

Tested with Google Chrome browser, and measured by Chrome's "Task Manager".

Code snippets tested are based on the examples in chapter 5.2.1 in this thesis.