Web UI Animation

Group 5

Rok Kogovšek, Alexei Kruglov, Fernando Pulido Ruiz, and Helmut Zöhrer

706.041 Information Architecture and Web Usability WS 2016
Graz University of Technology
A-8010 Graz, Austria

04 Dec 2016

Abstract

TO DO

Writing a survey can be a traumatic endevour. It might be a student's first foray into academic research. There are often obstacles and false dawns along the way. This survey paper takes a fresh look at the process and addresses new ways of accomplishing this daunting goal.

The abstract should concisely describe what the survey is about. State the areas which are covered and also those which are not covered. Market your survey to your readership. Also, make sure you mention all relevant keywords in the abstract, since many readers read *only* the abstract and many search engines index *only* the title and the abstract.

This survey explores the issues concerning the writing of an academic survey paper and presents numerous novel insights. Special attention is paid to the use of clear and simple English for an international audience, and advice is given as to the use of technical aids to production.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

[TODO: An academic survey paper presents a survey or overview of the state of the art in a particular field. Every chapter and every section should have some introductory text at the beginning, like this text. Never jump straight in to the first secion or subsection without one or more paragraphs of introductory text.]

1.1   Not a Series of Summaries

A survey is *not* simply a series of summaries of papers. If I have given you say 8 papers to start you off, what you should *not* do is: divide up the papers (read two each) and produce a series of 8 unconnected paper summaries.

1.2   Read All the Papers and Research Some More

Each of you should read *all* the papers and resources: both those I gave you and those you found yourselves. Make sure you search for more papers and resources yourselves. Not just a Google search. Search the ACM [**ACM-DL**] and IEEE [**IEEE-DL**] digital libraries, citeseer [**CiteSeer**], and mendeley [**Mendeley**]. You may want to use mendeley to collect your resources or maybe maintain a .bib file within an SVN repository.

Include a list of *all* the relevant papers and resources you have found and mark those you have chosen to focus on. Make sure *all* the papers and resources you found or were given appear in the bibliography.

1.3   Dividing up the Field

The hardest part of any survey is dividing up the field. Look for common concepts and threads in the papers and resources. Do they report similar or dissimilar results? Does one paper or resource support or contradict another?

Once you have all read all the papers: you need to construct a small hierarchy (taxonomy) to classify the concepts appearing in the papers and resources. Structure your survey into chapters and sections based on your taxonomy.

Chapter 2

Animation

"Animation is defined as changing some property over time. On the other hand, motion is the act of moving or the process of being moved. . . . To put it more simply, all motion is animation, but not all animation is motion."[Head, 2016]

2.1 Not Just Motion!

In animation we change an object's attribute(s) over time to achieve an objective. As stated by Head [2016], animation is much more than adding movement to an object. Movement of an object can be described by a change of its 6 degrees of freedom, them being the coordinates in 3D space and the rotation around the 3D space axises. However an object has usually many more changeable attributes or properties. One could animate color changing and fading, invisibility through transparency, growth with scaling, focus with blurring, etc.

2.2 Why Animation in Web UI?

While writing this survey, with so much animation theory, a hilarious idea was just coming up to mind. Have you ever stopped to think about why some people just seem to be more funny than others? And of course, this happens everyday. Just think about a good joke told by someone with no real humour. It is meaningless. Now, think about the same joke being told by a profesional comedian on TV telling jokes over the weekend. Way more funny, is it not? But the point is, why and how is this be possible? It has lots to do with how they speak and how they move! As well of other minor factors. That, I am afraid, is the same as with animation. To have a better understanding of what factors are involved and how they can improve the user experience in UIs, we look deeper into it in this section with Head [2016] as the reference point.

Animation, be it in a cartoon, webpage, app or anywhere else, adds a new level of communication which can be hard to explain , but easy to feel. It creates a special connection between what the message is intended to be, and the receiver. It allows an additional cannel of invisible information to exist, and makes the viewer feel part of the process, and gets his attention grabbed in a practical way.

We always need to understand, where we are and where are we going to. Animation can helps us with this and even can improve the use of resources. Just think about when trying to navigate through a small screen. Having some animated window coming out when you put your mouse on, does not only save space (which is already a good enough reason to implement it), but makes the entire process of navigating easier and simpler. A user does not really want to think while navigating, and this can be fulfilled with animation. They can be guided, orientated or helped in going through a process just with animation, which will of course improve their user experience and make them feel more sure and comfortable about what they are doing, without thinking they are in a real puzzle and need to make big use of their brains.

Putting it all together, we can effectively ensure that a well planned animation will improve decision making, as it will guide the user through the entire process, allowing him to be more aware about the real process or

task he or she is willing to achieve, and not about the real animation. Of course, this can be enforced by the fact that the user will feel listened and will probably trust more the site because of the animation being interactive, giving in some way feedback, making the user feel comfortable when its input is being taken into consideration.

Nowadays, it´s very common to access many sites through different devices. Just think about when trying to know any train timetable. You probably have the app in your Smartphone, but depending on where are you at the moment, you might just google it up and search it through their webpage. Having animation allows sites to connect context and media. A user will just find himself guided throught the process no matter where he is or what device is he using, because of being under some potential animation process.

But, how do these animations grab people´s atention? There are over twelve basic principles, written by Disney, but two of them are like the foundation of animation, and the rest basically build up from them. Timing and spacing. Timing can be understood as the time length of an action to happen, let's say the duration of it, while spacing could be explained as the speed changes during one of these actions in an animation. Head [2016] (Chapter 2, page 18) describes their impact with: *"Timing and spacing convey the mood, emotion, and reaction of an object."*

2.3   Aim For Invisible Animation

In animation, it is very common during the designing process to end up messing things up. As a general rule, as a designer, your one good way to show love to the user is actually not making his or her life harder accomplishing the task at hand. Certainly designers do not hold such ill intentions, but as said before, it is quite easy and happens very often just to have users unnecesarily waiting for an animation to finish or having a hard time finishing a task due to animation. It could be said that the perfect animation is the one which is not noticed. That´s not 100% accurate, but a good idea in general.

*"Good interface animations need to be flexible and always feel responsive to a user's input even if the animation is currently animating."*Head [2016] (Chapter 3, page 48)

Users need feedback to feel themselves listened and understood. Animation can be understood as a bridge which takes the user to a higher quality experience. If an animation doesn't respect having the user happily informed with feedback, its experience quality will of course drop and so will the trust towards the UI. A user will never feel comfortable in a system which does not take into consideration its input.

It is also important, that an animation should never be a show-off work. It should be always remembered, that in general, the user interacting with an animation, is probably willing to do or finish something. A UI user is not just laying there waiting for an animation to happen and look at it itself and enjoy it. That could be the aim of animators years ago where they were really telling stories and people were fascinted about it. But times changed, and now the timing requirements are no longer the same as in those days. Study and carefully think about the suitable timing for each animation, because a short amount of milliseconds can be the limit between failing or succeeding. Good timing is more an art than a science [Head, 2016].

2.4   Development of animation

There are lots of guidelines to help developers through the development of an animation, but what about some guidelines for the actual development itself? When coming up to a project, the animation issues are often discussed at the end of it. It is quite normal to just understand animation in a project as extra decoration, as something which is not mandatory, which will just make the whole running project look a little nicer. This should not be the way of proceeding, but it is the reality of many projects. Animation should be part of the design, due to the potential for a better user experience, like described in section **??**. Just bare in mind that when the project is finished, it is usually too late to replan the design involving animation at its full potential, and this leads to many projects to just leave beside some good animation ideas. Trying to makes things easier for the user (that is what animation is all about) at the end of a project, when everything is quited fixed, doesn´t sound
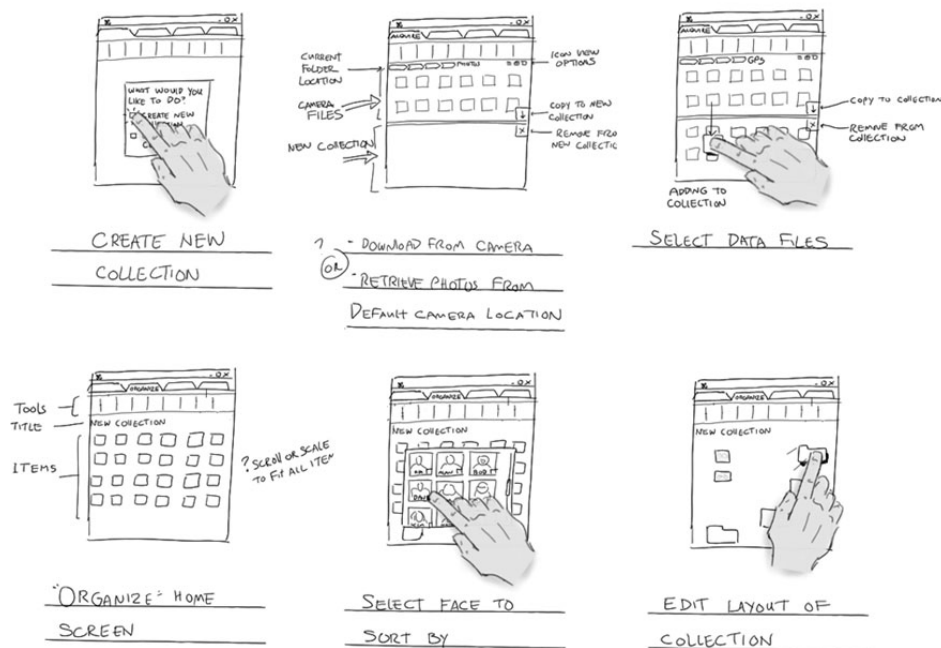
Figure 2.1: Example of storyboard sketching for drag and drop animation [Windows Dev Center, 2016].
[Used with permission from Microsoft - Microsoft Copyrighted Content Guidelines]

like a good plan. Always start planning animation schemes in early stages! It is always better to have a pause after each step of the project design to talk about animation, than to rather avoid it at all because its already too late! Remember that animation can replace a fixed design element which might make the user have a bigger quality of the experience, or less brain demanding. And of course, as taken as a part of the project design, style guides and documentation about the used animation should be reported. And no need to say, understanding the basics of animation principles, and how they can influence the user, and bearing in mind the accessibility issue ... as a conclusion, just make sure you don´t forget any of the points mentioned in the earlier sections of this survey. When coming up to the how to plan it, probably the best way in early stages is storyboard sketching. This is a great, simple and quick way of more or less understanding what we are doing (or planning to do). When moving further, however, building an animation prototype inside the design prototype can be very suitable before moving on to the next stage, as it can allow designers to really test and see if what they´re doing its actually as good and helpful as they think.

Chapter 3

Cascading Style Sheets (CSS)

Knowing the usefulness of animation in web UI and the correct way of animation planing are just the fundamantals for our conceptual plans. Those still need to be implemented to get the end product and here we usually hit a wall build from the various tools, that say they can all solve our problems. Even well established people in the field have stories as such to tell. Val Head actually started with animation due to an interesting Flash workshop. Flash was at that time the de facto king in its era, however as we know, that era is already dead. Nowdays we can acomplish all we could with Flash and more with just the core parts of the web, namely HTML, CSS and JS [Head, 2016].

3.1   Do Everything You Can With CSS

With Responsive web design (RWD) in our websites and animation being part of the design, see section 2.4, it should be natural to use the guidelines of RWD also in animation planning. Andrews [2016] teaches us that one of the RWD guidelines is also Progressive enhancement, which is best described with words of the conceptual authors Champeon and Finck [2003]: *"Leave no one behind. . . . accessibility is for everyone, not just the disabled".* With CSS nowdays being a core part of the web and at the same time being the lowest web component that enables animation with RWD guidelines[1], one should always implement with CSS and HTML alone as much of the desired animation as possible. One has only to make sure the browser support for the animated attribute.

   Other supporting arguments for use of CSS as the starting point for web UI animation beside responsiveness can be summarized with the the so called "Simple CSS Truths", a list of truths by Palermo IV [2015] enhanced with the teachings of Andrews [2016]:

CSS allows for separation of concerns - With CSS the form is separated from the page's HTML structure and content. Makes it easier to read, maintain and crawl the code.

CSS has a captive audience - Support for CSS development is huge. At the same time more and more libraries, tools and frameworks focus on improving and simplifing CSS development.

CSS is fast - External CSS speeds up HTML downlaod and loading compared to HTMLs with duplicated inline styles. Compared to JavaScript it also processes transitions and animations faster.

CSS is fault-tolarent - Browser-unknown enhancements are simply ignored by the browser, while the remainder is still used and displayed.

CSS is everywhere - Modern browsers embrace CSS and feature support by each can be easily found online.

---

[1]Of course one can just use an animated image, e.q. a GIF with an image sequence, and just append it with HTML into the design. However, this image will become a static component of the design and will not follow RWD guidelines.

3.2   CSS Animation Declaration

As stated in section 2.1, animation is about changing an element's attribute(s) over time. In CSS we can redefine it as a switch between CSS styles for a HTML element that happens gradually over time. It is stated that CSS animation should be done with *Animation* property(ies) and *Keyframe* rule(s). This may be the most efficient CSS way to accomplish animation, but CSS animation can also be achieved with *Transition* property(ies) and *selector* pattern(s)[W3Schools, 2016a; W3Schools, 2016b].

3.2.1   Animation Property and Keframe Rule

The prefered declaration for animation in CSS is done by setting animation properties to the element, that will change through time. This properties just define to which animation steps or keyframes the element is linked to and how the in-between-keyframe states are interpolated through time. W3Schools [2016a] explains the different properties as follows:

animation-name: Here the @*keyframe* rule name is used to link the rule to the element. Since CSS animation is a set of animation properties and keyframe rules, it is a neccasery property.

animation-duration: Like-wise to the name, duration of the animation is also a neccasery property, since it defaults to 0. Animation is defined as a change over a time, so with the time duration being zero, of course there is no animation. Actual working values can be either seconds (#s) or miliseconds (#ms), that define one cycle of the animation.

animation-timing-function: Defines the progression curve over time, that defines how the animation is interpolated. The valid values are linear, ease, ease-in, ease-out, ease-in-out, cubic-bezier(x1,y1,x2,y1) and steps(stepSize, start, end). Acording to Head [2016], most animators have a hard time imagining the effect of functions that start with ease*. While linear and steps are simple to understand and enough for simple animations, the cubic-bezier function should be used, since it can define any desirable curve. The way the cubic-bezier function works, can be observed in figure 3.1.

animation-delay: Defines how many seconds or miliseconds should the animation start be delayed. The default value is 0. It is useful, when we want multiple elements be animated one after another.

animation-iteration-count: We can define with a number, how many loops of the cycle should be done or set it to infinite cycles for a non-stop replay. The default value is 1.

animation-direction: With this property we describe in which way the animation is interpolated and what happens after the 1st cycle of animation. By setting it to the default normal, set it to reset the element into the state before animation and start it again. Reverse always starts from the end keyframe and goes towaqrds the starting keyframe, while alternate will alternate between normal and reverse by the oddness of the cycle number. Another option is the self-explanatory alternate-reverse.

animation-fill-mode: Defines what happens to an unplayed element, be it after finishing the animation or during a delay wait. By the default none the element has the CSS style outside the keyframe rules. Forwards will use the style of the last keyframe, while backwards will use the style of the first keyframe. A special option is both, that uses the styles of both the start and end keyframe.

animation-play-state: It is mostly a property used for testing and with control triggers. As the property name states it is either paused or running.

Same as with other CSS properties animation properties can be combined into a single property, simply called animation and where the property values follow as stated:

animation: name duration timing-function delay iteration-count direction fill-mode play-state;
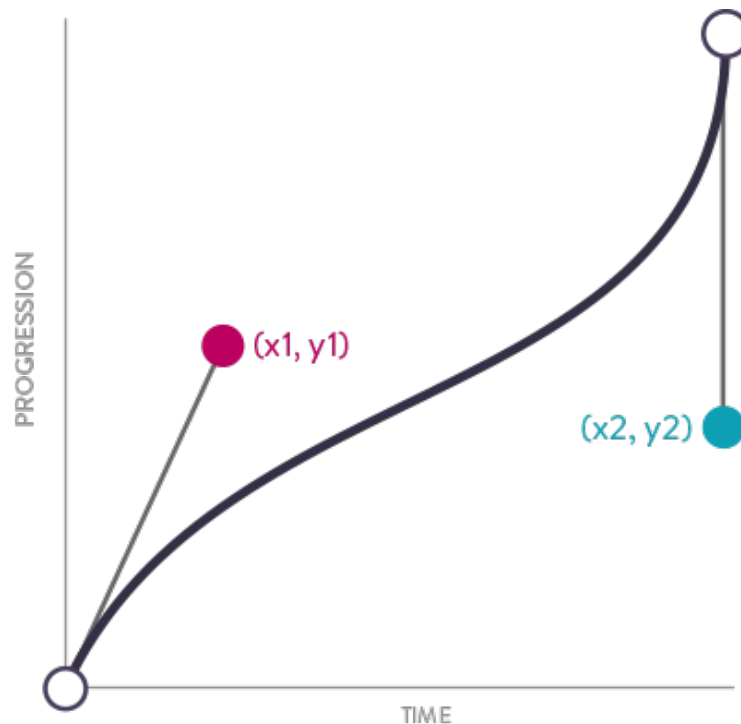
Figure 3.1: With coordinates for the location of the start and end weight we can define the desired function curve for progression through time by how much each half of the function should be deformed away from a linear function [Head, 2016]. [The image is from the book "Designing interface animation" by Head [2016], used under CC BY 2.0 / It is accessible at *https://www.flickr.com/photos/rosenfeldmedia/albums/72157671107313626.*]

While the animation properties just link the element to the desired changes, the actual CSS changes are defined in the @keyframes rule. In the rule we can set the keyframes, that are the finishing times, when a change should end. The keyframe times are simply described with percantage from 0% as the start keframe and 100% as the end keyframe. To this times we simply set, which styles should be used at the appointed time. The interpolation function set in with animation property will during proccessing interpolate the undefined percantages. For a better understanding see listing 3.2.1.

```
 1  div{
 2    animation: desiredName 4s linear 0s infinite alternate;
 3  }
 4
 5  @keyframes desiredName {
 6    0%    {background-color:red;opacity: 1;transform: scale(1);}
 7    25%   {background-color:yellow;transform: scale(0.8);}
 8    50%   {background-color:blue;transform: scale(1);}
 9    75%   {background-color:green;transform: scale(1.5);}
10    100% {background-color:red;opacity: 0.2;transform: scale(2);}
11  }
```

Listing 3.1: Simple example of non-motion animation with animation property and keyframes rule. A working example can be found between the attached code examples under nonmotionAnimationCSS.html.

3.2.2   Transition Property and Selector Pattern

The transition property can be used the same way as th animation property for defining the linking and inter-polation details. There are even similarities int hte properties:

transition-property: With this property we can link the the transition to a change in CSS, similar as with animation-name. However here we specify which of the CSS properties of the element should be inter-polated during the change. It can be one or more properties seperated with a comma. We also have the default option none and convenient all, that sets all propertis to interpolate.

transition-duration: Same as animation-duration property.

transition-timing-function: Same as animation-timinig-function property.

transition-delay: Same as animation-delay property.

The unified property is also defined by the same order, with just less options:

transition: property duration timing-function delay;

One can imagine the transition being a simplified version of the animation property, that can only have two keyframe, namely the start and end of the aniamtion or transtion. The start keyframe style is defined in the CSS style of the unchanged element, while the end frame is defined in a declaration where the specified transition property is changed. The easiest way to control such animations is with event seperators, such as :hover. This way we can define an animation to happen after hovering over the element. Since we can define just two keyframes and there ano repetitions, we are quite limited with the possible animations, which is also the reason for the animation property and keyframes rule to be used most of the time for animations. The limitations can be easily observed, when comparing the codes and results for 3.2.1 and 3.2.2.

```
 1  div{
 2    background-color: red;
 3    opacity: 1;
 4    transition: all 4s;
 5  }
 6  div:hover{
 7    background-color:green;
 8    opacity: 0.2;
 9    transform: scale(2);
10  }
```

Listing 3.2: Simple example of non-motion animation with transition property and hover selector. A working example can be found between the attached code examples under nonmotionTransitionCSS.html.

3.3   CSS Examples

3.3.1   Navigation Animation

As stated in [TODO: ref section useful] , animation can help with navigation through tha page. In the examples we show two typical cases of such usage.
    [TODO:

Figure 3.2: One the left we have a screenshot of code example hamburgerVariationCSS.html, which shows some of the different ways how hamburger icons are animated. On the right side we have hamburgerMenuCSS.html, that demonstrates the showing and hiding of the menu by hovering over the hamburger icon. [Screenshot taken by the authors of this survey. The code behind the pages is by using the Boyce, 2014; Couch, 2014 online snipets as a base.]

Hamburger Icon

]

Rand [2015] [Rand, 2015]
[TODO:

Current menu position indicator

]

3.3.2   Loading Animation

Another use of animation is user feedback, see [TODO: ref section useful] .
[TODO:

Rotating Icon

] 2D and 3D
[TODO:

Horizontal movement

]

dots, progress bar, pulz (wave motion)

3.4   Scalable Vector Graphics (SVG)

[TODO: section by A.K.]

Chapter 4

JavaScript (JS)

[TODO: CHAPTER INTRO IS MISSING - check chapter 2 and 3]

4.1 When to Use JS instead of CSS

As a rule of thumb, JS should not be used if the same effect could be achieved with plain CSS. This is due to performance and resource management reasons. Just when CSS is stretched to its limits, JS should be used. There is a rough distinction whether to use CSS or JS for particular kinds of tasks:

- Use CSS animations for simple transitions, like changing the state of a UI element.

- Use JS animations to get advanced effects like bouncing, stop, pause, rewind, or slow down (there is more control over animations).

- When choosing to animate with JS, considering using the Web Animations API or a modern framework (comfortable to work with) may be desirable.

- Using both CSS and JS works also well:

    - perform animations with CSS
    - control states with JS

4.2 Examples of Useful Animations with JS

[TODO: this!]   [TODO: citing with: Lewis and Thorogood [2016] or [Lewis and Thorogood, 2016]]

```
1  // create some nodes
2  var headline = document.createElement('h1');
3  var text = document.createTextNode('Dies ist eine Überschrift');
4  // "offline" node manipulation
5  headline.appendChild(text);
6  // adding node to DOM
7  document.getElementsByTagName("body")[0].appendChild(headline);
```

Listing 4.1: This is a code snippet where JS is used in a meaningful way.

Chapter 5

Concluding Remarks

Bibliography

Andrews, Keith [2016]. *Information Architecture and Web Usability*. 2016 (cited on page 7).

Boyce, Robert [2014]. *Paper Nav*. English. CodePen. 2014. `https://codepen.io/rboyce/pen/wlijs` (cited on page 11).

Champeon, Steven and Nick Finck [2003]. *Inclusive Web Design for the Future*. 2003. `http://hesketh.com/publications/inclusive_web_design_for_the_future.html` (cited on page 7).

Couch, Jesse [2014]. *Menu "Hamburger" Icon Animations*. English. CodePen. 2014. `https://codepen.io/designcouch/pen/Atyop` (cited on page 11).

Head, Val [2016]. "Designing interface animation" [2016] (cited on pages 3–4, 7–9).

Lewis, Paul and Sam Thorogood [2016]. *Web Fundamentals. CSS Versus JavaScript Animations*. English. Google Developers. 2016. `https://developers.google.com/web/fundamentals/design-and-ui/animations/css-vs-javascript` (cited on page 13).

Palermo IV, J. Michael [2015]. *CSS in the Modern World*. English. Microsoft. 12th Nov 2015. `https://www.sitepoint.com/css-modern-world` (cited on page 7).

Rand, Jesse [2015]. *The Genius—and Potential Dangers—of the Hamburger Icon (Flyout Menu). Meet the Hamburger Icon*. English. vital. 2015. `https://vtldesign.com/web-strategy/website-design-development/hamburger-icon-flyout-menu-website-navigation` (cited on page 11).

W3Schools [2016a]. *CSS3. CSS3 Animations*. English. W3Schools. 2016. `http://www.w3schools.com/css/css3_animations.asp` (cited on page 8).

W3Schools [2016b]. *CSS3. CSS3 Transitions*. English. W3Schools. 2016. `http://www.w3schools.com/css/css3_transitions.asp` (cited on page 8).

Windows Dev Center [2016]. *Hilo: Developing C++ Applications for Windows 7. Designing the Hilo User Experience*. English. Microsoft. 2016. `https://msdn.microsoft.com/en-us/library/windows/desktop/ff800706.aspx` (cited on page 5).