UNIVERSITY OF MUNICH
Department "Institute for Informatics"
Education and Research Units Media Informatics
Prof. Dr. Heinrich Hußmann

Master Thesis

Web-Based Creator for Activity Sculptures

Walter Rempening-Diaz
me@walterrempening.com

Working Time: 1. 12. 2014 to 1. 6. 2015
Supervisor: Simon Stusak
Responsible Professor: Prof. Dr. Andreas Butz

Acknowledgements

Abstract

The recollection of personal activity data has been greatly facilitated by the increasing amount of applications and devices that encourage users to measure their activity with the primary goal of health improvement. These devices range from mobile applications taking advantage of smartphone sensors to dedicated fitness trackers presented as modern watches and bracelets. Apart from the analytical insights about the data obtained through classic data visualizations, it is also possible to visualize the information through physical objects also known as activity sculptures. It has been shown that activity sculptures have a positive influence in users making them feel rewarded for their active lifestyle. To further study the process of visualizing activity information into sculptures an web-based activity sculpture creator was developed. This tool takes advantage of modern web technologies and offers a platform in which users can export their data and allows them to experiment creating variations of an activity sculpture which can also be exported for 3D printing. For the development of the configurator current product customization platforms where analyzed for gathering best practices in user interface and interaction design. In order for users to have a sculpture with a high degree of variability for the data to be mapped on 4 different sculpture prototypes were developed. For the validation of the configurator an online version was released and a user study was performed. User feedback showed that our prototype was easy to operate and that the obtained sculptures were appealing and meaningful them.

Zusammenfassung

Die Sammlung persönlicher Aktivitätsdaten wurde durch die zahlreiche Anzahl an Anwendungen und Geräte enorm vereinfacht. Diese Anwendungen und Gerätschaften, die hauptsächlich das Ziel haben, Nutzer zu einem aktiven Lebensstil ermutigen, können in Smartphones, wo sie die Vielfalt an Sensoren ausnutzen oder als tragbare Accessoires wie moderne Uhren oder Armbänder gefunden werden. Abgesehen davon, dass klassische Datenvisualisierungen Einblicke in den Aktivitätsdaten verschaffen können, ist es auch möglich den Datensatz durch physikalische Objekte, auch als Aktivitätsskulpturen bekannt, zu visualisieren. Es wurde bewiesen, dass Aktivitätsskulpturen Nutzer positiv beeinflussen, da die Nutzer sich für ihren aktiven Lebensstil belohnt fühlen. Um den Prozess der Visualisierung von Information in Skulpturen weiter zu forschen wurde ein Web-Konfigurator für Aktivitätsskulpturen entwickelt. Durch die Nutzung moderner Web-Technologien erhält der Nutzer eine Platform die ihm es erlaubt seine Daten unkompliziert zu exportieren und ermöglicht ihn die Gestaltung einer 3D druckbaren Skulptur. Für die Entwicklung des Konfigurators, wurden aktuelle Konfiguratoren analysiert mit dem Ziel Best-Practices im Bereich des Interface- und Interaktionsdesigns zu erkennen. Um den Nutzer eine breite Vielfalt an möglichen Anpassungen für die Skulptur, wurden 4 verschiedene Skulptur-Prototypen entwickelt. Letztendlich wurden für die Validierung des Prototyps eine online Demoversion veröffentlicht und eine Nutzerstudie durchgeführt. Die Resonanz der Nutzer zeigte, dass unser Prototyp einfach zu bedienen war und, dass die entstandene Skulptur ästhetisch und sinnvoll rüberkam.

Task Definition

Activity Sculptures are physical (3D printed) representations of personal tracking data (e.g. step count) that dwell between the artistic and the abstract. For this master's thesis the student will develop a web configurator that will allow to individually create said activity sculptures (a similar example can be seen in www.shapeways.com/creator/statement_vase).

The focus of the thesis will be the development of interaction concepts and their implementation in the configurator. The concepts will be examined and improved in smaller iterative user studies. Another important aspect is a seamless and easy import of external tracking data (e.g. export data from tracking apps). The result should be a stable working prototype that can be used for follow-up works.

Possible research questions

- What interaction concepts are possible? What are their advantages and disadvantages?
- What degree of freedom is possible and meaningful while designing a visualization?
- What is a possible design space for said activity sculptures?

Tasks

- Research and related works (e.g. data visualization, configurators)
- Development of interaction concepts
- Concept implementation
- Planing and executing several small user studies
- Written thesis and presentation of work

Requirements

- Programming skills in web development and computer graphics

I confirm that I independently prepared the thesis and that I used only the references and auxiliary means indicated in the thesis.

Munich, June 3, 2015

. . . . . .

Contents

1 Introduction

The presented work deals with two major topics: web customization platforms and activity sculptures. For the former topic interaction processes and usability aspects applied in current projects are of great interest as they provide a foundation on which the author's prototype will be built upon. The latter will help the user explore and engage with their activity data in a meaningful way, first in virtual and later in physical 3D space. The following sections offer an background information to the topic and a define the problem addressed in this work. To conclude this chapter a general overview of each chapter will be provided.

1.1 Background & Problem Definition

At its core, the problem to solve in this work is a data visualization problem. The field of data visualization is tightly interlaced with other fields such as statistics, psychology, design, human-computer interaction, computer and cognitive sciences to name a few. This makes it a science that requires a broad set of skills to master. Due to its multidisciplinary nature it is also difficult to define. A definition of data visualization suitable for this work would be the one described by Card et al.[7]:"The use of computer-supported, interactive, visual representations of abstract data to amplify cognition". In other words this definition states that the existent knowledge about a specific dataset can be increased by mapping the data to a visual space and by interacting with it through a computer system

The interrogatives and curiosities about a dataset are the start point of every visualization. This work aims to answer some questions about activity data and the possible forms of representing it in a physical sculpture. But why activity data? We are living in a time where there is more data flowing every second through the internet than all the data stored in the internet 20 years ago[40]. It is estimated that the measure of digital created data will grow by a factor of 300 from 130 exabytes in 2005 to 40,000 exabytes in 2020[16]. This forecast shows how the analysis of big data is playing an important role in the way how decisions are made in the industry. The large scale nature of big data observing how millions of users behave can also be put into a much smaller scale namely the quantified self. Big data and the quantified self could be compared to a telescope and a microscope. Both use the same principles to amplify the ability of humans to observe but on totally different magnitudes. The quantified self is a movement of people that make use of self-tracking tools to measure physical performance or vital signs for self improvement and was first described by Wired editor Gary Wolf[68]. Noticing the great momentum the movement was having Gary Wolf founded together with his fellow editor Kevin Kelly the Quantified Self Labs[69]. Even though people have been tracking themselves through the centuries and the improvement of performance by solely knowing one is being observed has been also been studied[41], the technological improvements in sensor development has made the process of gathering activity data much easier. So much that anybody with a smartphone can start tracking his own activity. Advocates of the quantified self movement see in this practice a tremendous potential for solving health challenges through big data[56].

It is therefore of great interest to develop new visual representations for the increasing amount of personal data available. The challenge for these visualizations is that they can engage people in a more deeper level as the data treated is a reflect of their own behavior. For this purpose designers and engineers have been taking advantage of digital fabrication systems, in particular 3D printing, to translate their visualizations from the screen to physical space. Activity sculptures have shown to be a suitable visualization for communicating abstract data into a tangible object. While it can be discussed about the usefulness of such an object is mainly determined by the information it can convey[70], activity sculptures have shown to influence people's behavior positively[36]. Other valuable characteristics of activity sculptures is the interaction possibilities and their physical properties[70]. Simply by being physically constructed they can approach the natural instinct

of interacting with objects through touch, by feeling its material and exploring its surface and structure.

As discussed before, one key aspect in the process of modern data visualization is that the interaction with the visual representation of the data occurs on a computer-aided system. One approach that has proved to be efficient in aiding users to manipulate objects to their specific need is the configurator. This tool supports the product configuration process satisfying every design and configuration constrain set by the manufacturer[21]. The development of configuration systems originated from the mass customization paradigm, that establishes a business model in which companies massively manufacture individually customized goods[12]. This paradigm is divided in different methods that offer each different degrees of freedom to customers. Out of this method the collaborative customization or the co-creation method uses of software tools that allow the customer to transfer their preferences directly to the product[49, 48]. Most of the configurators are deployed as web applications, in large due to the scalability and accessibility of web systems. Advancements in web technologies allow manufacturers to build more complex configuration systems. Because the configurator's degree of user friendliness can have a positive or negative impact on the completion of a sale, it is important to companies to understand how to guide the customer in a meaningful way to complete the sale[52, 2]. The work-flow proposed in research is usable to ensuring the successful guidance of the user throughout the process of developing a product suited to his needs. This same knowledge could be applied to the development of a visualization system.

To summarize this section, the motivation of this work could be resumed as follows. The increasing desire to quantify every aspect of the self and the advancements in digital fabrication open the field for a new kind of visualizations residing in physical space out of the constrains of the screen. This is a challenge for the data visualization field that can be addressed through co-creation production techniques and best practices to develop a software tool that includes the user in the process of achieving a visualization that is aesthetically and functionally meaningful.

1.2 Goals

The main goal of this work is to develop a system that can guides intuitively the user in each process of the visualization of his activity data. This all includes importing the activity data of the user and processing it in order to be visualized in a sculpture which will be further manipulated to users preferences and exporting it for 3D print. The aim of the web configurator is to perform all these tasks providing the user the best experience possible. For this the development of interaction concepts, that guide users through each step of the configuration process, plays an important role in the achievement of an enjoyable platform. Another goal is to ensure the interaction concepts in the prototype are understandable and easy to grasp. In order to achieve this goal users feedback was taken into account through user studies and questionnaires. The diversity of users chosen for the studies was made possible through local testers and through an online demo of the prototype. Furthermore the design of an activity sculpture that shows high variability in the configuration possibilities was an objective kept in mind throughout the prototyping phase. In order for the system to respond fast to user input a special set of technologies was needed. This work aims to take advantage of current edge technologies by implementing them in the prototype.

1.3 Content overview

The presented work takes the following structure. Chapter 2 presents current configurators in different fields of the industry and academic research. Further on current projects related to activity sculptures will be discussed. The final section of the chapter presents an analysis of activity data sources and current implementation of available fitness tracker APIs. In Chapter 3 the prototype design process will be presented. For this sketches and concepts for both sculptures and the configurator will be explained concluding with final thoughts about the final decision making. Chapter

4 deals with the development and implementation of the prototype. In this chapter the prototype's architecture and special features will be discussed. Chapter 5 is focused on the design and execution of the user study concluding with a discussion about the results and findings of the study. Chapter 6 concludes this work and chapter 7 states the ways on which this work can be further developed.

2 Related Work

In this chapter projects related to product configurators and activity sculptures will be presented. Each work presents a unique solution to the addressed problem, the approach each author took will be discussed and the adaptation of useful knowledge to this work will be explored. To conclude the chapter an overview of available vendor API for data import will be presented.

2.1 Web-based Interactive Product Visualization

This work has a particular interest in product configurators that make use of 3D computer graphics to visualize the product. The majority of modern web configurators are image based and make use of well designed backgrounds to place the product in well perceived environment. For example the UNU electric scooter configurator puts the scooter on a street background that changes as the user moves to the next step of the configuration (fig.2.1). Other systems may opt for a more minimalistic look, and will try to isolate the product and place it in a white background as seen on figure 2.2. Although this might work for some products the user still misses some of the benefits of interacting with a spatial representation the products[63]. One of the main challenges of developing configurator systems is the modeling of the relation between the product configuration and its visual representation and the correct rendering of the visual representation in real time[13]. The advantage of a 3D visualization system over an image based one, is that the different configurations can be generated on the fly instead of using complex logic systems to retrieve the correct image combination from an image database. On the following section, three product configurators will be presented that use novel 3D visualization technologies to offer users a robust interactive tools for designing unique products.

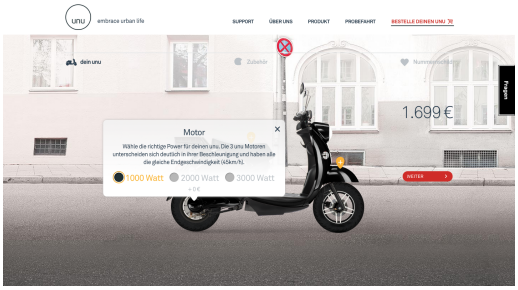


Figure 2.1: UNU electric scooter web configurator[17]

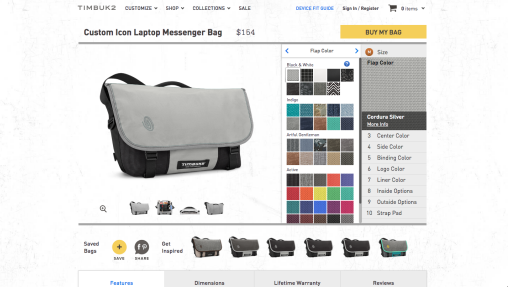


Figure 2.2: Timbuk2 bag web configurator[59]

2.1.1 Gates 3D Configurator

As part of an action-research project from Living Lab[45] Rolland et al.[52] developed a gate configurator for the French company Groupe Maine's gates. The objective of the authors was to showcase the possibilities of 3D Web technologies in e-commerce applications. The developed tool was build with the Unitiy3D[58] game engine, a flexible tool principally build for game development but it has proved to be also useful for architectural visualization and graphic intense web applications. After a user has logged in to the configurator, the platform allows customers to select from a variety of gate styles visualized as 3D models placed on the right side controls (see fig.2.3). The user has also the option of setting the environment in which the gate is being placed (left side controls). This can happen by either selecting a predefined environment of by uploading a picture of the user's home or place where the gate shall be installed later. Customers can position the gates in the uploaded photograph by operating dedicated slider controls. The main advantage of allowing customers to upload their own images is that this allows them to have a better idea

of how the selected gates will look in the final environment making them feel more comfortable about their decision. The authors state that they preferred the superimposition of a custom image rather than letting the user customize the environment with threes or buildings to make it look as close as possible because of the possible frame rate drop produced of handling many models and generating and because it is quite rare that somebody has a 3D model of his home.

One of the main aspects of the gates configurator in respect to the research purposes the authors had, was that of developing a tool that improved the visualization of products with the end of encouraging the purchasing of the product in an on-line medium. To validate the design ideas behind the gate configurator and analyze the impact it had on customers, the authors performed an empirical evaluation. The results of the 27 evaluated participants showed that the manipulation of objects in 3D space seem naturally and was also confirmed to be important to the participants to have this option. This shows that having a realistic view of the product improves the chances of sale.

Figure 2.3: Groupe Maine's gates configurator[52]

The gates configurator is a great example of how companies can make use of 3D visualization technologies to present their product to customers in a more convincing way. The authors of made a great choice of working with the Unity3D game engine as it allows them to port to other platforms as well, in this way developed software tools could be very easily ported to other mobile platforms like tablets or smartphones with bigger screens without the need of starting from scratch. This technology also offers desktop like frame rate performance in the web. The configurator has not many configuration possibilities which makes it easier for customers to understand but it excels at providing different visualization environments which in this case takes advantage of navigating in 3D space. One key statement of the authors was that tools like the configurator promote customer's participation in the developing of products. It was a shame there is no actual configurator in the Groupe Main's webpage for testing and it is unknown if the configurator was ever implemented in their website. The authors of the gates configurate see a great potential of 3D visualization technologies in fields such as e-learning, simulation and education.

2.1.2 The MakerVis Fabrication Tool

The MakerVis fabrication software is, in the words of its creators: *"a prototype tool that integrates the entire process of creating physical visualizations, from data filtering to physical fabrication"*[55]. If somebody attempts to visualize their information they will have to use a wide range of tools to achieve this. This is cumbersome and impractical as it requires a great amount of effort and in general it is a very conflict prone way of working with data. This motivated Swami-

nathan et. al to develop MakerVis as an attempt to offer a platform that unifies the needed tools to extract data, filter it and visualize it. Although the focus of the MakerVis is not the implementation of a complex tool but more the exploration of the different challenges encountered in the process of visualization.

The work-flow provided by MakerVis is composed of six steps all displayed in the user interface seen on figure 2.4. First users need to upload a CSV file containing the data to visualize. After that users can select between several visualization styles. Once a visualization is selected the data the user can begin mapping the data to the visualization by selecting data variables. Further on users can manipulate the visualization's geometry through an array of sliders and controls. In order to setup fabrication parameters and selecting the machine users can make use of the lower right section where the visualization is deconstructed layer by layer. Finally the STL file of the model can be exported for fabrication. A helpful functionality the authors implemented was the specification of printing materials through a JSON file. It is worth mentioning that MakerVis does not provide true 3D visualizations. The produced designs are indeed physical objects but the final result is a pseudo 3D visualization. In order to achieve this the authors use principally 2D fabrication methods employing laser cutting and CNC machines, only basic 3D printing support is provided. The visualizations users can choose from are more directed to traditional charts and bar graphs and not so much geared towards more artistic representations. Due to the modular engineering of tool, the visualizations can be expanded. The available tools for data manipulation is also limited and advanced manipulation should be better performed in an external tool. MakerVis was build with modern web technologies based on the JavaScript programming language like Node.js[26], Three.js[6], D3.js[4] and jQuery[35]. As some of this technologies were also used in the Activity Sculpture Configurator they will be explained in depth in Chapter 4.

Although the authors conducted a relative small user study, the results provided valuable information about what areas can be further improved. The user study showed that users wanted more detailed control about the decorative aspects of visualization. Providing tools for personalize text labels and scales increases would allow users to further customize their visualization. Another challenge encountered was that of material representation. It was difficult for users to decide on which materials to use and expressed the desire to physically interact with the available materials. Also the correct scale representation of the object was estimated wrongly by most users. The need of the ability to save and load visualizations was also expressed to be needed. This findings might sound as failures but they only make clear the limitations of the screen as a medium to transmit haptic properties of materials and that unifying every tool needed for data visualization in a tool is an immense challenge.

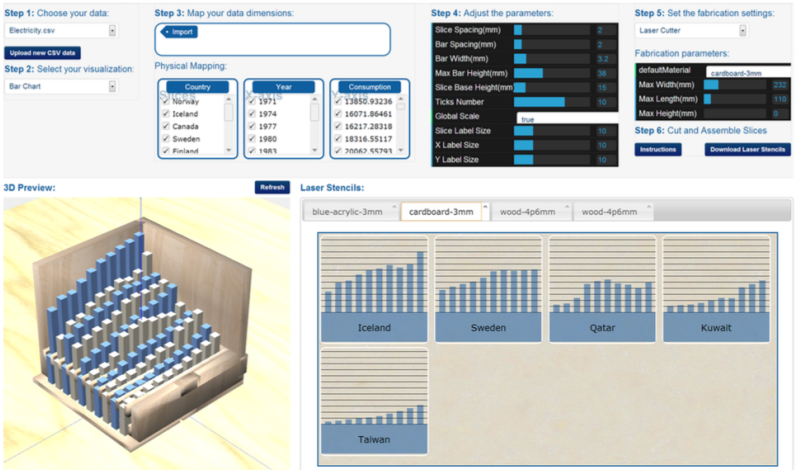


Figure 2.4: The MakerVis fabrication tool[55]

The MakerVis fabrication tool approximates closely the aim of the work discussed in this thesis. This work shares the same mission of building a tool were the whole process of visualizing a 3D object is contained. The MakerVis focuses more on data manipulation and fabrication as it basically dedicates two thirds of the whole screen to data filtering and fabrication parameters. None the less the interaction concepts and design layout served as a reference point for the activity sculpture configurator.

2.1.3 Twikit

The 3D printing market is a fast growing market estimated to reach US$16.2 billion dollars in value by 2018[51]. Many see the potential to offer services to facilitate the design and production of 3D printing goods. An interesting example is Twikit[60] a company located in Belgium that offers innovative software, fabrication and shipping solutions for brands and retailers looking to easily implement product customization to their product portfolio. The Twikit website[60] states that their goal is to "allow end-to-end 3D product customization, in the easiest way possible" and they see 3D product customization as "key in the future of products and experiences".

The range of services the belgian company offers covers every step in the development of a 3D configurable product. If needed Twikit is can aid brands in the development of their 3D product through collaboration with existing design teams. On the software side, Twikit offers a complex engine for 3D visualization, customization and fabrication called Twikbot. This engine allows companies to integrate customization tools into their current e-shop or application. Due to the modular nature of the engine, retailers can choose what kind of interface controls they need for the customization process and integrate them with the assurance that the design specifications and constraints will be met. The Twikbot also offers material configuration functionality for a variety of 3D printing techniques. Another integral component of the Twikbot is its Content-Management-System (CMS). The Twikbot CMS is a backend tool that manages and tracks currently configured and sold products and through an API cloud-based service it links to logistics and 3D print fulfillment providers.

The Twikit website showcases several study cases from small and large businesses that have integrated their software to sell appealing customizable products. As much as it would be interesting to know how Twikit performs in all areas of their services, for the purposes of this thesis, only the configurators showcased will be evaluated. One interesting case study in the website is the 3D Trophy Factory[60] which is owned and run by the same Twikit people. This e-shop sells a variety of trophy models were the text can be customized (see figure 2.5). It is a very simple configurator that allows customers to interact the trophy in 3D space through click and drag. Depending on the model customers can input up to three different text lines. The configurator offers the possibility to save and load configured trophies. If users have a custom 3D model they would like to print as a trophy, the e-shop supports uploading 3D model files. In some designs users can also change some text properties like randomize the positioning.

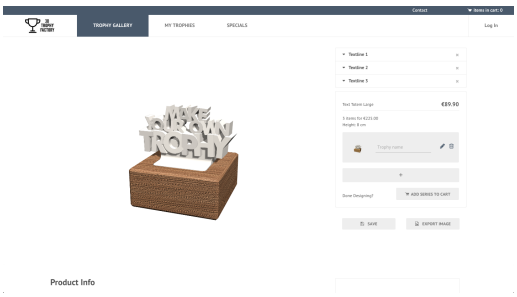


Figure 2.5: Twikit powered 3D Trophy Factory web configurator[5]

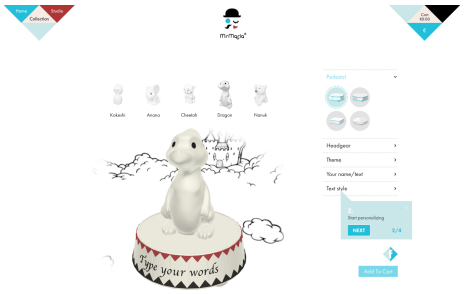


Figure 2.6: Twikit powered MrMaria Studio lamp web configurator[39]

The next Twitkit case study shows that the technology offered by Twickit can be easily used to enable brands no matter how big or small to start offering customizable products. Amsterdam based design studio Mr Maria[39] added a customizable toy lamp to their product line. The lamp configurator showcases the brand identity of the studio and allows customers to customize their lamp in a playful way. The lamp is composed by a model of an animal and a pedestal were a text is placed. Users can choose different animals to place over the pedestal and further customize it by adding different accessories like hats, different pedestal designs with different color themes and a text field for adding a custom text (see figure 2.6. The lamp is placed in a cartoon style background which can be changed to a dark view where the lamp glows simulating to be turned on.

The 3D Trophy Factory and the Mr Maria lamp configurator were both very appealing and fun to work with. Twikit does not say much about the technologies used in the framework but a view into both e-shop's source code showed that it makes use of the Three.js[6] graphics library. In summary, the toolkit offered by Twikit seems to make quality visualizations with good performance as the interaction with the products is always fluid. It would be interesting to see how much does Twickit charge customers for their service but the website only offers a contact form to get information.

2.2 Activity Sculptures

The advancements in digital fabrication technologies have allowed the development of new visualization metaphors transitioning from the screen to physical space. Activity sculptures or data sculptures are a fairly new concept in the data visualization field and its usefulness is still largely misunderstood. Much work has been done trying to give activity sculptures its place comparing them to other physical visualizations like ambient and casual information visualizations[32]. A understandable to differentiated activity sculptures from other visualizations by highlighting their perceived physical qualities[63, 70]. Data sculptures are per se, the embodiment of the data in a tangible and visible form[70]. This gives activity sculpture a more broader representation space making it difficult to find a metaphor were the influence of the data on the object's form can still be related producing, as Vande Moere called it, a metaphoric distance[63]. Activity sculptures can communicate also through the affordance of an object which is the perceived potential to perform an action through the object. This property calls users to interact with the sculpture in an explorative manner[32, 62], which could be take advantage of haptic qualities of objects and materials to communicate information through more sensory channels[3].

In the following sections current work on the use of activity sculptures to visualize physically different kinds of activity data will be presented.

2.2.1 Sweat Atoms

SweatAtoms is a visualization system developed in a research project by Koht et al.[36]. The authors aimed to better understand how the representation of physical activity data though material artifacts can be used to reflect on physical activity. In this work the authors concentrated in visualizing heart-rate data through 5 different sculptures each one emphasizing a different aspect of the data (see figure 2.7. For visualizing heartbeats per minute an extruded graph was used. Higher fluctuation in the heart-beat values were visualized as a flower mapping the heart-rate value and duration to the length and width of the petal. The amount of the performed activity in a day was mapped to the size of a frog. The time spent in each of the six different heart-rate zones (resting, recovery, aerobic, anaerobic, speed and alarming) was mapped to each side of a dice. And lastly a ring depicts through circles the number of active hours in a day, the diameter of these added circles is affected by the increment of the heart-rate.

The authors conducted a two week field study with 7 participants with different demographics and exercise habits. Each participant received a 3D printer, a heart-rate sensor and an iPod

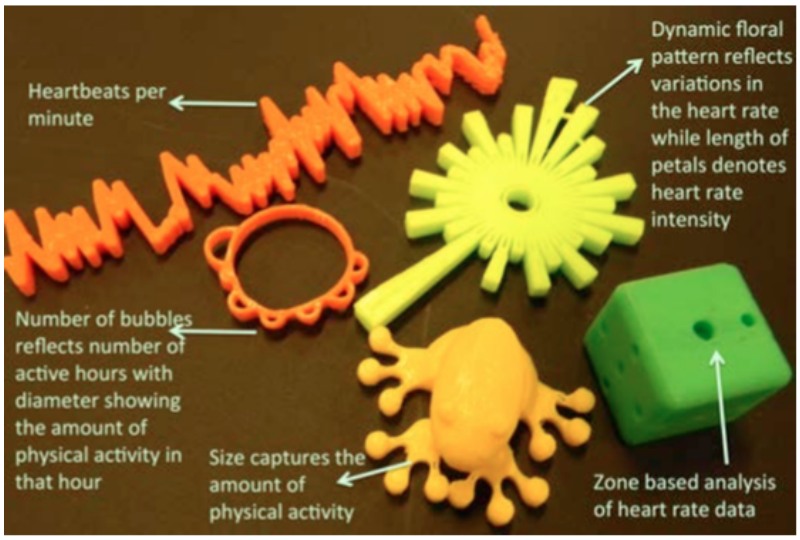


Figure 2.7: SweatAtmos activity sculptures[36]

touch. Participants were required to wear the hear-rate sensor every day. The sensor was paired via bluetooth to the iPod and automatically loaded the data to the SweatAtoms system when the monitoring stopped. SweatAtoms then generated the 5 sculptures from the data. The models were then send to the authors for STL export. Once converted the participants received the STL via email. Participants then proceeded to print the 5 sculptures with the provided 3D printers at home, a procedure that took around two hours. Users kept a diary reflecting on the sculptures and their activity every day.

The study showed the participants engaged differently with each sculpture. The favorite sculpture was the frog sculpture. The graph was described by a participant as “not very exciting as we can see the same on a virtual screen”. On the other hand a very active participant liked to see its graph sculpture as he could see how active he had been. For the flower participants expressed different ideas as they could interact with it. A female participant saw the flower suitable for wearing it as earrings. The utility of the rings was often questioned, and was regarded as the least appealing sculpture of the 5. Some users started grouping and assembling the sculptures to form a bigger sculpture. Some users expressed that the dice made them reflect on their lifestyle by analyzing the time they spent on each heart-rate area. The sculptures were also subject of conversations with work colleagues and friends. Users also stated that the value of their heart-rate had gained more significance as the sculptures provided allowed them to “touch and feel” their data, this made the sculptures more valuable to participants than screen based visualization. The uniqueness of the objects was repeatedly mentioned by users to be very appealing and the fact of being able to constantly reflect on their activity motivated them to exercise with more intensity.

The SweatAtoms study is a great example of how through activity sculptures motivated users to reflect more about their activity and made them take action. It is surprising to see how participants did not have any trouble at all with the fabrication of the sculptures at their home. The different technologies integrated seamlessly providing a solid set of tools to visualize and easily construct the sculptures. The metaphors utilized to map the data were varied and showed that visualizations that are useful in virtual space not are not necessarily appealing as objects. On the other hand sculptures that were not extremely abstract but rather communicated the data through a playful metaphor, like the frog, were the ones that users liked the most. Overall the SweatAtoms was a very successful project that produced interesting findings regarding activity sculptures and their influence on users.

2.2.2 Mental Fabrications

Mental Fabrications is an art project from artist Ion Popian[50] that aims to visualize brain activity produced in response to different types of stimuli in the form of a 3D printed landscape were the intensity of the activity creates valleys and hills (see figure 2.8). Popian states normally architects design spaces and buildings and study how these spaces are perceived by people. This motivated him to develop a concept were the design of environments is originated by the perception of people about a space. Through a NeuroSky MindWave Headset[43] electroencephalogram (EEG) Popian is able to meassure the brain activity while users are wandering through different environments for a prolonged time frame. In this way it is possible to study how users perceive their environment through their . By analyzing the way the brain reacts he attempts to design a space or environment that provokes a certain stimulation to users.

The Mental Fabrications project was produced in collaboration with programmers that developed a software in Max/Msp[1] that translated the EEG data from the headset to produce hills and valleys in a plane and exported for 3D printing. For a series of exhibitions of the work, Popian developed a film that was presented to visitors wearing the headset. The brain activity was then measured and visualized in the landscape. Visitors could then self reflect on how calm they were at certain scenes of the film or what sequences provoked hills in the landscape as a consequence of high brain activity. With the collaboration of enthusiastic participants the exhibit produced several landscape visualizations enabled by the usage of sound, textures and imagery in the film, showing visitors the process of generating visualizations from data. To expand the sculpture a second set of data was being generated in the exhibition, namely a camera was filming the movement and proximity of the visitors and was later visualized as a second landscape in a wire-frame style which was then interposed to the landscape generated by the brain activity of the viewers.

Even though this project is more art oriented it proposes interesting ideas of designing environments. Technology is at a point where even EEG devices are easily obtainable allowing designers, or architects in this case, to use data to create new design concepts. Among the flood of quantified self applications Mental Fabrications offers a fresh approach to activity data exploring brain activity which can be viewed as strongly linked to emotions and thoughts. The produced sculptures are really elaborated and as seen in other studies provoked participants to analyze how their brain reacted to the imagery. This work It will be interesting to see how technology enables the visualization of different forms of activity through new sensors.

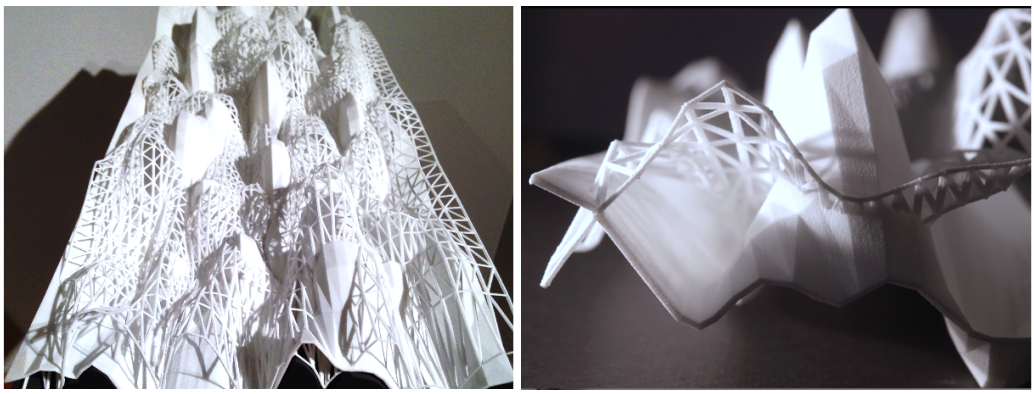


Figure 2.8: Mental Fabrication's brain activity landscapes[50]

2.3 Fitness Trackers

An integral part of the activity sculpture configurator developed in this work was the gathering of the activity data as it is basically the first step in the visualization pipeline. As shown in many

examples[36, 55, 50] through fitness trackers we have the possibility to gather information in an non obtrusive way, facilitating the task immensely. Added to these tracking devices many of the manufacturers also provide a set services for developers to encourage the development of third party applications that can take advantage of their fitness tracking product. In order to make the gathering and exporting of data as simple as possible for users, the author decided to use a wearable fitness tracker in the form of a wristband or clip and make the data available to the web configurator through a web Application Programing Interface (API). By doing this, the user does not need to export the data through USB sticks or handle CSV files. Working with web APIs ensures that the configurator will always read the data in the same format because of the standardization and organization an API provides. Further details of web API implementation and functionalities will be addressed in chapter 4. Even though there are many applications for smartphone the user experience design goals explained before narrowed the options down to dedicated fitness trackers. In the next section an overview of considered fitness trackers and their accompanying API will be presented.

Nike+ FuelBand SE

The Nike+ FuelBand SE[28] is a wristband equipped with sensors providing information for daily step count, burned calories and Nike Fuel, a unit introduced by Nike to measure daily activity. The wristband does not provide that much information by its own, but when used with the accompanying iPhone and Android applications which makes use of the device GPS sensor to provide positioning data and routes from running sessions. This product was first considered because of the author's previous experience with the Nike+ Running App[29]. Nike provides a web API and several Software Developer Kits (SDKs) for different platforms in their developer portal[27]. The web API provides only basic documentation and most of the features of the wristband and the app like GPS data and activity data are supported, although in comparison with the other APIs reviewed it is very limited. The major drawback of the Nike+ environment is that its only available to selected partners. A contact form is available but it is unknown what selection process has to be undergone and most importantly how long does it take to receive an answer from Nike. This was the main reason of why the Nike+ FuelBand SE was discarded to be used for this work.

Jawbone UP 3

The Jawbone UP 3[33] is a really well designed, both aesthetically and functionality wise, fitness tracker. It tracks steps, exercise, calories burned, sleep and heart-rate. Jawbone also provides a smartphone application where users can also track meals and set activity gaols. The Jawbone API[34] offers a great documentation with example code and detailed explanations. It provides developers access to users information about any measure the tracker and the app can perform. It even offers information about marketing strategies for developers seeking to develop a third party product for Jawbone products. The API is available to the public in general. The only point of consideration about using the UP 3 for this work was the high price tag of around €180.

Misfit Shine

The flagship product of Misfit is the Shine tracker[65]. The tracking possibilities is limited to steps, calories and distance but the Shine compensates this with extreme robust materials, up to 50m waterproof and it does not need to be charged. The Misfit API[64] had not been released at the time of researching available options which was the main reason for discarding the tracker. As of today, the API provides a short and concise documentation and is openly available.

Fitbit Flex

The Flex fitness tracker from Fitbit[23] measures steps, distance, calories burned, active minutes and sleep monitoring. All in all the flex is a very complete tracker with an affordable price tag. At the time of performing the research Fitbit had announced two new trackers with heart-rate and GPS support which are now both available. The API offered to developers[22] would serve the purposes of the activity sculpture configurator well as it provided access to all tracked data and it was well documented. One aspect that caused interactivity about its usefulness was that at the time of researching available options it did not support heart-rate data. The API is also available to everyone with a Fitbit account.

Withings Pulse Ox

Thw Withings Pulse Ox[67] is a compact tracker that enables users to keep track of steps, distance, elevation, calories burned, activity time, sleep, heart-rate and saturation of peripheral oxigen (SPO2). Out of all available trackers in the price range of €100 to €150 the Pulse Ox tracks the most types of activity data. This was a very attractive feature as the budget for this work was limited and having many variables for visualization was highly sought after. Withings provides also an app for smartphones that offers an overview of all the activity tracked by the device. The Withings API[66] has decent documentation, but just enough to get started but all the tracked information is available. Because of the affordable price of the device and the openly available API the Pulse Ox was chosen to gather the activity data for this work.

Figure 2.9: Fitness Tracker Wristbands. From left to right: Nike+ Fuel Band SE[28], Misfit Shine[65], Jawbone UP 3[33], Fitbit Flex[23] and the Withings Pulse Ox[67]

Summary

In general all available fitness trackers in the market offer good functionality for monitoring daily activity and would serve well any quantified self enthusiast. For more advanced users that are looking to develop their own tools the provided APIs depend in large to the functionality offered by the sensor and the mobile app. For the purposes of this work the decision was narrowed down to the Wihtings Pulse Ox, the Fitbit Flex and the Jawbone UP 3 as these brands showed more presence in tech and quantified self blogs. The Withings Pulse Ox was chosen because it offers a wide range of activity data, its affordability and well supported API. The authors experience actually working with the fitness tracker and the API will be further discussed in chapter 4.

3 Prototype Design

This chapter presents the process of developing the designs for the activity sculpture and the web configurator. In the first section the requirements for the activity sculpture are presented and four different designs that attempt to fulfill the requirements are presented. Finally the validation process for the sculpture to be designed will be discussed. The second section is devoted to the design process of the customization system. This comprehends not only the user interface but also the user experience while operating the system. After a process where three different concepts were ideated, through the validation of the design the final prototype is chosen.

3.1 Activity Sculpture Design

The activity sculpture and the configurator are strongly interconnected as depending of the sculpture design, the sculpture will influence the quantity of controls to be taken into consideration for the design of the configurator giving users greater or lesser freedom for manipulating the the sculpture. As we discussed in section 2.2, activity sculptures portray different attributes inherit from their physical nature. This why a different approach for designing physical data visualizations is needed. For this purpose the design taxonomy proposed by Vande Moere et al.[63] was used as a guide to better categorize the qualities of the developed designs. This taxonomy is has two dimensions which describe the design space of activity sculptures: *representational fidelity* and *narrative formulation*.

The representational fidelity attempts to explain the decision made for the embodiment of the data in form. This includes the chosen metaphor for mapping the data to the object and the resulting metaphorical distance, this is the level of abstraction used to represent the data through the metaphor. In order to better explain the abstraction level Vande Moere's taxonomy uses concepts from semiotic studies. According to C.K. Ogden et al.[46] semiotic signs can be explained through their three major concepts: the signified, the signifier and the sense. The signified is an object that represents a physical thing or and idea. The signified is represented by the signifier who tries to bring the same experience an observer would have with the signified. The sense is the experience brought by the signified. Furthermore signs can be categorized into iconic, indexical and symbolic. Iconic representation occurs when the signifier resembles the signified, like a picture or a diagram. Activity sculptures are iconic when they resemble the metaphor they are interpreting. Examples of this could be the heart-beat extruded graph sculpture in discussed in section 2.2.1 or the MakerVis visualizations in 2.4. An indexical sign has a sensory feature that correlates directly to the signified. The signifier points to the signified through an actual connection, like dark clouds point to a forthcoming rain or smoke points to fire. Activity sculptures can be classified as indexical when through the use of a property directly related to the data. An example of such a sculpture would be the SweatAtoms frog in section 2.2.1 or the . The most complex kind of sing is the symbol, as it does not bear any resemblance to the signified. The relationship between the signified and the signifier has to be taught by convention in order to be understood. Activity sculptures making use of symbolic representation are the hardest to understand as the relationship to the data has to be learned as it is not apparently displayed. Example of symbolic sculptures would be the landscapes in the Mental Fabrications project discussed in section 2.2.2. Through the definition of iconic, indexical and symbolic representation we can derive their metaphorical distance resulting in indexical having the closest distance, symbolic the furthest and iconic a medium distance[55].

The narrative formulation of activity sculptures is a product of the physical form and the affordance of the object which influences the user's ability to discover information through interaction and perception. This quality is strongly interconnected to the representational fidelity as the level of abstraction in which the data is presented will form the properties in which the sculpture communicates. As discussed in section 2.2 the affordance of an object describes to the viewer the object's potential to perform an action. The level of abstraction of the sculpture will influence how

inviting the object is to the viewer depending on the user's level of familiarity with the metaphor used.

Computer aided design systems offer almost endless possibilities in the design of activity sculptures, allowing designers to create complex structure designs that with manual methods it would be almost impossible to conceive. Even though this might be the case in the digital realm, translating the virtual object into a physical object may be still a challenge. An important aspect to be considered while developing activity sculptures is the manufacturability of the sculpture[55]. 3D printing machines still are challenged by certain types of structures depending on the technology that is being used to manufacture (granular vs extrusion methods). Therefore it is important to take into account how challenging the manufacturability of the sculpture will be.

The the aforementioned design considerations of activity sculptures the requirements for the activity sculpture were formulated as follows.

- The sculpture has to be aesthetically appealing to users
- Motivate users to self reflection
- Because of the wide range of activity data types obtained through the fitness tracker, the sculpture has to be able to visualize as many variables as possible
- In order to provide users a relatively high level of freedom while customizing the sculpture, the sculpture has to offer multiple configuration options
- The sculpture has to be extended to new interaction forms
- The sculpture has to be 3D printable by current 3D printing technologies

With the defined requirements the author formulated four designs exploring different possible approaches. Due to space and layout considerations, the sketches drawn for the prototypes will not be displayed in this chapter only fragments. Due to the large sizes of the sketches all prototype sketches were placed in their entirety in the appendix of this work. For the sculpture prototypes sketches please refer to section A.1, for configurator prototypes sketches refer to section A.2.

3.1.1 3D Graph

The first prototype is based on a line chart but augmented to describe more data variables. The idea was inspired by multiple exposure images of Olympic athletes in the middle of their performance. The multi exposure technique allows photographers to take a snapshot of the athlete at a specific point in time. The resulting image shows a group of athletes in different positions completing a cycle of a movement (see figure 3.1). The concept of presenting snapshots of the athlete's position over time is actually a parallel concept to standard charts and charts where the value of a variable is presented at a specific point in time. Only transferring the line-chart to a physical space would have been not appealing enough and very limited as it would have been only possible to visualize one variable over time. The first modification made in order to improve the aesthetic of the sculpture was to give the chart-line volume and a triangulated or low polygon count (lowpoly) aesthetic, two properties that can be explored thanks to physical space. With this modification the sculpture gained the ability to visualize one more variable. The first variable influences the hight changes in the Y axis of the chart-line over time and by adding the volume to the line we can change the radius of the line according to the value changes of the variable over time.

Up until this point the 3D graph would look as shown in figure 3.2. It visualizes up 2 variables over time. Even though the data is being visualized as an abstract triangulated body with hight and radius changes the sculpture could take advantage of the depth dimension not only to display volume but to steer the body also in this dimension. Through this realization, the 3D graph now

Figure 3.1: Athlete's movements captured in a multiple exposure image[10]

can map 3 variables. To further enhance the aesthetic of the sculpture the concept of the athlete's snapshot in specific points of time was explored. For this a set of avatars performing different sports were developed. These avatars would intersect the graph body in specified intervals by the user. By adding different avatars performing a variety of sports users can chose the avatar that represents best the sport or activity the user did to generate the data. The sculpture could be expanded to support different decoration styles. For example apart from the triangulated look of the graph body, other visualization like a wire-frame option or a smoothed look could be offered. As shown in figure 3.2, the whole structure looks as if it would float in the air with the help of a support structure, which could be manufactured separately from transparent acrylic so that they fade away centering the attention in the sculpture. This support structure would be inserted into a wooden or acrylic plate. The author explored the idea of engraving a summary of the data the sculpture represents. In this way users have both the abstract visualization and the actual data in the same place close enough to analyze. This might be useful while showing it to others, as it can help them better grasp how the sculpture was generated. By adding a plate to the sculpture we open a new set of configuration possibilities for users to edit in the configurator.

The 3D graph embodies activity data through iconic and indexical representations. Indexical because as stated before, the concept emerged from an expanded line-chart an the graph body points to the path line-chart would have but abstracted and stylized and with the avatars in different positions point to movement in activity. The avatars are a strong icon of a human body in motion. The 3D graph shows a somewhat short metaphorical distance as the sculpture points strongly to the a line-chart metaphor used to embody the data. In the narrative formulation of the sculpture again the chart-graph resemblance and the plate tell the user this object is to be analyzed or at least admired as the plate does not allow grabbing the sculpture. Self reflection is highly encouraged as the metaphor motivates comparison of values over time.

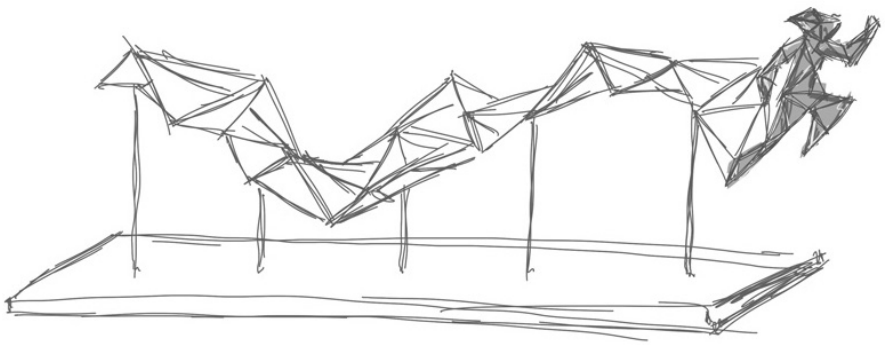


Figure 3.2: 3D Graph sketch

In summary, the 3D Graph prototype is a highly aesthetic activity sculpture that is also functional as the both the sculpture and the data could be analyzed in the same object. The sculpture can visualize data generated on a large time span. it has a rather low capability of mapping several variables because of the constraints of the line-chart metaphor. This could be improved by adding more graph bodies representing each 3 variables over time. The 3D graph offers a wide range of configuration options that would make it a rather interesting object to customize in the configurator. The manufacturability of the sculpture is attainable but also complicated as the sculpture, the plate and the supporting elements have to be fabricated not to mention the engravings on the plate.

3.1.2 Activity Landscape

The activity landscape is an activity sculpture that embodies the data generated in a single day's activity by utilizing the data to generate a terrain (see figure 3.4. This concept of this sculpture came from the idea of generating a ground path from the GPS location points recollected while a user was jogging or riding his bike. After the workout other activity information like heart rate or elevation change would be mapped to the GPS locations. The user would then select in the configurator which variables he wants to map to different aspects of the terrain like and experiment with different variables for terrain elevation or map other variables to the radius of hills for instance. The activity landscape would also feature a plate for showcasing the sculpture. This plate as in the 3D Graph would feature engraved information about the work out like variables visualized, duration of the workout and calories burned. Because the terrain is based mostly on GPS information, the plate could also have an picture or engraving of a map where the workout was performed. As seen in section 2.2.1 were some users stacked sculptures to form a bigger sculpture, several activity landscapes could be printed to form a bigger landscape. This could be used as a motivation to users to explore new parts of their city, it would be interesting to see how the landscape would look like after a year running or biking. This concept could also be applied to users who maybe travel constantly or like to ride their bikes in national parks around the world. After each travel they could bring the memories of their adventure in an activity landscape and put them together at their homes.

Figure 3.3: Procedurally generated landscape [47]

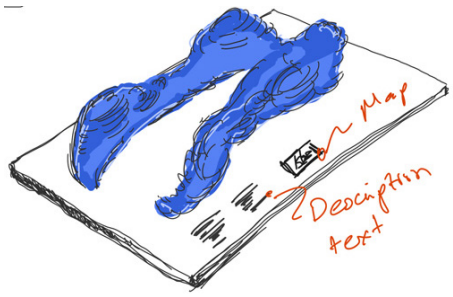


Figure 3.4: Activity Landscape sketch

The inspiration of this idea was taken from the game development were with advances in processing power terrains for games are not being loaded from the assets library as 3D models bur rather generated in real time even with complex erosion simulations[47] (see figure 3.3). After looking at several generated terrains the concept of utilizing activity data to manipulate the generation of a terrain emerged. The configuration process would have been interesting to see as every time the user selected a different variable to manipulate the terrain, new sceneries would have emerged.

The activity landscape embodies the activity data through iconic representation as some of the terrain resembles some characteristics of the data like elevation and geographical position for

example thus exhibiting a medium metaphorical distance. The level of affordance is low as it only invites the user to contemplate and self reflect.

To summarize, the activity terrain provides an interesting approach to activity visualization and maybe also to procedural terrain generation in games. The level of customization in the configurator is high. It has to be clarified that the complexity of designing terrains was not studied in depth by the author, therefore it remains unknown how many variables are possible to visualize through the terrain. The added plate with the map can give users a stronger sense of belonging to their city. The idea of a personalized souvenir from adventure rides would be also interesting to explore. The sculpture does not exhibit any properties that could be proved difficult to manufacture. Same as the with the 3D Graph the plate could only make the fabrication process somewhat longer.

3.1.3 Activity Flora

The activity flora is an activity sculpture that utilizes activity data as input parameters for the generation of a sculpture resembling a leaf. The concept for this prototype is to take the data generated in a single day or during a single workout and materialize it in a leaf like structure. In the sketch showed in figure 3.5 activity data variables can be mapped to properties of the leaf. For example the size of the leaf could be affected by the traversed distance during that day, the contour shape could be mapped to elevation changes. The main branch in the middle of the leaf could be manipulated by the velocity changes, furthermore he characteristic branch structure inside the leaf could be influenced by the average heart-rate value (see figure 3.5). For the inner structure of the leave a generative system could be implemented to simulate better the leaf structure. In the configurator users could select which variable to map to each leaf generation parameter to generate a series of leafs. The customization possibilities could be further expanded with the addition of leaf tails and material options. To further explore the concept of grouping sculptures, a sketch with a stand that holds several leafs was drawn like a very stylized flower pot. The leaf stand or pot would also be able to be customizable in the configurator.

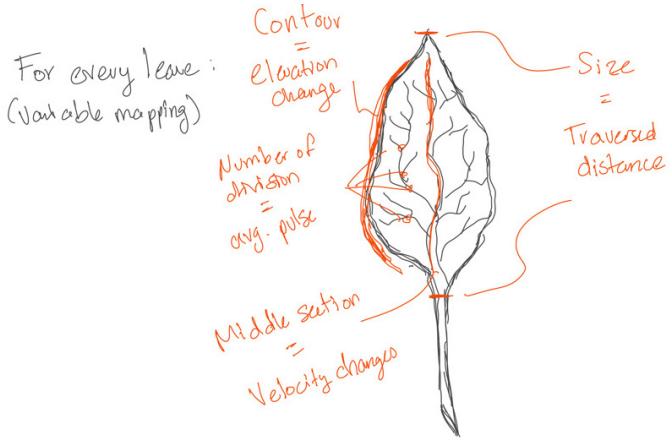


Figure 3.5: Activity Flora sketch

The idea was inspired by the works of New York based generative design studio Nervous System[31] who since 2007 have been pioneering the design of generative objects for 3D printing. The majority of their designs is based in the simulation of natural structures or patterns through algorithms (see figure 3.6. The concept of generating an object from activity data with an aesthetic inspired by nature was very appealing to the author. It would be interesting to see how users engage and perceive an object that was generated by their own activity but that also looks like something

that lived, like a petrified leaf. Maybe in the future when 3D printing is also possible to do with biologic materials some kind of bacteria culture could be utilized to print the leaf alive for real.

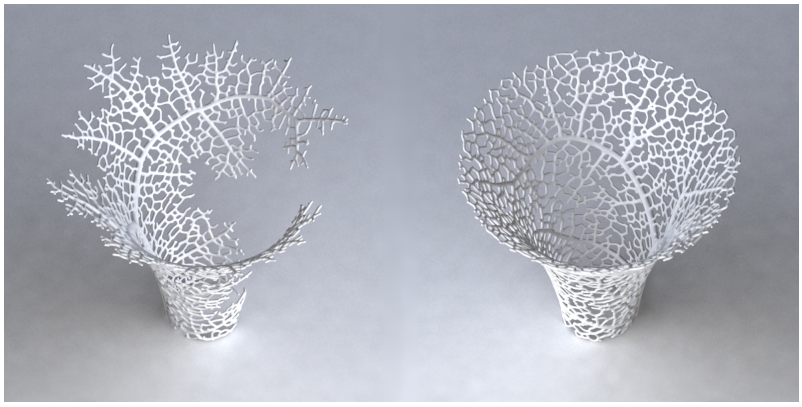


Figure 3.6: Sculptures made with generative algorithms based on hyphae growth as seen on leave and coral structures[30]

The activity flora embodies the data mostly iconic representation as the sculpture resembles a leaf but some information is abstracted completely because of the generative system resulting in a high medium metaphorical distance. The leaf prototype invites the user to contemplate the sculpture and due to the metaphor of the leaf self reflection will be harder for the viewer as the data is mapped not entirely to the sculpture but also to the system generating the leaf's structure.

To conclude this prototype, the activity leaf showcases a mixed approach to activity sculpture design, mapping some variables to physical properties of the leaf and using activity data as input for a generative system. The configuration possibilities are rather scarce and focus more on secondary elements of the sculpture like the stand or the leaf tail. The concept of metaphorically giving an object natural characteristics would be interesting to explore in the context of activity sculptures.

3.1.4 Activity Vase

The last developed prototype is the activity vase, a sculpture were the structure is conformed of a stacked radial chart with smoothly connected the axes. Every stacked radial chart represents a day's worth of activity. The user can choose in the configurator how many variables he wants to add to the vase by populating the radial chart. The height of the vase would be then configured by choosing the time span to be visualized which translates in the number of stacked radial charts (see figure 3.8. In this way the we can offer the user different outputs from one sculpture. Users could gain different usages of the sculpture depending on the visualized time span. For short time spans the sculpture could be expanded or adapted to be used as earrings or as a key-chain, longer time spans could be then used to make a pen holder or even a mug. Users could print each day the visualized data and maybe stacked them in a pole. As the sculpture better visualizes larger time spans from at least a day to weeks or months a sculpture representing two months in a year could be visualized, for example at the beginning and at the end of an intense workout plan an inner solid sculpture would represent the first month and an outer sculpture enclosing the first would represent the last month as it is expected that the activity performed will contain higher values in some variables, as the user reaches longer distances in their trainings for example. The sculpture could be further customized in the configurator through 3D printing material options and maybe even a simple plate were the labels of the variables are engraved.

The concept of the activity vase was inspired by a 3D printed sculpture that maps two or more facial profiles to the sides of a vase and then unites them through extrusion. The final product

is called Fhaz (see figure 3.7), which is basically an expansion of Rubin's vase[53] to visualize more than to profiles in a 3D printed object. In order to transfer the concept of Rubin's vase to an activity sculpture the idea of utilizing a stacked radial chart proved to be a suitable solution.

Using Vande Moere's taxonomy, the activity vase embodies the activity data through a symbolic representation as the visualization has no resemblance to the metaphor. As the abstraction of the represented data in the sculpture is high the metaphorical distance is also high. The activity vase offers viewers the freedom to explore the sculpture through touch and self-reflection because of the viewer's low familiarity with the object. The narrative has to be figured out through exploration and interaction with the object.

The activity vase is a very flexible sculpture in terms of data mapping. Through the radial chart metaphor the visualized amount of variables is only limited by the users decisions in the configurator. Due to its abstract representation the sculpture could be adapted to be used as everyday accessories and not only limited to standing in a shelf. The configurability of the sculpture is not very high but this could be addressed through the addition of different rendering styles like wire-frame to the configurator. The manufacturability of the sculpture is attainable as the solid structure of the vase does not challenge current fabrication methods.

Figure 3.7: Fhaz: Procedurally generated vase based on facial profiles[44]

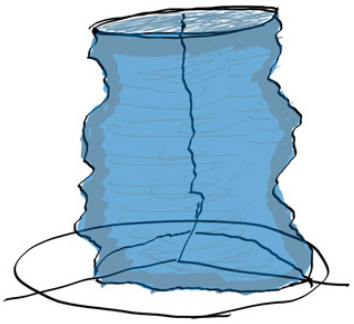


Figure 3.8: Activity Vase sketch

3.1.5 Prototype Validation

After an extensive development process it was time to decide what prototype would be chosen to implement in for the web configurator. The decision was made by the author and the tutor through an iterative evaluation process. During the design process the prototypes were analyzed and discussed in order to discuss possible challenges or interesting concepts. Table 3.1 offers an overview of how each prototype compares to each other. For the analysis Vande Moere's taxonomy, the estimated complexity of the implementation of the sculpture and the customization capacity of the sculpture were considered. Estimating the implementation of the sculpture was vital, as the time of developing for the configurator and the sculpture were limited. The customization possibilities offered by the sculpture were also important as it was important to offer users an appealing sculpture that can be customized to their liking and more configuration options mean more unique sculptures.

With this considerations in mind the activity vase fitted best the criteria. It is a very dynamic sculpture that offers a high variable mapping capability due to the radial chart metaphor, making it possible to visualize a high number of variables in the same sculpture. Unlike all other sculptures that are limited to 4 variables. The implementation complexity was estimated as medium because it can be achieved through common geometry construction algorithms. In comparison both the activity terrain and the activity leaf would have demanded a great amount of research on terrain

generation and natural structure simulation. An important factor for choosing the activity vase was that it does not rely on a specific property of the data to be generated. For example the activity landscape depends heavily on GPS data to be generated. After analyzing the fitness-tracker that were intended to be used in this work, none of them supported GPS data. Due to the high cost of fitness-tracker that supported GPS and to the complexity of using a smartphone to address this issue, the activity terrain was discarded. The abstract nature of the activity vase, even though not familiar to most users it provides a representation that can be expanded to use in wearables or other artifacts that appeal to both genders, reaching in this way a wider audience. And finally it offers also a reasonable customization capacity that can be later expanded if needed.

The final implementation of the sculpture and the possible control features will be discussed in Chapter 4 in section 4.4.2. Following the design process of the configurator will be discussed.

Sculpture Name	Metaphor	Semiotic Representation	Variable Mapping Capacity	Estimated Implementation Complexity	Customization Capacity
3D Graph	Line Chary	Indexical /Avatar Iconic	4	Medium high	Medium high
Activity Landscape	Terrain	Iconic	min. 2	High	High
Activity Flora	Leaf	Iconic	4	High	Low
Activity Vase	Vase/ Cylinder	Symbolic	Unlimited	Medium	Medium

Table 3.1: Activity Sculpture Prototype Comparison

3.2 Configurator Design

The configurator's main goal is to guide a user through decision making, while keeping consistency with design constraints[2]. The design of a user friendly web configurator is not an easy task as it involves thoughtful design of the configuration process, the controls with which the user will manipulate the object and the constraints that will ensure that the chosen configuration fulfills the design restrictions. Abbasi et al. made an in depth study of 111 web configurators[2] providing helpful insight of most used techniques. In their research Abbasi et al. divided the functionality of a configurator in three areas: *configuration options, constraints* and *the configuration process*.

Configuration options comprehend the user interface widgets that are used to change values, choose options and manipulate the object in general. The most common widgets are combo box items (drop-down lists), images, radio buttons, check boxes and text boxes. It was found that 83% of the analyzed configurators preloaded their controls with default values and if the user did not input a value in a control it continued with a default value. If users miss filling in values in specific controls or input fields the configurator can notify the user if he has left any mandatory field empty or when continuing to the next section the user will be notified that he left an empty field.

Constraints can be used to proof the correctness of user input values or also to hide functionality from novice users. In order to achieve ensure the correctness of the input data several checks can be performed. Type correctness ensures the value's type matches the expected type: integer, strings etc. Rage correctness tests if the value resides in the lower and upper bounds. Value formatting specifies an expected format and case sensitiveness for dates, colors or email addresses. 62% of the studied configurators performed these checks in a prevention pattern where the configurator notifies the user of a mistake before he submits the input value. It is common to hide constrain the visibility of certain controls if not activated.

Configuration processes can take mainly three forms. A single step configurator displays all controls in a single view. Basic multi-step configurators display the options in graphical containers that can be displayed successively or in a single view. Hierarchical multi-step configurators divide the controls are similar to basic multi-step configurators but allow the subgrouping of controls in a graphical container. Multi-step configurators can activate their controls in a step-by-step manner or they can make all controls available at any moment. For the design of multi-step configurators is important to keep in mind that some kind of decision propagation system has to be implemented in order to keep the decisions made in a previous step available and display their impact to the product.

Another important guideline for the design of the configurator was the ISO 9241-110:2006 standard for the ergonomics of human-system interaction[11]. The standard offers a series of recommendations for user interface designers and developers of interactive systems. The Volkswagen web configurator made use if the guidelines in the standard and they proved to be useful in the design process[37]. The following recommendations were taken into consideration for the design and prototype phase of this work.

Suitability for the task expresses the interaction should be suitable for the user's task and skill level. Meaning the user is given meaningful aids to accomplish his task and the interface supports functionality for different kinds of users e.g. "advanced" or "novice".

Self-descriptiveness attempts to make clear to the user what tasks should be performed next. Descriptions have to be reached easily by the user and several different views of the product.

Controllability. The user should be able to control the speed and order of the interaction as well as other navigation aids.

Conformity with user expectations. This consistency of the interface is important to meet the user's expectations. In order to achieve this the design process has to be logically understandable to the user. Fore example the user should know at all times in which step of the configuration process he is.

Error tolerance. The interface reacts reasonably to all user input and mechanisms that prevent the user to make mistakes should be implemented. If an error occurs the interface has to be forgiving and the error should be easily reversible.

With the obtained guidelines from both the extensive study of Abbasi et al.[2] and the ISO 9241-110:2006[11] the configurator prototypes took the following form.

3.2.1 Ideation Process

The configurator prototypes were ideated in an iterative process where different concepts were developed and discussed between the author and his tutor.

The first prototype was a multi-step based configurator where every steps presents a new set of visual controls. This first prototype welcomed the user with a home screen for the configurator. In this screen the user can login with his tracker account or he can scroll down and see a gallery of sculptures other users have made. If the user keep scrolling down a tutorial section is presented. This section is intended to be only a brief introduction of the workflow of the configurator. After logging in the user lands in the data dashboard. This is the first step of the configurator and serves to get the user acquainted with his data. In this step a line-chart with three tabs at the top displaying different data categories is shown from which the user can chose variables to visualize. The user knows in which step he is at all times with a navigation bar at the top of the screen. The current step number is highlighted. The user can further explore the data with zoom controls and also with an on hover radial chart displaying the value of the variables visualized for the given point in time. The next step is the sculpture selection menu. This step allows the user to select a specific object he can make out of the activity with a short description. Once the user has chosen a sculpture style the sculpture configurator is shown. In this step the user can customize the sculpture by selecting from a variety of materials and colors. The user can also chose between different connection modes,

which alters the way the vertices of the sculpture are connected. At the bottom of this view the selected variables are shown in both radial and line chart views. Thinking of the configurator as a commercial product it would be helpful to have an overview of the price and see how different configurations influence it. When the user is happy with the sculpture he may continue to the accessories configurator. In this step labels and other accessories or secondary elements can be customized. In the final step a summary of the used materials and visual styles is offered and the sculpture can be then printed.

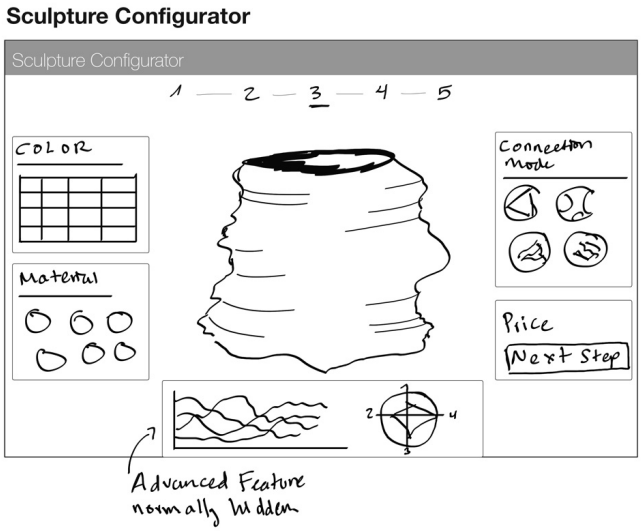


Figure 3.9: Sketch of the step based configurator prototype

In the second design iteration it was intended to simplify the interface and move from a multi-step based with step-by-step views to a basic multi-step design. The welcome screen first introduced in the first prototype would also remain for all the design iterations. After the user has logged in to the configurator the workspace view is shown. This view contains all the controls for the configurator. In the left side a panel contains several views that can be navigated. Each view has controls for a specific property. The panel shown in figure 3.11 has the variable catalog from which the user can chose variables to map to the sculpture. By selecting a variable the sculpture would be instantly updated. The geometry of the sculpture can be customized in another panel with sliders to manipulate hight and radius parameters. In this same panel a date field can be used to select the data range of the visualized data.

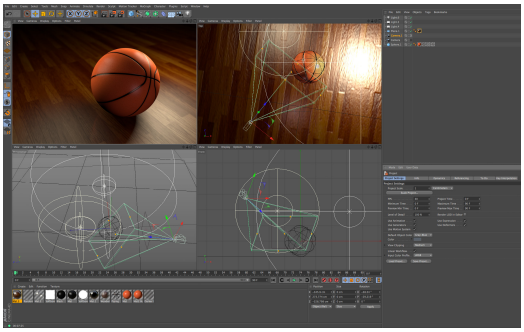


Figure 3.10: 3D modeling software Cinema 4D graphical user interface[8]

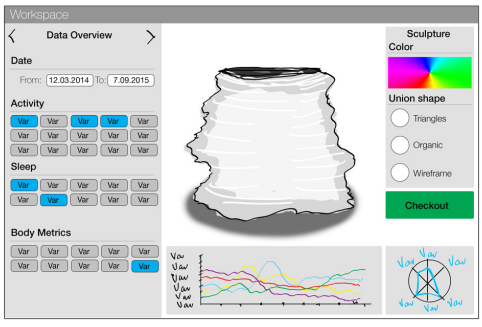


Figure 3.11: Sketch of 3D modeling software inspired configurator prototype

At the right side a panel with material and color options is shown. Again at the bottom a

line- and radial-charts visualize the selected variables. After finishing the user is brought to a summary view of the selected materials and variables visualized. A strong inspiration for this concept were 3D modeling software packages like the one shown in figure 3.10. The normal layout of a modeling software offers users an overview of the scene in different angles and a series of navigation menus and panels with material and assets windows.

The final design iteration of the configurator was conceived with the goal of further simplify the interface by hiding all panels and instead focus the users attention to the sculpture (see figure 3.12). After logging in the user is taken to the configurator. The sculpture will be surrounded by a circle with floating labels of the selected variables. To edit the visualized variables a button with a plus symbol is placed in the circle around the sculpture. If the user presses this button a panel pop ups displaying the variable list. After selecting a variable the sculpture will update instantly. This prototype has three more buttons. Floating on the right a calendar button can be used to display a slider with which the user can set the data range to be visualized. The other two buttons are used to manipulate geometry and material options.

In the following section the comparison of the 3 prototypes will be discussed, closing with an argumentation about the chosen prototype.

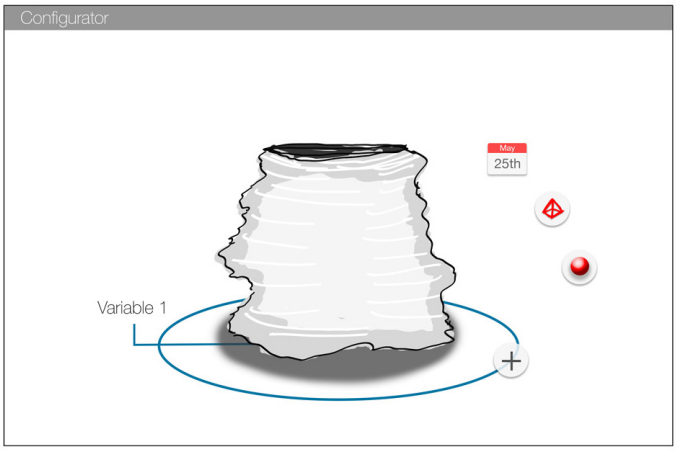


Figure 3.12: Sketch of a simplified panel-free configurator prototype

3.2.2 Prototype Validation

The main difference of the 3 prototypes is how the the user is guided through the configuration process. A welcome view in all prototypes will introduce the worklfow to the user allowing him to get comfortable with the system. The first prototype simplifies the available visual controls by dividing the process in steps. Each step is specialized in a single aspect of the configuration process, this has the benefit of allowing the user to concentrate on a single task an move on. The downside of such an implementation is that it requires state persistence functionality to store the configurations made throughout the process. For the amount of controls available for the activity sculpture configurator this option might be an overkill due that the essential steps can be broken down to two steps.

On the other hand simplifying the interface so much as in prototype 3 where all controls are accessed through buttons could offers not enough information about how they can achieve their goals. Although this prototype encourages exploration as it is the only way the user can figure out how to customize the sculpture, this takes the configurator to a more sculpting like approach.

Prototype 2 sits somewhere in the middle between prototype 1 and 3. The step clutter is reduced to a single view but through the display of control panels with text the user has visual

queues as to what the available tasks are. Through the left panel that accommodates different controls through a cycling tab system the clutter of customization controls is avoided allowing the user to concentrate in the controls that are visible in the panel at the moment.

In summary the configuration developed configurator prototype offers a simplified workflow through a basic multi-step approach design with a simple layout that displays the controls in a logically organized manner thanks to a tab panel. The available controls will be reduced to buttons, toggles and sliders that easily configurated in the system and most importantly the user can not make any mistakes, liberating the user of frustrating experiences of correcting mistakes. By having only one view the configurator reduces the implementation complexity and does not require a persistence system.

Chapter 4 is devoted to the technical implementation of the configurator's prototype discussing technologies used, the overall architecture and the challenges encountered in the developing process.

4 Implementation

The implementation of the activity sculpture configurator and the activity sculpture were undertaken in a period of around two months. The developed web configurator in this work makes use of cutting edge technologies for both web development and 3D visualization. This project is a demonstration of the capabilities of current web technologies that allow the developing of fully functional data visualization systems that offer integration with other web services for data interchange and support real time interactive 3D visualization.

This chapter will discuss all aspects of the technical implementation of the prototypes discussed in chapter 3. The technical requirements for the system will be discussed in the first section followed by an overview of the used technologies. Further on the modules conforming the system and their relationship will be explained in the architecture section. THe configurator section covers the implementation of the controls used by the user to manipulate the sculpture and the generation of the sculpture which are the main functionalities covered in the front end side of the application. The integration of the Whithings API and the manipulation performed on the received data are covered in the backend section of the chapter. Finally an overview of the challenges encountered while developing the system is presented.

4.1 Requirements

Through the analysis of current works described in chapter 2,the requirements of the prototypes in chapter 3 and in view of the forthcoming user study the functionality and technical requirements of the web-based activity sculpture creator were formulated as follows.

- Implement the designed configurator workflow views
- Users have to be able to login with their personal Withings account
- Query user's data with the Withings API
- Store user's data and profile information in a data base
- Implement user interface control widgets like buttons, sliders and toggles for the customization area of the configurator
- Implement the designed activity sculpture geometry
- The visualization of the sculpture has to be rendered in real time
- Changes made in the configurator have to be reflected in the sculpture instantly. avoid the use of update buttons
- Users have to be able to navigate the sculpture in 3D space through rotation and zooming interactions
- STL file export support for 3D printing
- Support for current browsers
- Deployment of a fully working version in a web server for the general public. Anybody with a Withings account should be able to use it an visualize their data.

The required functionality is beyond what it would be needed for a basic working prototype. The developed configurator in this work is a fully functional system that fulfills every requirement

specified in this section. The stated requirements involved the implementation of other functionality to ensure the requirement was fulfilled. For example the login functionality implies the implementation of an Oauth authentication solution(explained later in section 4.5.1) in order to integrate with the Withings API. Also because the configurator is dealing with personal data the security of the system is important, as a basic approach to address this, the configurator implements session handling.

In the following section the used technologies to tackle the extensive list of requirements will be presented.

4.2 Technology

The web configurator was developed with a modern stack of technologies that enable the use of the JavaScript language not only for frontend but also for the backend, the database and graphics programming. The MEAN stack is a collection of technologies that enables developers to write web applications in a single programming language. MEAN is the acronym for the technologies conforming the stack: *MongoDB*[42], *Express*[54], *AngularJS*[25] and *Node.js*[26]. The acronym was first used by Valeri Karpov, MongoDB engineer, in a blog post[61] where he discussed the benefits of using the same language across all technologies involved. Among the benefits Karpov states their team have experienced an increment in the performance of the developed software and also in the team's productivity. In short the MEAN stack's frameworks will be introduced.

MongoDB is an "open-source, document database designed for ease of development and scaling"[42]. Document based are conformed of collections (the equivalent of tables) which can contain several documents (rows) that to any schema are in essence JSON Objects (JavaScript Object Notation). MongoDB does not enforce any schema to the stored data, making it a very flexible environment to work with. In MongoDB queries are performed in JavaScript providing a seamless integration of the JavaScript language, from querying the information to structuring it in JSON objects. The stored JSON objects are encoded to the BSON format which stands for Binary JSON to enhance efficiency and provide additional data types. Even though working with a schema-less database brings great flexibility, it is recommendable to bring some consistency to the MongoDB documents. For this purpose an object modeling library called *mongoose*[38] has found broad adoption in the developer community. Mongoose provides functionality to easily model the application's data with type casting, custom query functions and other niceties. To illustrate how a mongoose schema looks like, listing 4.1 shows an excerpt of the user schema designed for the configurator.

Listing 4.1: An excerpt of the user mongoose schema

```
1  // load the mongoose module
2  var mongoose = require('mongoose');
3  var Schema = mongoose.Schema;
4
5  var userSchema = new Schema({
6    profile: {
7      name: String, //data type definition
8      gender: Number,
9      age: Number,
10     location: String
11   },
12 // more fields...
13   settings: {
14     startDate: Date,
15     endDate: Date,
16     show: Boolean
17   },
18   sculptures: {type: Array, "default": []}
19 });
```

Node.js[26] and **Express**[54] are the frameworks used to develop the web server of the application. Node.js is backend JavaScript, based on the "V8" chrome JavaScript runtime providing an event-driven architecture and non-blocking i/o allowing for high performance real time applications. *Node.js* is conformed of basic modules that provide network functionality but it requires the developer to build everything from the ground up. To address this issue the JavaScript library *Express* was developed. Express provides a rich set of features like routing, middleware support and HTTP utilities that allow rapid development of network applications. Listing 4.2 shows how a simple server is implemented with Node.js and Express.

Listing 4.2: Basic Node.js and Express server example

```
1  var express = require('express');
2  var app = express();
3
4  app.get('/', function (req, res) {
5    res.send('Hello World!');
6  });
7
8  var server = app.listen(3000, function () {
9    var host = server.address().address;
10   var port = server.address().port;
11   console.log('Example app listening at http://%s:%s', host, port);
12 });
```

AngularJS[25] is the fronted framework in the MEAN stack. *AngularJS* is an opens-source project developed by Google for the development of Single Page Applications (SPAs) where the whole content of a web page is rendered in a single web page and the content is either completely loaded at the beginning or its injected according to user interactions. AngularJS is based on a Model-View-Whatever design pattern (MVW) that allows developers to choose the design pattern that better accommodates their coding style. Whichever design pattern is chosen, AngularJS enforces developers to structure code into modules providing a strong organization in the code base. The functionality provided by AngularJS ranges from bidirectional data-binding, routing, form validation, server communication tools and directives that allow the creation of custom HTML syntax. With frameworks like AngularJS SPA allow to take much of the server workload to the frontend. Another feature of AngularJS is its extensibility. For the purposes of this work the *Angular Material*[24] was utilized to implement user interface controls and theming. The *Angular-Material* module is also developed by Google and it provides a solid framework for developing user interfaces utilizing the "Material design" philosophies.

The graphic visualization of the sculpture was implemented with the *Threejs*[6] library which provides an approachable abstraction of the *WebGL* API. WebGL is a standard that provides hardware accelerated 3D graphics rendering for the web[18]. Based on the OpenGL ES 2.0 standard, WebGL allows the development of real time visualizations in JavaScript through the use of the HTML 5 canvas element. *Threejs* offers developer a wide set of tools and abstractions for rapid development of graphic applications in the web. An example of a basic Threejs application is composed of a scene object that contains cameras, lights and mesh objects that will be rendered to the canvas through a WebGL context. Listing 4.3 provides a basic example of setting up a the renderer and attaching it to the DOM, loading cameras and lights, creating the scene, adding a cube mesh and finally rendering it.

In the following sections the utilization the benefits of developing with the MEAN stack with other technologies will be highlighted through examples of modules developed for the web configurator.

Listing 4.3: Basic Threejs 3D app example

```
var camera, scene, renderer;
var geometry, material, mesh;

renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );

camera = new THREE.PerspectiveCamera( 50, window.innerWidth/
    window.innerHeight, 1, 10000 );
camera.position.z = 500;

scene = new THREE.Scene();

geometry = new THREE.CubeGeometry(200, 200, 200);
material = new THREE.MeshNormalMaterial({shading: THREE.FlatShading});
mesh = new THREE.Mesh(geometry, material);

scene.add( mesh );
renderer.render( scene, camera );
```

4.3 Architecture

By using the MEAN stack to develop the web configurator certain architecture decisions are enforced. One being the use of the backend more as a RESTFul web API (Representational State Transfer) for querying data than for requesting the generation of pages. RESTful APIs define a modern architectural model over the HTTP protocol to perform service operations over a system's data[15]. A great advantage of utilizing RESTful architectures is that it provides good scalability as the backend can be completely decoupled from the client implementation, meaning one backend can be used to provide data to all sorts of clients from web clients to mobile or desktop applications. The configurator's backend is composed of a web service that implements the Withings API for authentication and requesting sensor data, session handling for assuring only logged in users can access the configurator's backend, the user data API that interacts with the MongoDB database to perform CRUD (Create, Read, Update, Delete) operations on the gathered user data and WebSocket communication support. WebSocket is a protocol based on a TCP connection that provides full-duplex communication to client-server applications[14]. The advantage of WebSockets is that it allows servers to push data to clients without being requested. The configurator makes use of WebSockets through the *Socket.io*[9] library which implements the use of WebSockets in an event-driven pattern. Third party libraries utilized in the backend are contained in Node.js modules. Figure 4.1 provides an overview of the modules conforming the backend and frontend sides of the configurator.

The frontend of the configurator is a SPA implemented in AngularJS. As discussed before, the backend is implemented as a Web API, relegating the generation of the HTML content to the AngularJS application. The frontend implementation follows a Model View Controller (MVC) pattern for better separation of concerns. Due to the flexibility of AngularJS this is was implemented through the extensive use of controllers attached to views. In AngularJS views are basic HTML templates that describe the visual aspect of the web site. Controllers implement logic functionality to manipulate the application's data or model. The model part of the MVC pattern is implemented by Angular services. Services in Angular are modules that can interface with databases through web APIs or they can be used to abstract functionality and reuse it across the application. Views can host several controllers who at the same time can access several services.

Even though views in the configurator are divided in several parts for re-usability across the application, they where separated in three main areas: the core, the configurator and the dashboard.

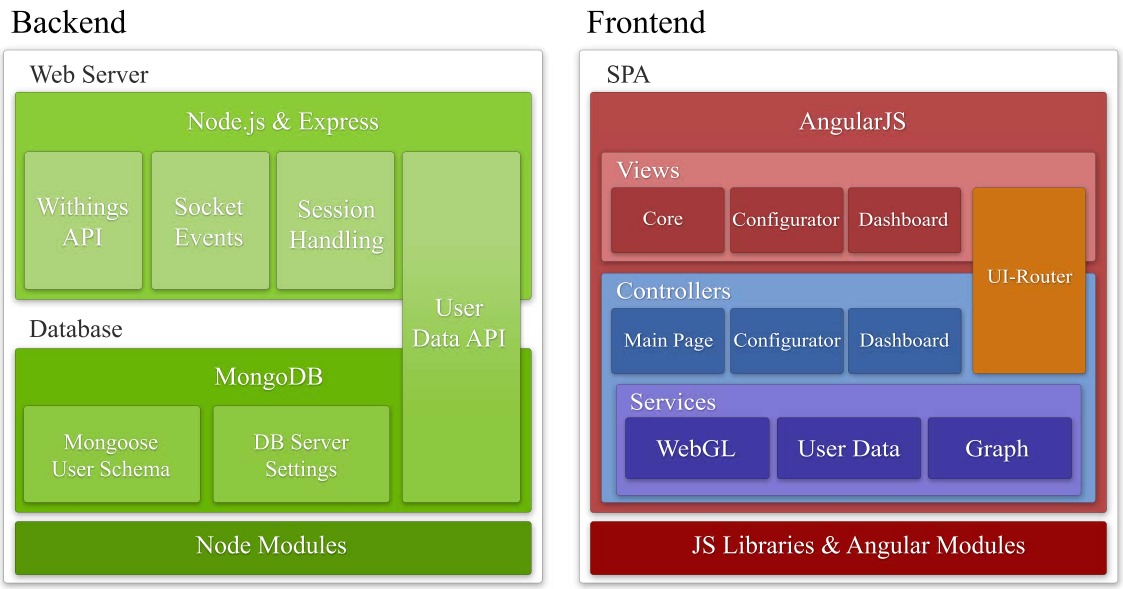


Figure 4.1: Activity Sculpture web configurator's architecture

The core comprehends partial views utilized for the welcome page, the consent window and the 404 error page. The configurator's view contains several modules with User Interface (UI) controls and the canvas for the sculpture. The dashboard view is composed of the dashboard view that shows the sculpture gallery and charts visualizing the user's data, and the setup form.

The attached controllers to these views provide the logic to achieve the intention of each section. Due to the specific requirements of each section in the configurator, the controllers are also organized in modules for the main page, the configurator and the dashboard. The main page controller provides the logic for starting the authentication process. The configurator controller is the most complex as it implements the logic the UI controls and the rendering of the sculpture. The configurator controller makes heavy use of the graphics service module, which manages Threejs scenes, cameras and the generation of the sculpture. Further more the configurator controller reaches to the user data service to load the data for the generation of the sculpture. The user data service contains several modules for data manipulation and transforming and provides a user object that holds all the data received from the Withings API. The user object follows a singleton pattern implementation, making a single instance of the object available throughout the application. This is specially helpful as the dashboard configurator also needs the data provided by the user object in order to visualize generate the charts and load the sculpture gallery. For the generation of charts the dashboard controller makes use of a graph service that implements a JavaScript SVG library called *D3.js*[4], which stands for "Data driven documents". The *D3.js* library has gained broad popularity in web data visualization as it provides great flexibility through data-driven DOM manipulation and the use of widely supported CSS, SVG and JavaScript technologies.

An important part of every AngularJS application is the mapping of views and controllers to routes. In SPAs the rendering of websites is performed client side and not requested from the backend as in other architectures. In order to achieve this the application defines routes that are in essence URL addresses that represent a certain part of the application. When a route is called the application maps the address to a view with controllers presenting to the user the requested site. AngularJS provides a routing module but for this work a third party angular module called UI-Router was utilized. The UI-Router framework developed by the Angular-UI team[57] provides routing functionality by organizing views into a state machine, unlike AngularJS that is based on URLs. Every state can have a controller and a view with child views assigned to it, it is not strictly

necessary to map a state to a URL but it can be done. Added to that states can contain nested stated states for complex routing implementations.

Other secondary functionality is implemented through the use of JavaScript libraries and angular modules.

In order to better illustrate how the different parts of the application interact with each other a flow diagram of the application can be seen in figure 4.2 and 4.3. The flow diagram explains in a simplified way the tasks the client, the server, the Withings API and the database perform and the actions the user can perform. The diagram also offers information as of which steps are involved in each of the parts of the application going from the very start where the user starts the configurator in the browser, going over the authentication process, setting up user configurations, presenting the dashboard and finally the configuration process. In upcoming sections some of the most important steps are going to be explained more in depth.

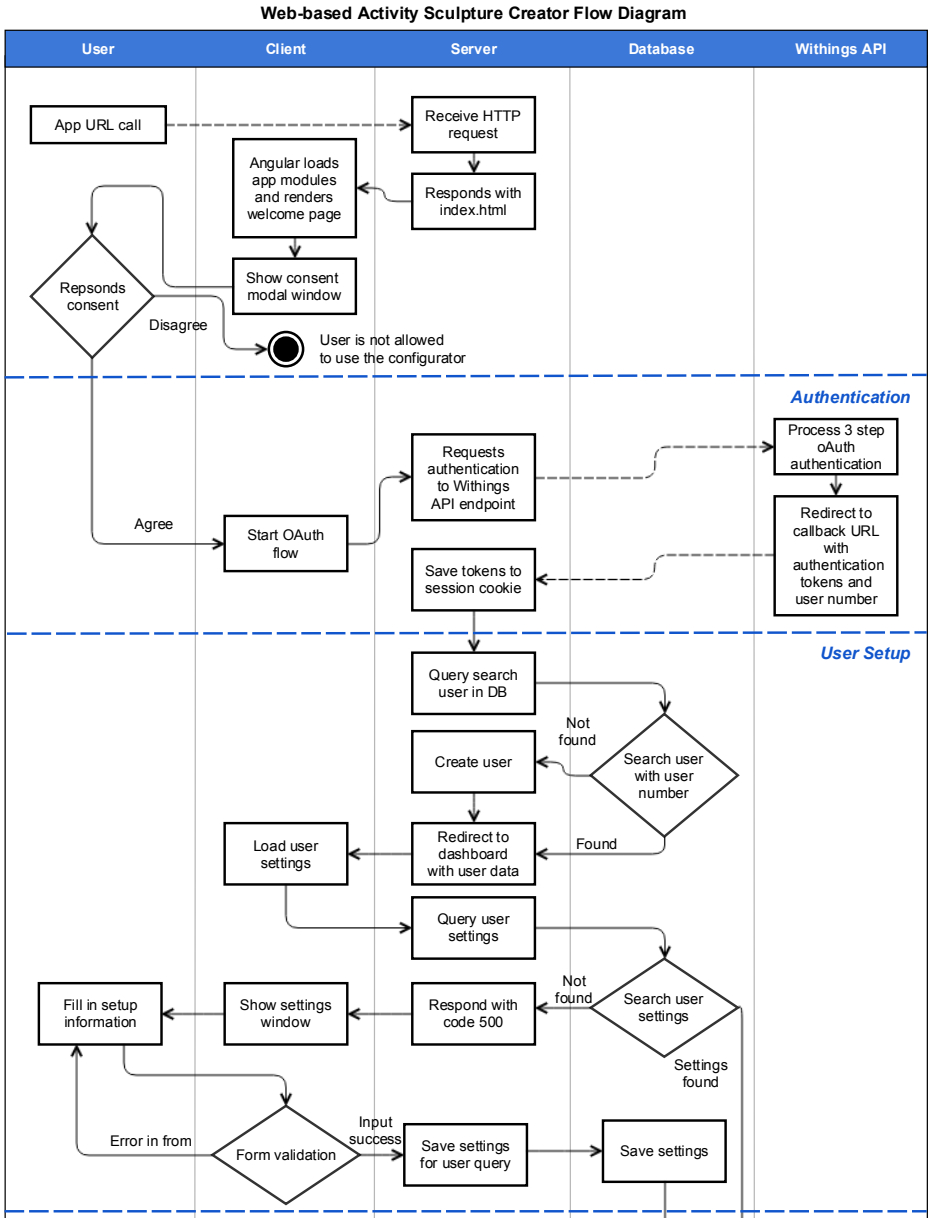


Figure 4.2: Flow diagram part 1

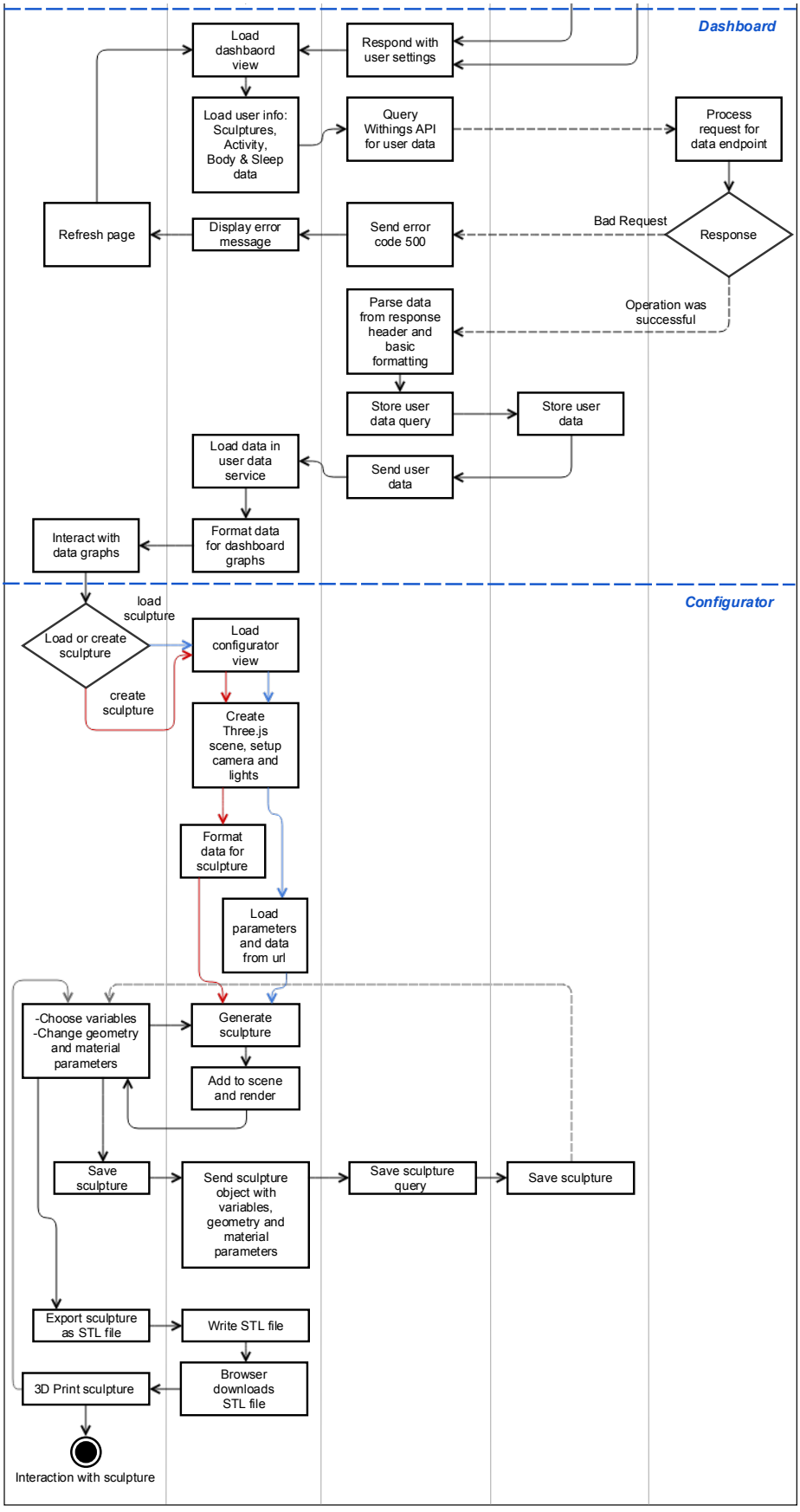


Figure 4.3: Flow diagram part 2

4.4 Configurator

The configurator section of the activity sculpture creator has two main tasks. The first task being offering users a set of controls for selecting the variables to be mapped into the sculpture and for customizing different aspects of the sculpture. The second task consist in generating the activity sculpture based on the parameters set by the user and rendering it. The steps involved in the execution of both of these tasks are performed by the Angular Application and the interaction of the user (see last section of figure 4.3).

4.4.1 Sculpture Generation & Rendering

The generation and rendering of the sculpture is contained in the graphics service module of the application. This service module is composed of 5 submodules: *wcViewport*, *wcScene*, *wcCamera*, *wcModel* and the *WCVaseGeometry* class.

The *wcScene* is an Angular service that exposes the Threejs scene object. In the same way the *wcCamera* service contains code for configuring the camera and exposing it to the rest of the application. The *wcViewport* submodule is an angular directive, a custom DOM element or attribute attached to an element that extend the element with defined functionality. In the case of the wcViewport it contains setup code for attaching the threejs renderer to the canvas element in the configurator view and it adds the camera from the *wcCamera* service to the *wcScene* scene object and ultimately renders everything. The *wcModel* service manages the generation of the sculpture by passing the parameters of the UI controllers to the *WCVaseGeometry*, and through the *wcScene* service adds the generated sculpture to the scene object which is rendered every frame.

The algorithms employed for the generation of the activity vase geometry in the *WCVaseGeometry* class follow the same principles for generating a circular cylinder with equal top and bottom radii. The parameters involved are radius, height and number of height and radial segments. The first two parameters are obvious, radius defines the size of the top and bottom circles and height sets the size in the Y axis. The radial segments define the number of segments that will conform the bottom and top circles. In the same way hight segments are used to define the number of divisions for the sculptures hight. In other words radial and hight segments With the parameters set, generating the sculpture would require calculating the position vertices of the radial segments for each hight segment. For the activity vase some enhancements where made to this simple principle. As stated in the prototyping phase of the activity sculpture 3.1.4 the idea is to stack radial diagram each with a set of axis representing the activity of one day. In the implementation of *WCVaseGeometry* the radial segments represent the chosen data variables. Each axis will have a different value as the represent different data variables with different magnitudes and units, for example for one day the step count could be 8000 steps and heart-rate 110 bpm. To address this issue all values are normalized to a range that is then added to a ground radius. The hight segments represent the number of days visualized. The algorithm calculates for every day the vertex position of each radial segment from the chosen variables by normalizing the data values and adding that to the base radius. The *WCVaseGeometry* integrates seamlessly with the Threejs. For this in the generation of the sculpture other information had to be generated in order to best integrate with Threejs like calculating vertex and face normals which are needed by some lighting techniques and for mapping textures and materials. Because of the radial diagram concept behind the sculpture does not work well for sculpture visualizing less than 3 variables, further logic had to be implemented in order to always ensure the generation of a 3D sculpture. With one variable the algorithm did not render anything at all, and with two variables the sculpture was a plane with the contours of the axes. This issue was addressed by extruding the same axis over the radius with a predefined number of radial segments. When two variables where chosen, half of the radial segments displayed one variable's values and the second half the other variable's values.

In order to provide more customization possibilities two more visualization styles where de-

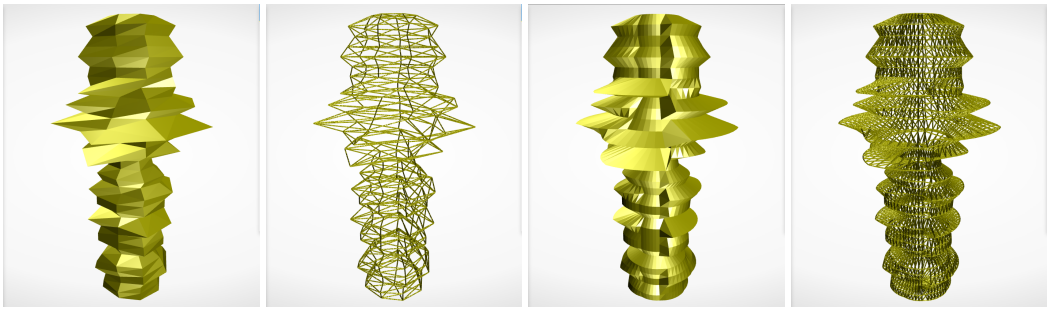


Figure 4.4: Normal and interpolated styles with their wire-frame visualization

veloped. The first is a simple wire-frame visualization that renders only the edges of the faces representing the sculpture. The second style softens each corner's axis by adding more radial segments between each axis, which is achieved through linear interpolation. Figure 4.4 shows the implemented visualization styles for the activity vase.

For further technical details regarding the implementation of the activity vase, the complete source code for the *WCVaseGeometry* class can be found in the Appendix B.2 of this work.

4.4.2 Sculpture Manipulation

The customization of the sculpture is implemented in the configurator view and its controller who makes use of the graphic and api service modules. When the user calls starts the configurator for creating a sculpture the configurator view loads two panels filled with UI-controls and the *wcViewport* for setting up the visualization engine.

The customization possibilities are influenced by the design and the implementation of the activity sculpture. The panels displayed by the configurator's view are placed at each side of the screen leaving the sculpture in the middle. The left panel contains two tabs which can be selected for manipulating different parameters. The first tab is the "Data" tab and offers an overview of the data variables categorized in activity, body and sleep. Under activity users can choose the from three levels of activity intensity duration namely intense, moderate and soft, elevation, calories burned, distance walked and step count. The body category offers heart pulse and SPO2 short for peripheral capillary oxygen saturation. The sleep category allows users to visualize the duration to sleep time, wake up count and duration, and the duration of deep and light sleep. The second tab in the left panel provides controls for manipulating the geometry of the sculpture. Through slider widgets users can manipulate the size of the base radius, the height and the days visualized. Through toggle widgets users can toggle the interpolate visualization style. If turned on the sculpture will immediately update and a slider for manipulating the number of segments will appear. For reference purposes the labels for each variable are displayed under the sculpture. The display of the labels can also be toggled on off from the geometry tab. The panel on the right has contains two tabs, one for material configurations and a second one for sculpture saving and export. The material tab allows users to change the color of the sculpture through a color picker widget. A slider for the material shininess is also provided. Lastly a toggle widget for activating the wire-frame visualization style is provided. If activated a slider for adjusting the thickness of the wire-frame lines appears. The export tab provides a summary showing the data range of the visualized data and a list of the visualized variables. Under the summary, the save and export section allows users to save the sculpture or export it to a STL file.

All user interaction is implemented through data binding from the configurator's view to the controller. Data binding provides a means for synchronizing data from the view to the model's data. When the user manipulates a slider, toggle or the color picker in the panels the data is synchronized in the controller and passed to the *wcModel* service which then updates the sculpture.

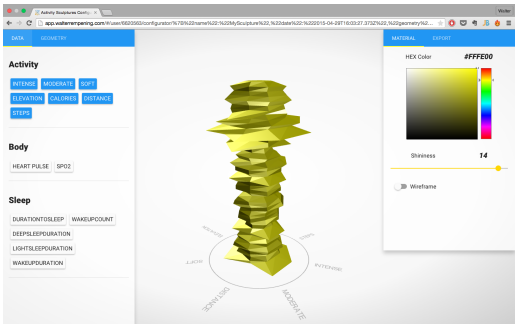


Figure 4.5: Data and material tabs active

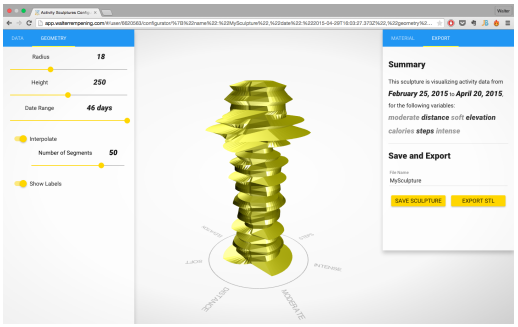


Figure 4.6: Geometry and export tabs active

For selecting the variables the process more complex. When the user selects a variable button the variable's name is stored in an array that is used in the controller for extracting the data from the User object provided by the *wcUserData* service. The controller adds or removes variable's data according to the actions of the user and passes it to the *wcModel* in charge of updating the sculpture. Assigning a controller to the view is as simple as adding the "ng-controller" directive in the view and assigning the desired controller (see line 2 of listing 4.4). Once the controller is assigned the view can access controller fields and methods. Angular utilizes a double curly braces syntax for reading controller data as seen in line 9 of listing 4.4 where a span element is displaying the current value of the height segments in the UI geometry parameters objects. For updating the actual value of the height segments the slider widget implemented through the *angular-material* directive "md-slider" which contains the ng-model angular directive that is used to update the controller's field as seen line 11 of the same listing.

Listing 4.4: Example of bidirectional data binding in AngularJS

```
1  <!-- controller-view.html -->
2  <div ng-controller="PanelController as pctrl">
3    <!-- ... -->
4    <div flex="10" layout="row" layout-align="space-around center">
5        <span class="slider-name">Date Range</span>
6      <span class="slider-value">{{ pctrl.uiGeoParams.heightSegments }}
         days</span>
7    </div>
8    <md-slider flex
9      min="{{ pctrl.sliderParams.heightSegments.min }}"
10     max="{{ pctrl.sliderParams.heightSegments.max }}"
11     ng-model='pctrl.uiGeoParams.heightSegments'
12     ng-change="pctrl.onUiParamsChange()" >
13   </md-slider>
14   <!-- ... -->
15 </div>
```

For updating the sculpture every time the user changes a parameter all widgets call the "onUiParamsChange()" method through the ng-change angular directive in line 12 of listing 4.4, which listens for input events. Listing 4.5 shows an excerpt of the controller's code where the slider parameter object is defined and the "onUiParamsChange()" method is called. This code example shows how easily is to bind controller's data into the view, not only for displaying or synchronization purposes but the controller's fields can also be used to set constrains and other attributes for UI-controls. The PanelCotnroller's code in listing 4.5 also exemplifies how other services can be injected to the controller. On lines 2 through 4 the controller injects the *ModelService* from the *wcModel* module. On line 15 the controller calls the "updateMesh" method from the *ModelService*.

Listing 4.5: Binded controller data and the called controller function from the view

```
// PanelController.js
var controlls = angular.module( 'wcControlls', [] );
controlls.controller( 'PanelController',
  ['ModelService', function ( ModelService ) {
  // Configuration object
  this.sliderParams = {
      heightSegments: {
        min: 1,
          max: $scope.data.values.length - 1
      },
  // definition of more fields...
  }

  this.onUiParamsChange = function () {
    ModelService.updateMesh( this.uiGeoParams, this.uiMatParams );
  };
}]);
```

The implementation of the "updateMesh()" method is described in listing 4.6. The "updateMesh()" method requires two objects containing geometry and material parameters which are then passed to the *WCVaseGeometry* class in the "makeSculpture()" method on line 11 for updating the sculpture. This is achieved by looking in the *SceneService* for the old sculpture removing it, creating a new one and adding it to the scene object again (see lines 7 through 12). The *ModelService* exemplifies how Angular encourages strong concern separation through the use of services and modules that can be injected to controllers and services across the whole application taking advantage of reusing code as much as possible.

Listing 4.6: Sculpture update function

```
// wcModel.js
var model = angular.module( 'wcModel', [] )
model.service( 'ModelService',
  ['SceneService', function ( SceneService ) {

  this.updateMesh = function ( geoArgs, matArgs ) {
      var oldSculpture = SceneService.scene.getObjectByName( 'vase' );
      if ( oldSculpture !== undefined ) {
        SceneService.scene.remove( oldSculpture );
      }
      sculpture = makeSculpture( geoArgs, matArgs );
      SceneService.scene.add( sculpture );
  };
}]);
```

The configurator is an example of how the use of SPA frameworks like AngularJS are a suitable option for the implementation of highly interactive web applications that integrate well with other libraries like Threejs or D3js. As a side note, due to space constraints and the high complexity of the code base, the source code showed in this chapter was simplified for explanation purposes. For the full source code refer to the appendix B.1.

4.5 Backend

The backend of the activity sculpture configurator serves as a web service from which the configurator's front end queries user data that is processed by the web server through the Withings API and some data manipulation modules. The following section attempts to explain how the Withings API integrates with other parts of the web server and the subsequently data manipulation tasks performed.

4.5.1 Withings API Integration

The Withings API is used to complete to tasks: user authentication and data import. In order to have access to user's data from third party applications as the configurator, users have to give these applications their approval. The Withings API implements authentication through the *OAuth* 1.0 protocol. *OAuth* is an open standard that provides a process for authorization of third-party applications to server resources or for clients to access server resources using credentials of other web services[19]. *OAuth* is commonly used in web services where users may login in using their twitter, facebook or google account. The process of authentication is performed through several HTTP calls. The Withings API requires third party developers to create a developer account and register the application. Every application is assigned an API key and secret. These keys are needed for the *OAuth* process as they are have to be present in every HTTP call. The authentication process involves 3 steps. For the Withings API developers first request an end-user token, a successful response from the API provides an oauth token and oauth token secret. With the received tokens developers redirect users to web site where they can login with their credentials and authorize the third-party application the use of their data. If the authorization was granted the developer receives an oauth token and the user id. The final step consists in requesting a user data access token which once received allows third party applications to access data for the specified user.

In order to simplify the process of generation the URLs for the HTTP calls in the *OAuth* authentication process the web configurator makes use of the *Passport*[20] framework which supports different authentication methods and also provides session handling. Being also written in JavaScript the framework integrates seamlessly with the backend's Node.js and Express architecture. Through the use of the *Passport* frame work the whole *OAuth* authentication process is reduced to only to two API calls (see listing 4.7). The first API call (lines 2 through 5) defines the address the client calls when login in to the configurator. *Passport* uses preconfigured API Keys from and Withings developers account and starts the token interchange process. After clicking the login button the user will be redirected to a Withings website where his credentials are required. After he has granted permission to the configurator to access his data he will be redirected to a predefined callback URL address which is the configruator's webserver's second API call (lines 7 through 13 of listing) that will store the received user data access tokens in a cookie and redirects the user to the Angular frontend.

Listing 4.7: Passport OAuth authentication implementation

```
1  // routes.js
2  app.get('/auth/withings', passport.authenticate('withings'),
3      function(req, res, next) {
4        console.log("Withings OAuth flow started");
5  });
6
7  app.get('/auth/withings/callback', passport.authenticate('withings',
8    {failureRedirect: '/' }),
9    function(req, res, next) {
10     res.cookie('user', JSON.stringify(req.user), {maxAge: 2592000000});
11     res.redirect('/#/user/' + req.user.id);
12   }
13 );
```

After this point the configurator's backend is able to request user's data through the Withings API. The Withings API provides access to all measured information about the user being it activity, body and sleep measures, which can be requested for a single date or for a date range from up to 200 days. As an example listing 4.8 shows example code for a query for a single days activity data and the received data in a JSON object. Every Withings API request URL has to contain a series of parameters like the oauth consumer key, signature, token and others, which makes the generation of the requests quite cumbersome. To address this issue a *Passport* extension module

Listing 4.8: Withings API query and response example

```
// Request URL
https://wbsapi.withings.net/v2/measure?action=getactivity&userid=29&
    date=2015-05-15&more_parameters...

// API JSON response
{
   "status": 0,
   "body": {
       "date":"2015-05-15",
       "steps":6523,
       "distance":4600,
       "calories":408.52
       "elevation":18.2,
       "soft": 5880,
       "moderate": 1080,
       "intense": 540,
       "timezone":"Europe/Berlin"
    }
}
```

called **Withings-API**[66] was used. This module abstracts the generation of request URLs to a JavaScript object making it a comfortable way of managing API calls.

4.5.2 Data Processing

The JSON objects in which the Withings API packs the response data integrate well with the configurator's architecture because *MongoDB* works seamlessly with JSON objects allowing for direct storage of the received data. The requests performed by the configurator's backend ask always for several days worth of data and unfortunately the received JSON object contains the data not ordered by date but in semi random order. Before storing the data to the database a sorting algorithm is performed on the data and stored ordered by date.

The configurator visualizes the activity data in the charts displayed in the dashboard section and in the activity sculpture in the configurator section. Due that the data visualization in both sections is performed by different frameworks namely D3js and Threejs respectively, each require the data in a unique format. Because the visualizations are rendered by the frontend the formatting and processing of the data is performed by the api module consisting of the *wcUserData* service and a utility class for data transformations called *wcDataUtils*. The *wcUserData* stores a User object that contains all the activity data, stored sculptures for the user and the system settings configured by the user. For other controllers to gain access to the data, they only have to include this service and depending on the visualization to be performed the *wcDataUtils* class contains formatting methods that return the data with the correct format for the needed visualization type.

4.6 Challenges

5 User Study

5.1 Study Design

5.2 Questionnaire

5.3 Participants

5.4 Procedure

5.5 Results

5.6 Limitations

6 Conclusion

7 Future Work

List of Figures

Listings

References

[1] C. '74'. Cylcing '74: makers of max', 2015. last accessed 25-05-2015, https://cycling74.com/.

[2] E. Abbasi, A. Hubaux, M. Acher, Q. Boucher, and P. Heymans. What's in a web configurator? empirical results from 111 cases. University of Belgium, 2013.

[3] F. Bara, E. Gentaz, P. Colé, and L. Sprenger-Charolles. The visuo-haptic and haptic exploration of letters increases the kindergarten children's understanding of the alphabetic principle. *Cognitive development*, 19(3):433–449, 2004.

[4] M. Bostock. D3 data driven documents, 2013. last accessed 22-05-2015, http://d3js.org/.

[5] T. bvba. 3d trophy factory, 2015. last accessed 22-05-2015, http://www.3dtrophyfactory.com/en/start-creating/tr_texttotem_large.

[6] R. Cabello. Three.js, 2010. last accessed 22-05-2015, http://threejs.org/.

[7] S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[8] M. Computer. Maxon 3d for the real world, 2015. last accessed 27-05-2015, http://www.maxon.net/de/products/cinema-4d-studio.html.

[9] CONTRIBUTORS. socket.io, 2015. last accessed 30-05-2015, http://socket.io/.

[10] D. Coulter. Multiple exposure 28, 2015. last accessed 27-05-2015, http://www.dylancoulter.com/Multiple-Exposure/28.

[11] I. FDIS. 9241-110: 2006. ergonomics of human system interaction-part 110: Dialogue principles. *International Organization for Standardization (ISO). Switzerland*, 2006.

[12] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. *Knowledge-based configuration: From research to business cases*. Newnes, 2014.

[13] M. C. Felice, J. B. Ferreira Filho, M. Acher, A. Blouin, and O. Barais. Interactive visualisation of products in online configurators. *17th International Software Product Line Conference co-located workshops*, 2013.

[14] I. Fette and A. Melnikov. The websocket protocol. 2011.

[15] R. T. Fielding and R. N. Taylor. Principled design of the modern web architecture. In *Proceedings of the 22Nd International Conference on Software Engineering*, ICSE '00, pages 407–416, New York, NY, USA, 2000. ACM.

[16] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the Future*, 2007:1–16, 2012.

[17] U. GmbH. Unu embrace urban life, 2014. last accessed 20-05-2015, https://unumotors.com/get-your-unu.html.

[18] K. Group. Webgl, opengl es 2.0 for the web, 2015. last accessed 30-05-2015, https://www.khronos.org/webgl/.

[19] E. Hammer-Lahav. The oauth 1.0 protocol. 2010.

[20] J. Hanson. Passport, 2015. last accessed 30-05-2015, http://passportjs.org/.

[21] G. Hedin, L. Ohlsson, and J. McKenna. Product configuration using object oriented grammars. In *System Configuration Management*, pages 107–126. Springer, 1998.

[22] F. Inc. Fitbit, 2015. last accessed 26-05-2015, https://www.fitbit.com/de/flex.

[23] F. Inc. Fitbit flex wireless activity & sleep wristband, 2015. last accessed 26-05-2015, https://www.fitbit.com/de/flex.

[24] G. Inc. Angular material, 2015. last accessed 30-05-2015, https://material.angularjs.org/latest/#/.

[25] G. Inc. Angularjs, 2015. last accessed 30-05-2015, https://angularjs.org/.

[26] J. Inc. Node.js, 2012. last accessed 22-05-2015, http://nodejs.org/.

[27] N. Inc. Nike+ developer portal, 2014. last accessed 26-05-2015, https://developer.nike.com/content/nike-developer-cq/us/en_us/index/documentation/api-docs.html.

[28] N. Inc. Nike+, 2015. last accessed 26-05-2015, https://secure-nikeplus.nike.com/plus/products/.

[29] N. Inc. Nike+ running gps app, 2015. last accessed 26-05-2015, http://www.nike.com/us/en_us/c/running/nikeplus/gps-app.

[30] N. S. Inc. hyphae: surface growth, 2010. last accessed 27-05-2015, http://n-e-r-v-o-u-s.com/projects/albums/networks-sketches/content/hyphae-surface-growth-1/.

[31] N. S. Inc. Nervous system, 2015. last accessed 27-05-2015, http://n-e-r-v-o-u-s.com/.

[32] Y. Jansen, P. Dragicevic, and J.-D. Fekete. Evaluating the efficiency of physical visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2593–2602. ACM, 2013.

[33] Jawbone. Buy up3 by jawbone the most advanced tracker known to man, 2015. last accessed 26-05-2015, https://jawbone.com/store/buy/up3.

[34] Jawbone. Up for developers, 2015. last accessed 26-05-2015, https://jawbone.com/up/developer.

[35] T. jQuery Foundation. jquery write less, do more, 2015. last accessed 22-05-2015, https://jquery.com/.

[36] R. A. Khot, L. Hjorth, and F. Mueller. Understanding physical activity through 3d printed material artifacts. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3835–3844. ACM, 2014.

[37] R. Konstanzer, D. Koenig, and I. Andrei. Volkswagen car configurator. Mar 2007.

[38] LearnBoost. Mongoose, 2015. last accessed 30-05-2015, http://mongoosejs.com/.

[39] M. Maria. Mr maria shop all sweetness and light, 2015. last accessed 22-05-2015, http://mrmaria.com/customizable/.

[40] V. Mayer-Schönberger and K. Cukier. *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, 2013.

[41] R. McCarney, J. Warner, S. Iliffe, R. van Haselen, M. Griffin, and P. Fisher. The hawthorne effect: a randomised, controlled trial. *BMC medical research methodology*, 7(1):30, 2007.

[42] I. MongoDB. Mongodb, 2015. last accessed 30-05-2015, https://www.mongodb.org/.

[43] NeuroSky. Biosensors & algorithms ,ecg, eeg, neursky, 2015. last accessed 25-05-2015, http://neurosky.com/biosensors/eeg-sensor/biosensors/.

[44] Nicholas and M. Desbiens. Fahz, it's your face in a vace, 2015. last accessed 27-05-2015, http://www.fahzface.com/.

[45] E. N. of Living Labs. European network of living labs, 2014. last accessed 21-05-2015, http://www.openlivinglabs.eu/aboutus.

[46] C. K. Ogden, I. A. Richards, B. Malinowski, and F. G. Crookshank. *The meaning of meaning*. Harcourt, Brace & World New York, 1946.

[47] J. Olsen. Realtime procedural terrain generation. 2004.

[48] F. T. Piller. User innovation: Der kunde als initiator und beteiligter im innovationsprozess. *Drossou, O., Krempl, S.(Hg.): Open Innovation. Freier Austausch von Wissen als soziales, politisches und wirtschaftliches Erfolgsmodell. Hannover*, pages 85–97, 2006.

[49] B. J. Pine. *Mass customization: the new frontier in business competition*. Harvard Business Press, 1999.

[50] I. Popian. Mental fabrications, 2015. last accessed 25-05-2015, http://www.3dtrophyfactory.com/en/start-creating/tr_texttotem_large.

[51] C. P. Release. 3d printing market to grow to us$16.2 billion in 2018, 2014.

[52] R. Rolland, E. Yvain, O. Christmann, E. Loup-Escande, and S. Richir. E-commerce and web 3d for involving the customer in the design process: the case of a gates 3d configurator. In *Proceedings of the 2012 Virtual Reality International Conference*, page 25. ACM, 2012.

[53] E. Rubin. *Visuell wahrgenommene figuren*. Ripol Classic, 1921.

[54] StrongLoop. Express, 2015. last accessed 30-05-2015, http://expressjs.com/.

[55] S. Swaminathan, C. Shi, Y. Jansen, P. Dragicevic, L. A. Oehlberg, and J.-D. Fekete. Supporting the design and fabrication of physical visualizations. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3845–3854. ACM, 2014.

[56] M. Swan. The quantified self: fundamental disruption in big data science and biological discovery. *Big Data*, 1(2):85–99, 2013.

[57] A.-U. Team. Angular-ui/ui-router, 2015. last accessed 1-06-2015, https://github.com/angular-ui/ui-router.

[58] U. Technologies. Unity game engine, 2015. last accessed 21-05-2015, http://www.unity3d.com.

[59] Timbuk2. Timbuk2 custom classic messenger bag, 2014. last accessed 20-05-2015, http://www.timbuk2.com/customizer#/product/18-custom-classic-messenger-bag/size/2/customize.

[60] Twikit. Twikit, tweak it, make it! 3d customization, 2015. last accessed 22-05-2015, https://www.twikit.com/.

[61] K. T. e. a. M. Valeri Karpov. The mean stack: Mongodb, expressjs, angularjs and node.js, 2013. last accessed 29-05-2015, http://blog.mongodb.org/post/49262866911/the-mean-stack-mongodb-expressjs-angularjs-and.

[62] A. Vande Moere. Beyond the tyranny of the pixel: Exploring the physicality of information visualization. In *Information Visualisation, 2008. IV'08. 12th International Conference*, pages 469–474. IEEE, 2008.

[63] A. Vande Moere and S. Patel. The physical visualization of information: Designing data sculptures in an educational context. *Visual Information Communications International (VINCI'09)*, 2009.

[64] M. Wearables. Build misfit, 2015. last accessed 26-05-2015, https://build.misfit.com/docs/start.

[65] M. Wearables. Misfit shine premium fitness + sleep monitor, 2015. last accessed 26-05-2015, http://misfit.com/products/shine#shine-trackitall.

[66] Withings. Withings documentation, 2015. last accessed 26-05-2015, http://oauth.withings.com/api.

[67] Withings. Withings pulse ox, 2015. last accessed 26-05-2015, http://www.withings.com/de/withings-pulse.html.

[68] G. Wolf. Know thyself: Tracking every facet of life, from sleep to mood to pain. *Wired Magazine*, 24(7):365, 2009. http://archive.wired.com/medtech/health/magazine/17-07/lbnp_knowthyself?currentPage=all.

[69] G. Wolf. The quantified self, self knowledge through numbers, May 2012. last accessed 19-05-2015, http://www.quantifiedself.com/.

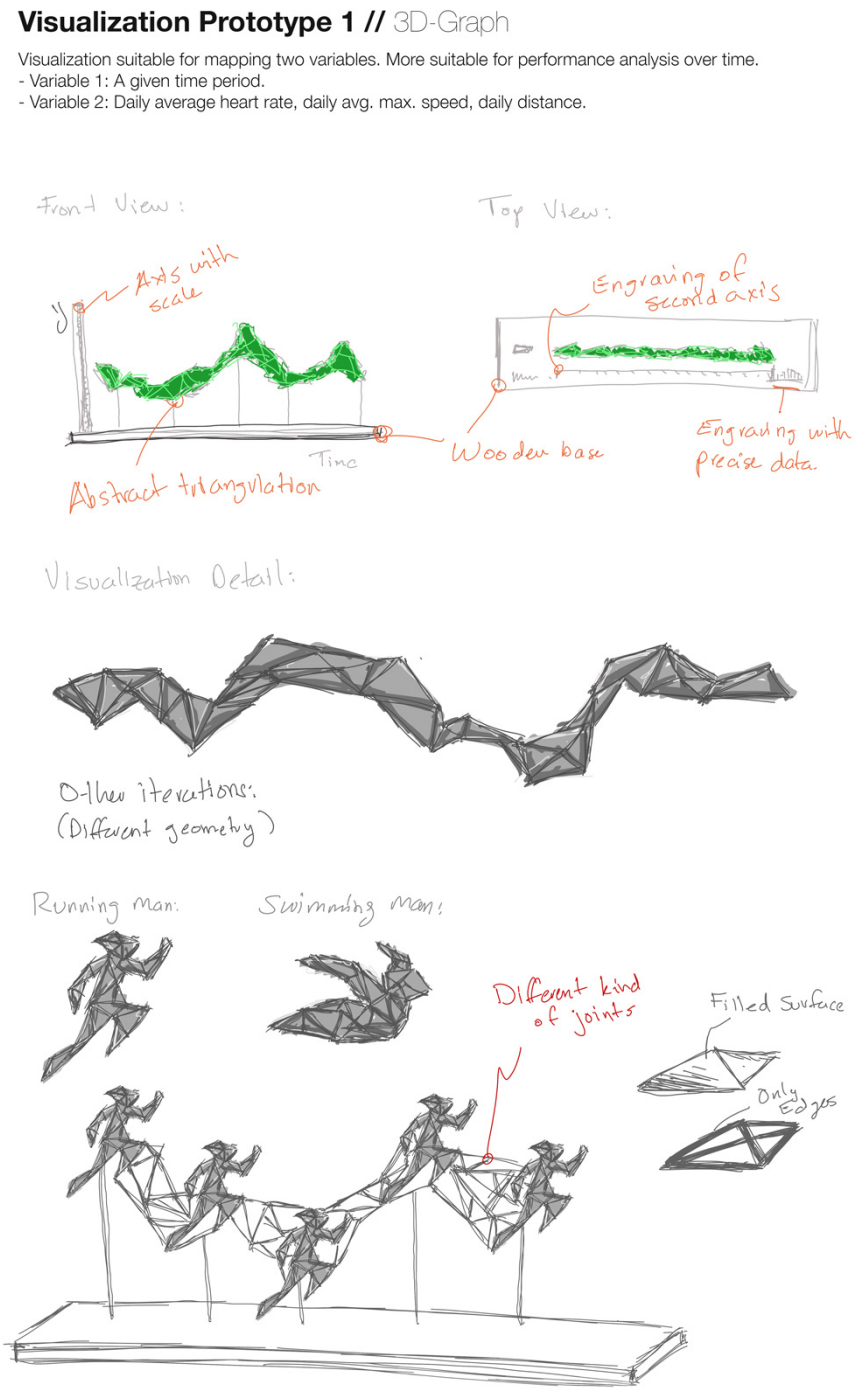
[70] J. Zhao and A. V. Moere. Embodiment in data sculpture: a model of the physical visualization of information. In *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*, pages 343–350. ACM, 2008.

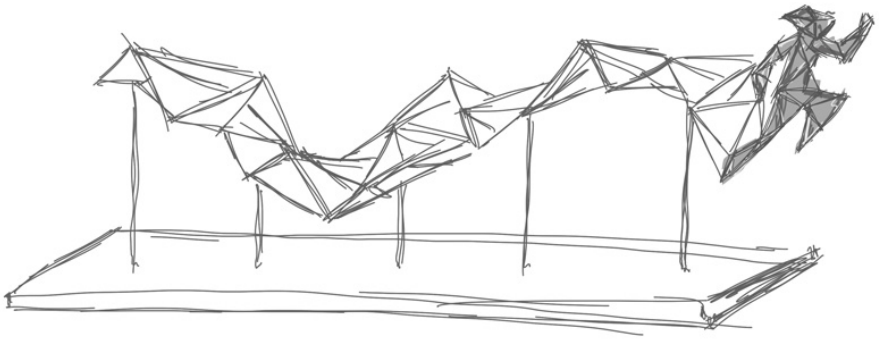
Appendix

A Prototype Sketches

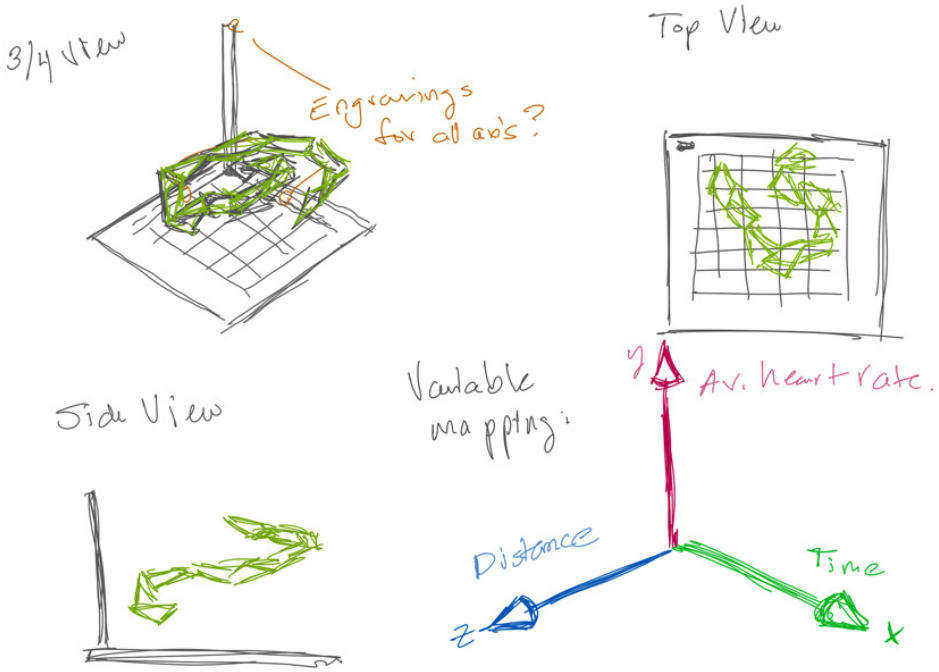
A.1 Sculpture Prototypes

3D Graph Sculpture Prototype (continues in next page)

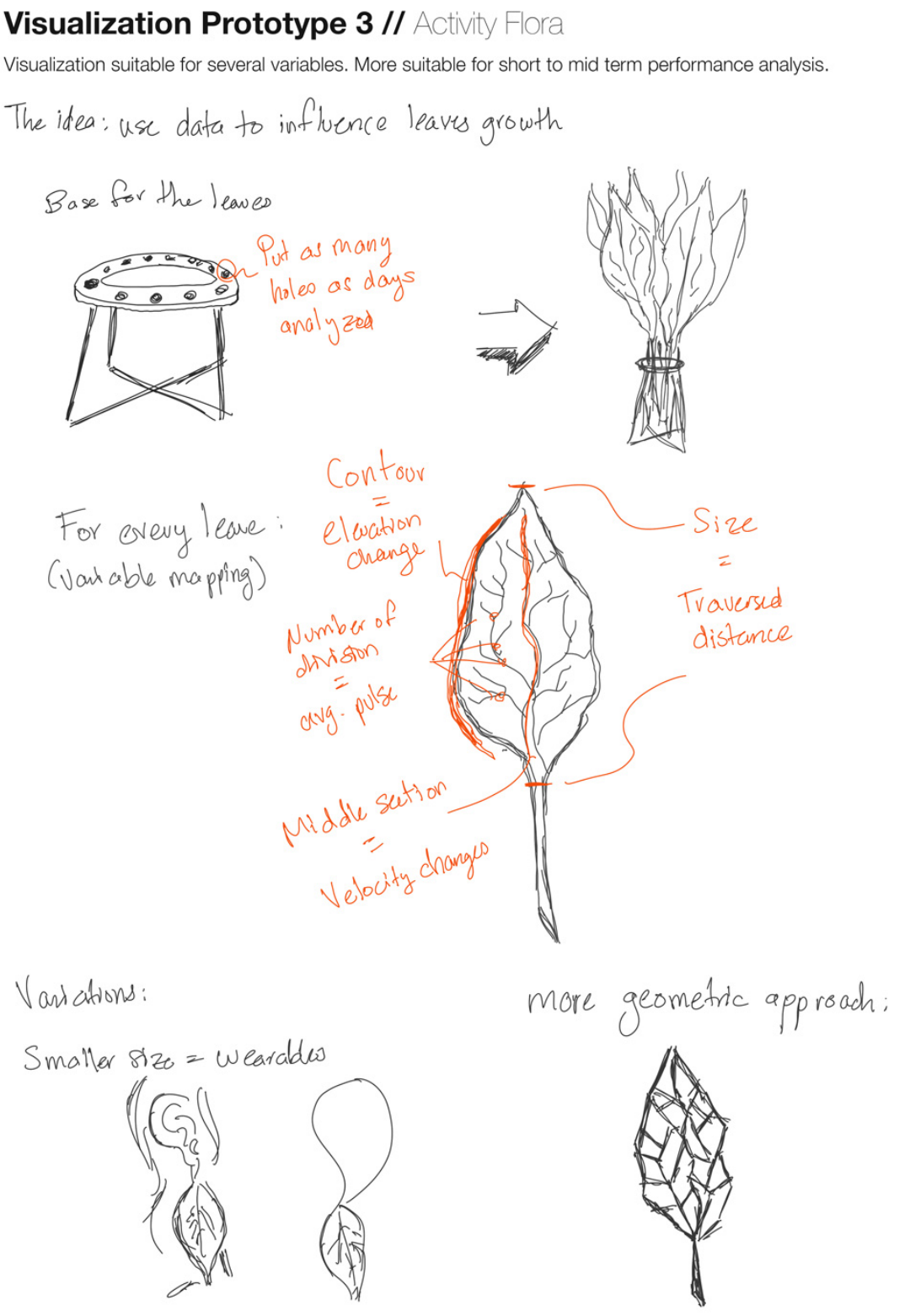




Variation for 3 mapping variables



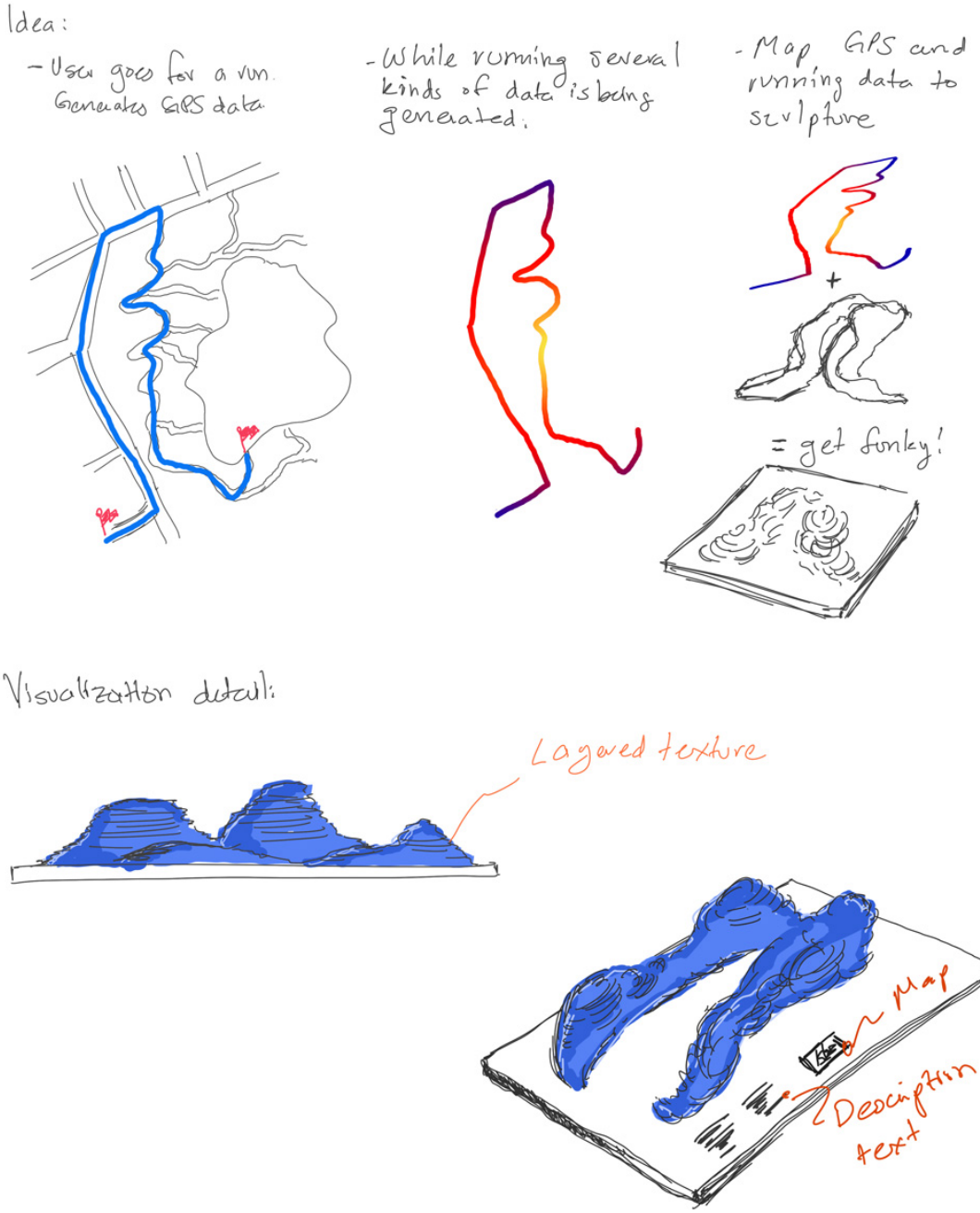
Activity Flora



Activity Landscape

Visualization Prototype 2 // Activity Landscape

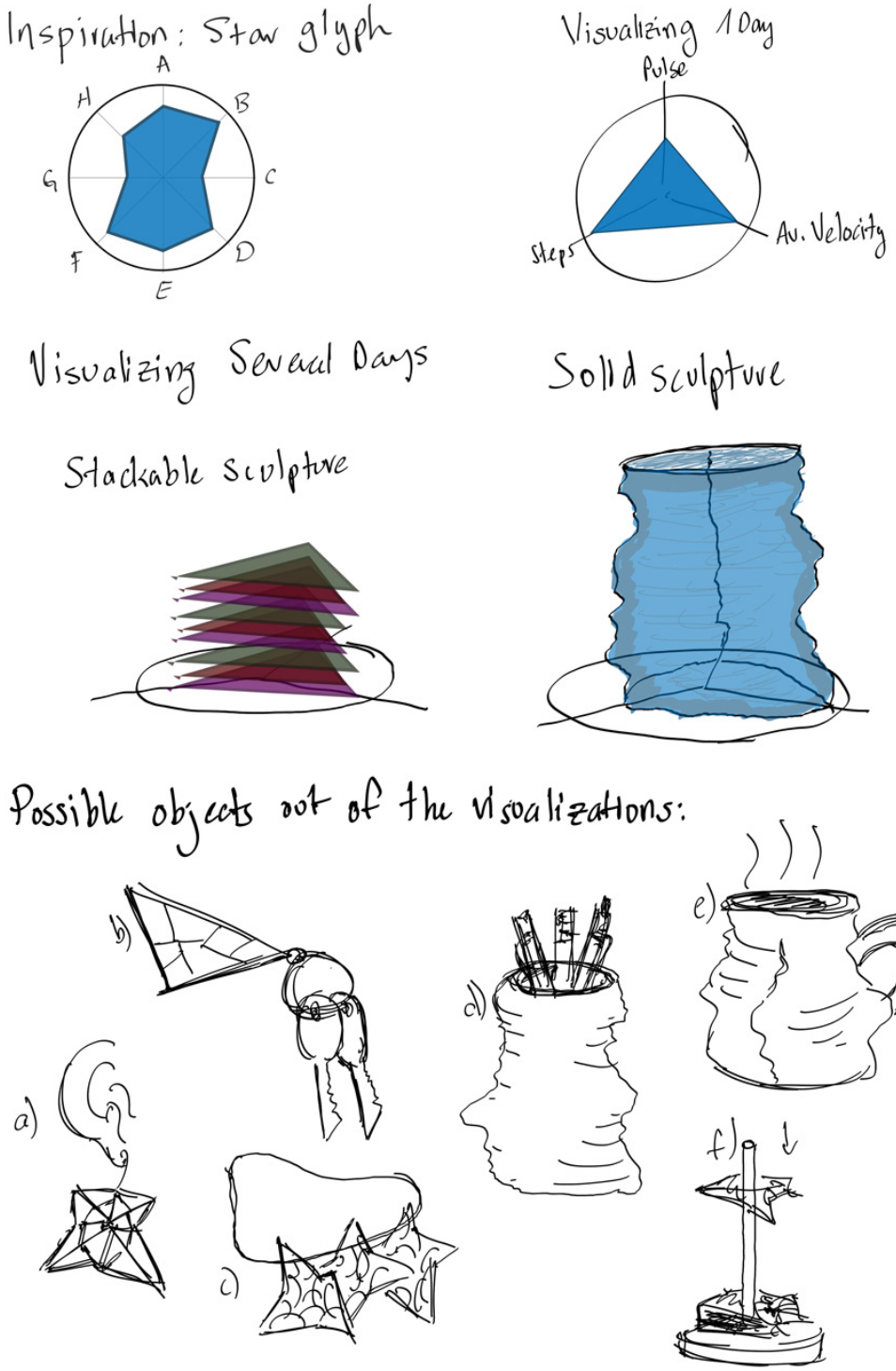
Visualization suitable for mapping two variables. More suitable for performance analysis using localization
- Variable 1: GPS Localization points.
- Variable 2: hear trate, speed, altitude.

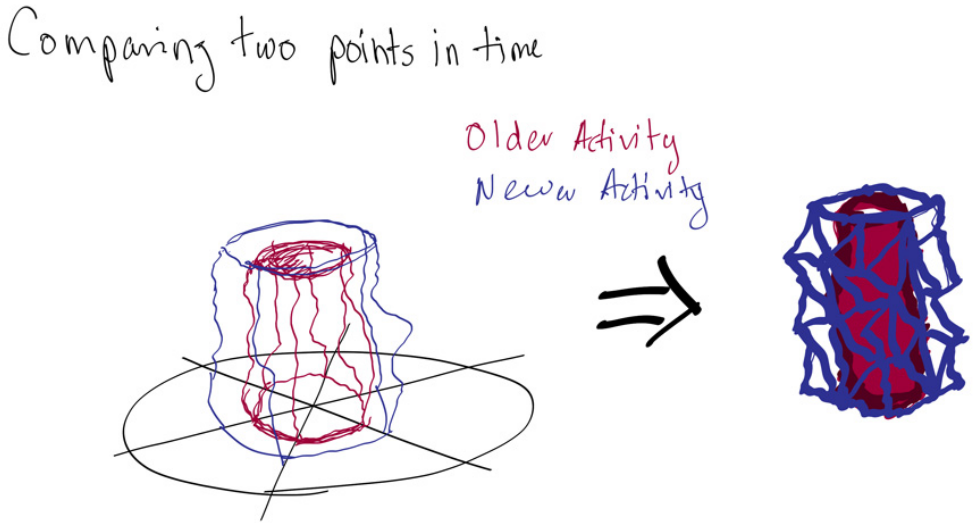


Activity Vase (continues in next page)

Visualization Prototype 4 // Activity Vase

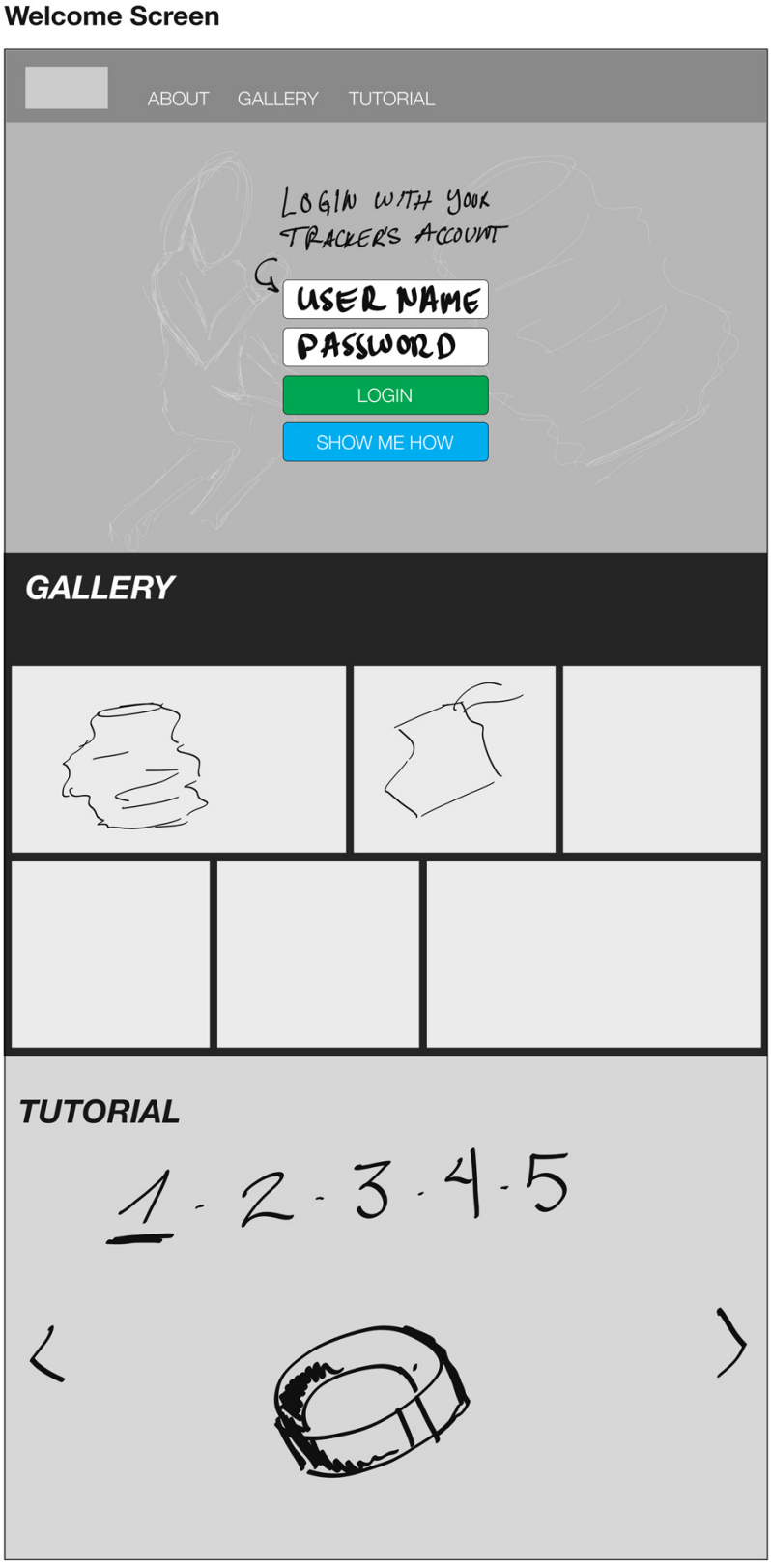
Visualization suitable for several variables. Suitable for visualizing activity data from one day or from longer time spans: a week or a month.



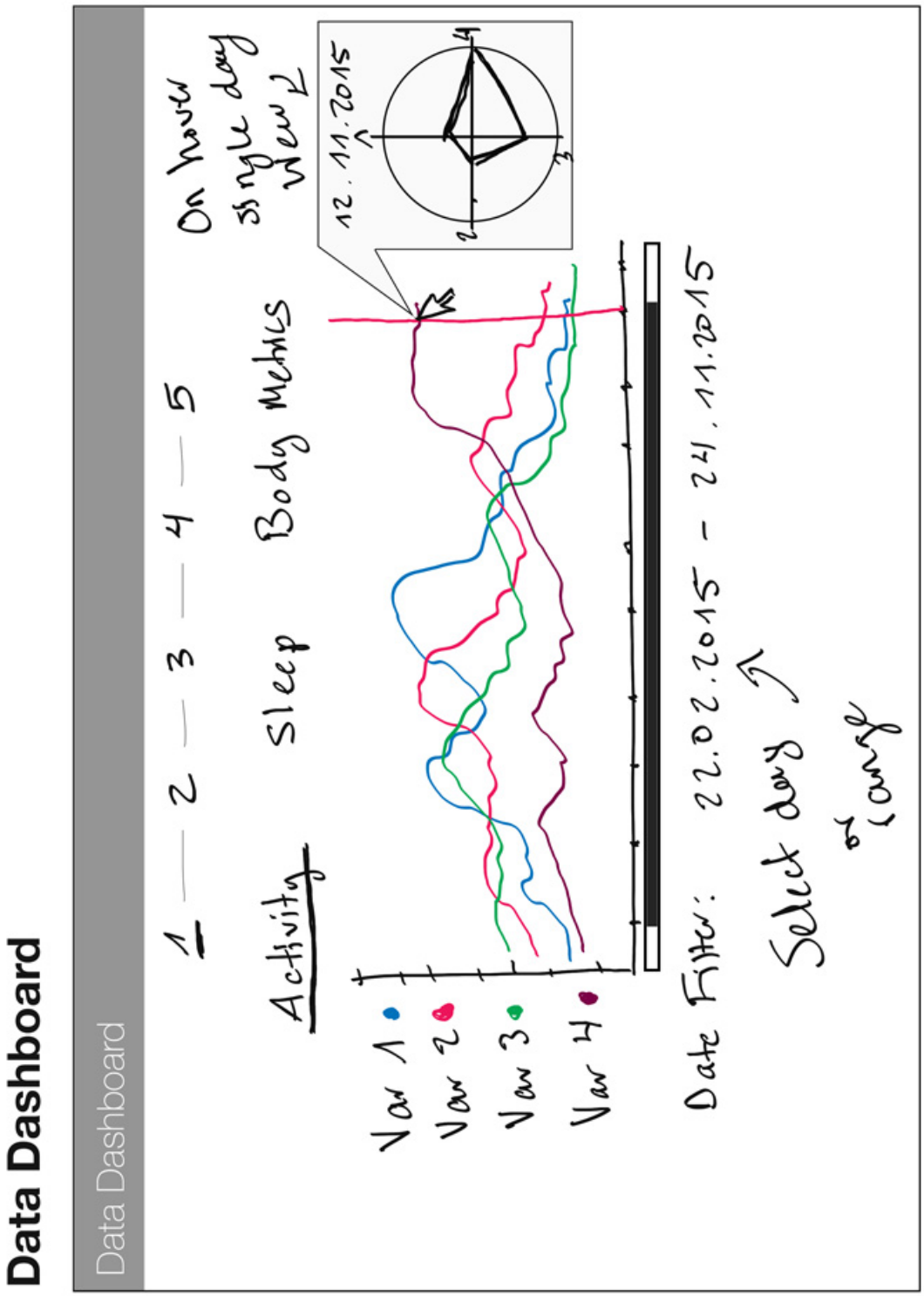

Comparing two points in time
Older Activity
Newer Activity

A.2 Web Configurator Prototypes

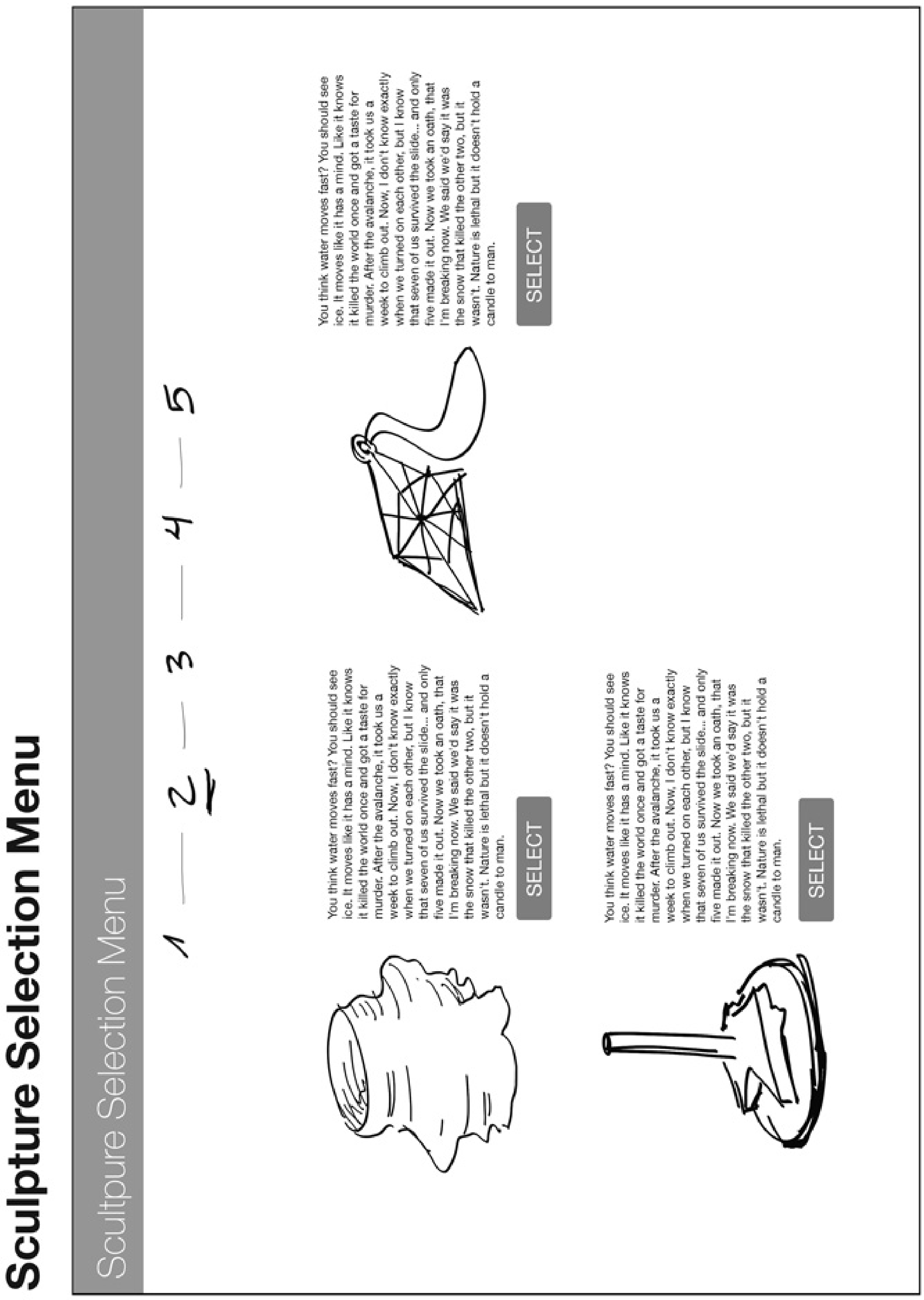
Configurator Protype No.1 Welcome Screen



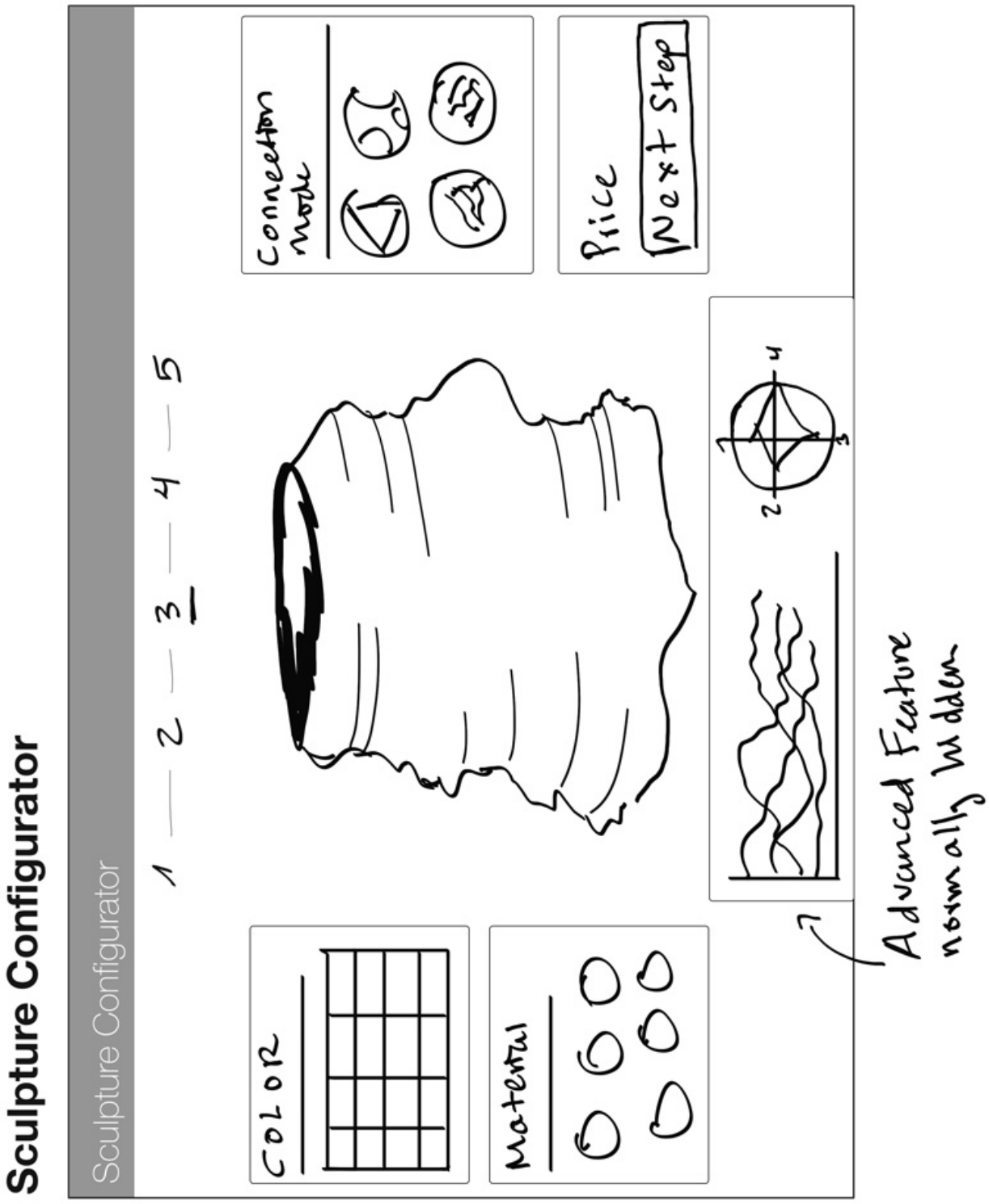
Configurator Protype No.1 Dashboard View



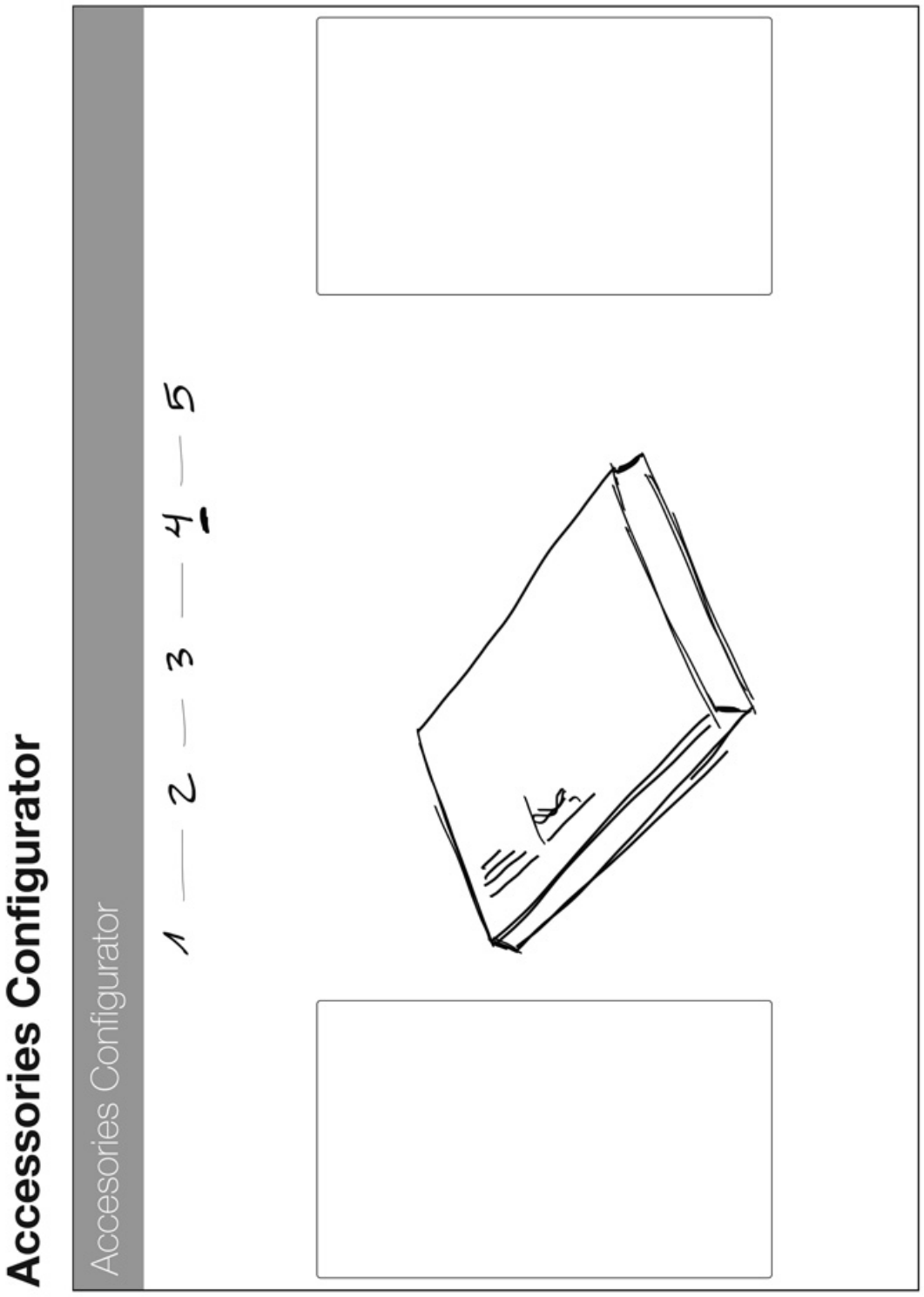
Configurator Protype No.1 Sculpture Selection View



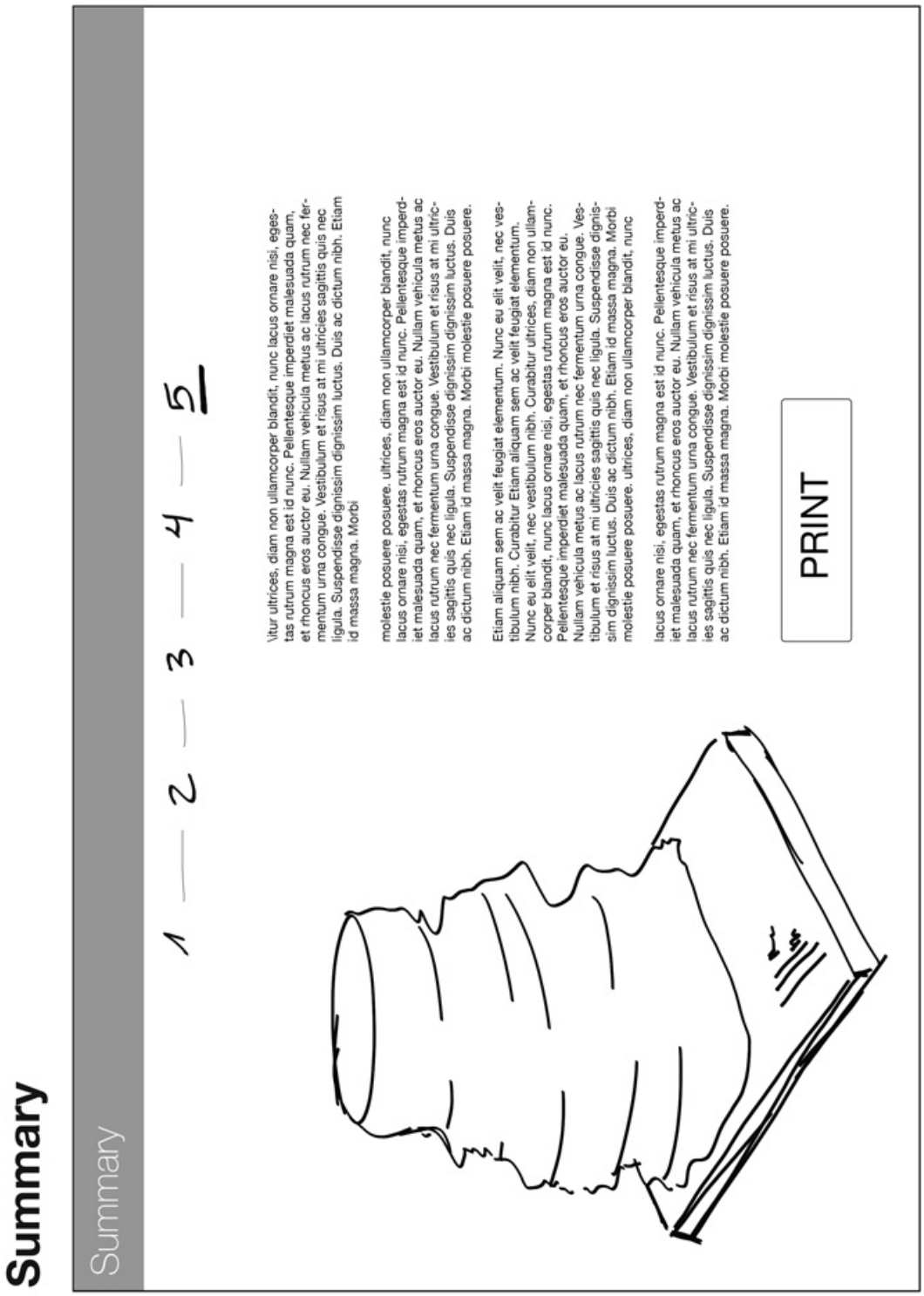
Configurator Protype No.1 Scultpure Configurator View



Configurator Protype No.1 Accessories View

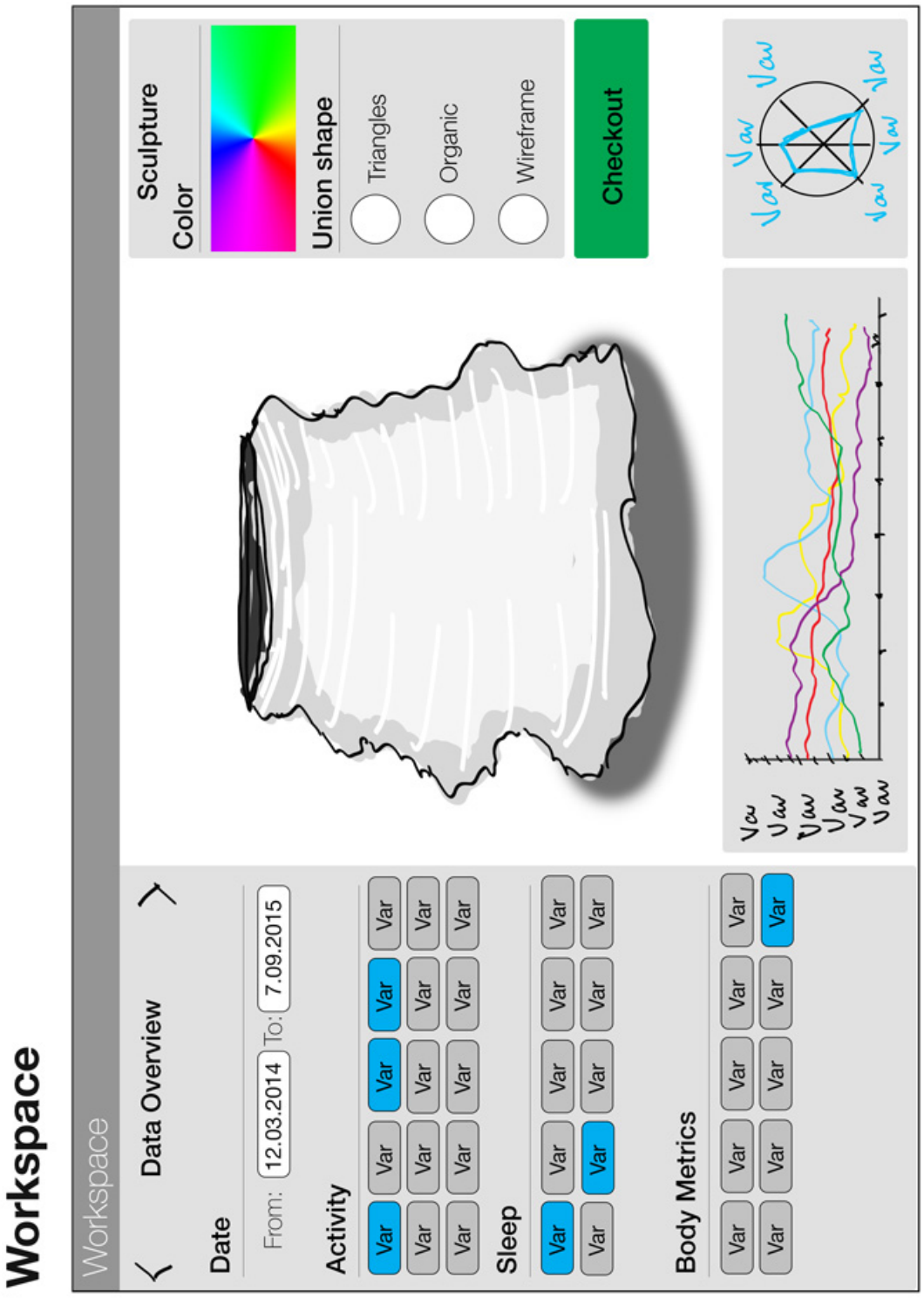


Configurator Protype No.1 Summary View

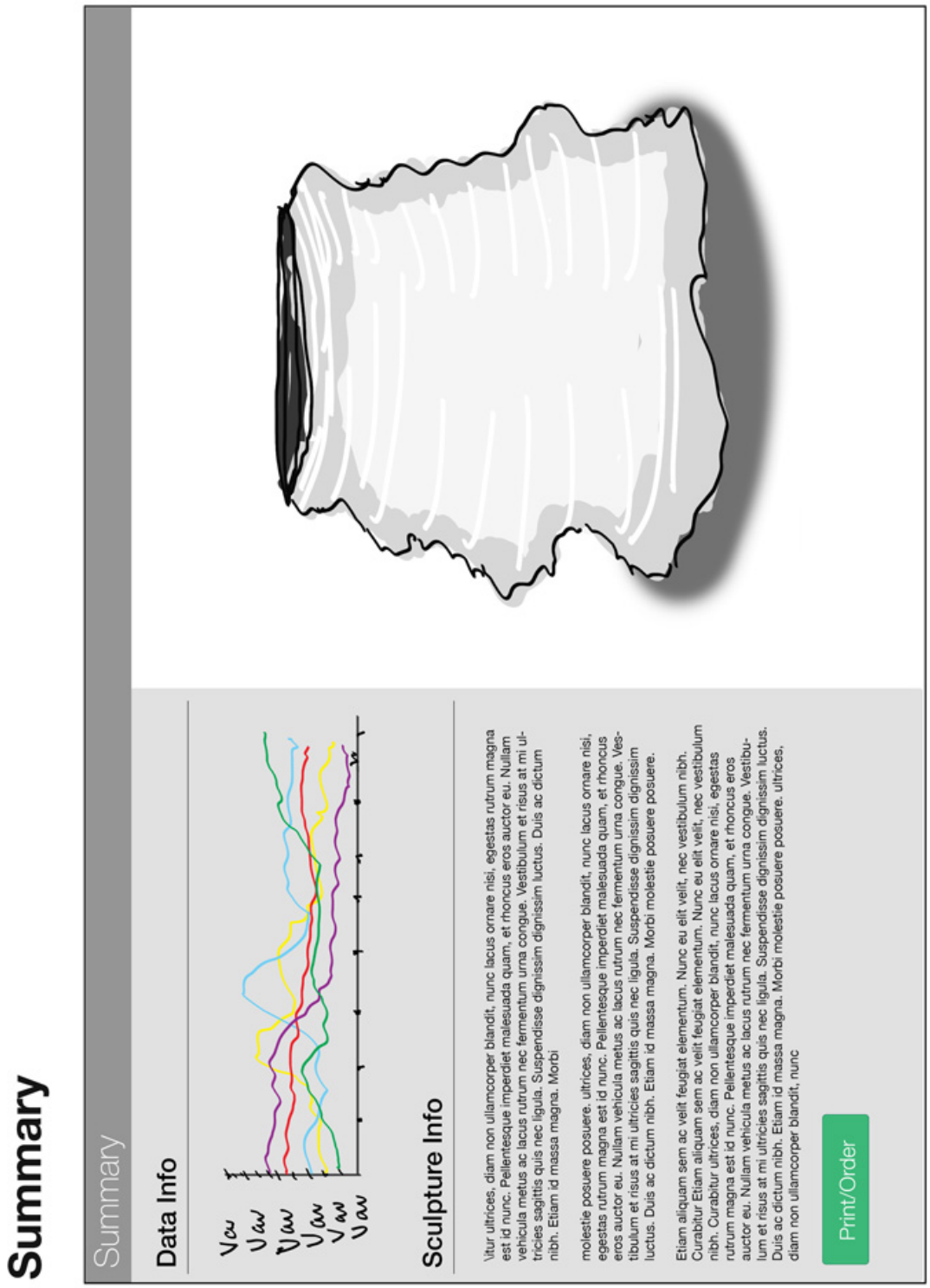


Summary

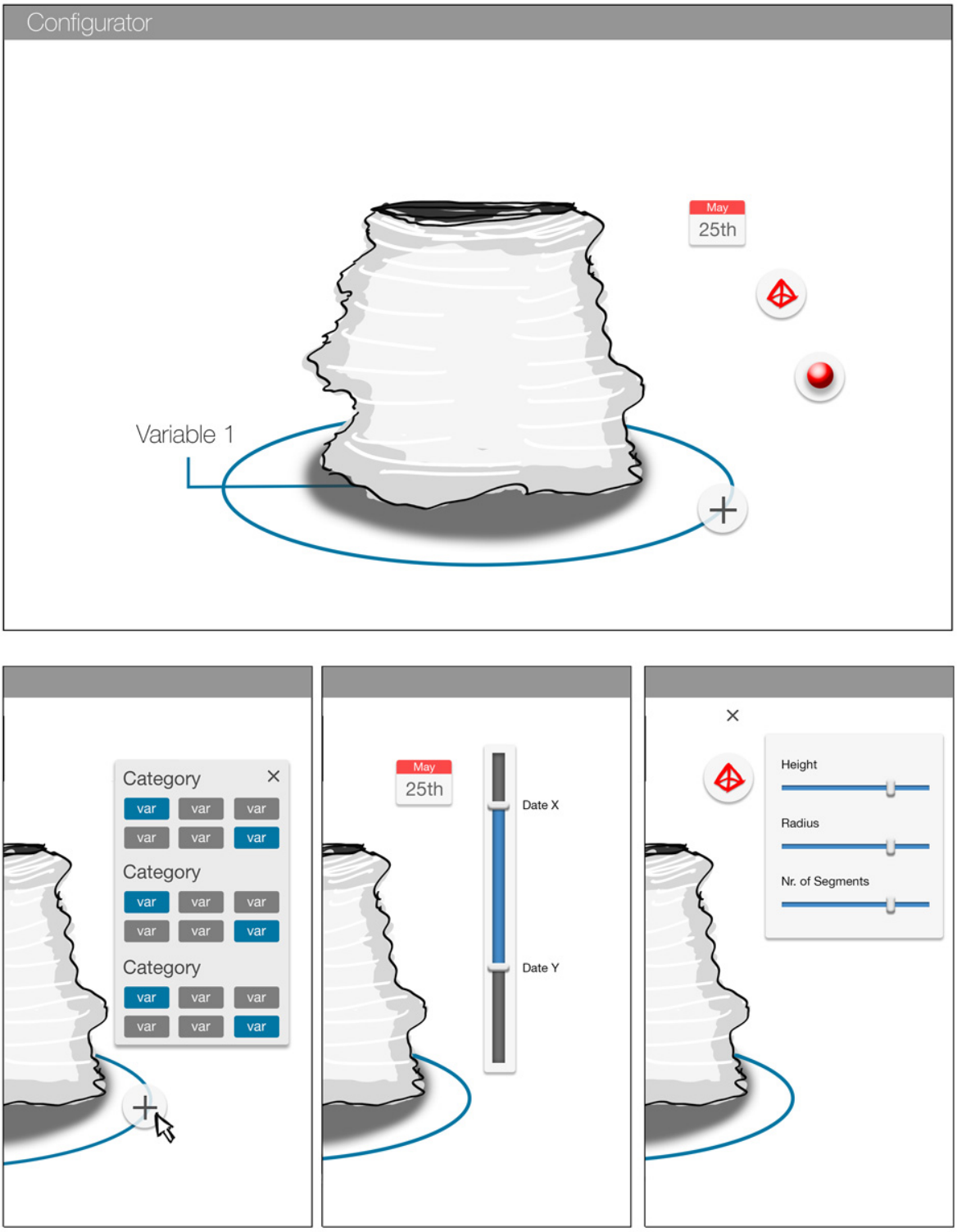
Configurator Protype No.2 Configurator View



Configurator Protype No.2 Summary View



Configurator Protype No.3



B Source Code

B.1 Source Code Repositories

Github: https://github.com/WalterRempening/Activity-Sculpture-Web-Configurator
Author's Gitlab Server: http://gitlab.walterrempening.com/root/web-configurator

B.2 Activity Vase Sculpture Generation Code

Listing B.1: WCVaseGeometry.js

```
/*****************************************************************
* Activity Vase Sculpture
* author: Walter Rempening Diaz
* Threejs geometry object for the generation of
* the activity vase sculpture
*****************************************************************/

var WCVaseGeometry = function ( data, outerRadius, innerRadius, height,
                                radialSegments,
                                heightSegments, definition, interpolate
                                    ) {

  THREE.Geometry.call( this );

  this.parameters = {
    data: data,
    outerRadius: outerRadius,
    innerRadius: innerRadius,
    height: height,
    radialSegemnts: radialSegments,
    heightSegments: heightSegments,
    definition: definition,
    interpolation: interpolate
  };


  interpolate = interpolate !== undefined ? interpolate : false;
  definition = definition !== undefined ? definition : 50;
  outerRadius = outerRadius !== undefined ? outerRadius : 20;
  innerRadius = innerRadius !== undefined ? innerRadius : 15;
  height = height !== undefined ? height : 100;

  radialSegments = radialSegments || 8;
  heightSegments = heightSegments || 1;

  var heightHalf = height / 2;
  var x, y, vertices = [], uvs = [];
  var tanTheta = (outerRadius) / height;
  var radialIncrement, segmentIncrement;

  if ( !interpolate ) {
    segmentIncrement = radialSegments === 1 || radialSegments === 2 ?
        definition : 0;
  } else {
    segmentIncrement = definition - 1;
  }
  for ( y = 0; y <= heightSegments; y++ ) {
    var verticesRow = [];
    var uvsRow = [];
    var v = y / heightSegments;
```

```
49
50      if ( !interpolate ) {
51
52        for ( x = 0; x <= radialSegments + segmentIncrement; x++ ) {
53          var radius, percent;
54          if ( radialSegments === 1 ) {
55            radialIncrement = 0;
56            radius = data[ radialIncrement ][ y ] + outerRadius;
57          } else if ( radialSegments === 2 ) {
58            if ( x < (radialSegments + segmentIncrement) / 2 ) {
59              radialIncrement = 0;
60            } else {
61              radialIncrement = 1;
62            }
63            radius = data[ radialIncrement ][ y ] + outerRadius;
64          } else {
65            radialIncrement = x;
66            radius = data[ radialIncrement ][ y ] + outerRadius;
67          }
68
69          var u = x / (radialSegments + segmentIncrement);
70          var vertex = new THREE.Vector3();
71          vertex.x = radius * Math.sin( u * 2 * Math.PI ); // Math.PI is
                for thetaLength
72          vertex.y = -v * height + heightHalf;
73          vertex.z = radius * Math.cos( u * 2 * Math.PI );
74
75          this.vertices.push( vertex );
76
77          verticesRow.push( this.vertices.length - 1 );
78          uvsRow.push( new THREE.Vector2( u, 1 - v ) );
79        }
80      } else {
81        // number of steps between each segment
82        var step = Math.floor( (radialSegments + segmentIncrement) /
              radialSegments );
83
84        for ( var k = 0; k < radialSegments; k++ ) {
85          for ( x = 0; x <= step; x++ ) {
86
87            var radius, percent;
88            if ( radialSegments === 1 ) {
89              radialIncrement = 0;
90              radius = data[ radialIncrement ][ y ] + outerRadius;
91              var u = x / (radialSegments + segmentIncrement);
92            } else {
93              percent = x / step;
94              var start = data[ k ][ y ];
95              var end = data[ k + 1 ][ y ] !== undefined ? data[ k + 1 ][
                    y ] : data[ k ][ y ];
96              var rlerp = lerp( start, end, percent );
97              radius = rlerp + outerRadius;
98              u = (x / (radialSegments + segmentIncrement)) + (k /
                  radialSegments );
99            }
100
101
102           var vertex = new THREE.Vector3();
103           vertex.x = radius * Math.sin( u * 2 * Math.PI ); // Math.PI
                is for thetaLength
104           vertex.y = -v * height + heightHalf;
105           vertex.z = radius * Math.cos( u * 2 * Math.PI );
```

```
106
107             this.vertices.push( vertex );
108
109             verticesRow.push( this.vertices.length - 1 );
110             uvsRow.push( new THREE.Vector2( u, 1 - v ) );
111           }
112         }
113       }
114       vertices.push( verticesRow );
115       uvs.push( uvsRow );
116     }
117
118     for ( x = 0; x < radialSegments + segmentIncrement - 1; x++ ) {
119       var na, nb;
120       na = this.vertices[ vertices[ 1 ][ x ] ].clone();
121       nb = this.vertices[ vertices[ 1 ][ x + 1 ] ].clone();
122
123       na.setY( Math.sqrt( na.x * na.x + na.z * na.z ) * tanTheta ).
            normalize();
124       nb.setY( Math.sqrt( nb.x * nb.x + nb.z * nb.z ) * tanTheta ).
            normalize();
125
126       for ( y = 0; y < heightSegments; y++ ) {
127
128         var v1 = vertices[ y ][ x ];
129         var v2 = vertices[ y + 1 ][ x ];
130         var v3 = vertices[ y + 1 ][ x + 1 ];
131         var v4 = vertices[ y ][ x + 1 ];
132
133         var n1 = na.clone();
134         var n2 = na.clone();
135         var n3 = nb.clone();
136         var n4 = nb.clone();
137
138         var uv1 = uvs[ y ][ x ].clone();
139         var uv2 = uvs[ y + 1 ][ x ].clone();
140         var uv3 = uvs[ y + 1 ][ x + 1 ].clone();
141         var uv4 = uvs[ y ][ x + 1 ].clone();
142
143         this.faces.push( new THREE.Face3( v1, v2, v4, [ n1, n2, n4 ] ) );
144         this.faceVertexUvs[ 0 ].push( [ uv1, uv2, uv4 ] );
145
146         this.faces.push( new THREE.Face3( v2, v3, v4,
147           [ n2.clone(), n3, n4.clone() ] ) );
148         this.faceVertexUvs[ 0 ].push( [ uv2.clone(), uv3, uv4.clone() ] )
              ;
149       }
150     }
151
152     // side gap connections
153     for ( var h = 0; h < heightSegments; h++ ) {
154       var na, nb;
155
156       na = this.vertices[ vertices[ h ][ 0 ] ].clone();
157       nb = this.vertices[ vertices[ h + 1 ][ 0 ] ].clone();
158
159       na.setY( Math.sqrt( na.x * na.x + na.z * na.z ) * tanTheta ).
            normalize();
160       nb.setY( Math.sqrt( nb.x * nb.x + nb.z * nb.z ) * tanTheta ).
            normalize();
161
162       var v1 = vertices[ h ][ radialSegments + segmentIncrement - 1 ];
```

```
163       var v2 = vertices[ h + 1 ][ radialSegments + segmentIncrement - 1
              ];
164       var v3 = vertices[ h + 1 ][ 0 ];
165       var v4 = vertices[ h ][ 0 ];
166
167       var n1 = na.clone();
168       var n2 = na.clone();
169       var n3 = nb.clone();
170       var n4 = nb.clone();
171
172       var uv1 = uvs[ h ][ radialSegments + segmentIncrement - 1 ].clone()
              ;
173       var uv2 = uvs[ h + 1 ][ radialSegments + segmentIncrement - 1 ].
              clone();
174       var uv3 = uvs[ h + 1 ][ 0 ].clone();
175       var uv4 = uvs[ h ][ 0 ].clone();
176
177       this.faces.push( new THREE.Face3( v1, v2, v4, [ n1, n2, n4 ] ) );
178       this.faceVertexUvs[ 0 ].push( [ uv1, uv2, uv4 ] );
179
180       this.faces.push( new THREE.Face3( v2, v3, v4,
181         [ n2.clone(), n3, n4.clone() ] ) );
182       this.faceVertexUvs[ 0 ].push( [ uv2.clone(), uv3, uv4.clone() ] );
183     }
184     // top cap
185     this.vertices.push( new THREE.Vector3( 0, heightHalf, 0 ) );
186
187     for ( x = 0; x < radialSegments + segmentIncrement; x++ ) {
188       var n1 = new THREE.Vector3( 0, 1, 0 );
189       var n2 = new THREE.Vector3( 0, 1, 0 );
190       var n3 = new THREE.Vector3( 0, 1, 0 );
191
192       if ( x !== radialSegments + segmentIncrement - 1 ) {
193         var v1 = vertices[ 0 ][ x ];
194         var v2 = vertices[ 0 ][ x + 1 ];
195         var v3 = this.vertices.length - 1;
196
197         var uv1 = uvs[ 0 ][ x ].clone();
198         var uv2 = uvs[ 0 ][ x + 1 ].clone();
199         var uv3 = new THREE.Vector2( uv2.x, 0 );
200       } else {
201         var v1 = vertices[ 0 ][ radialSegments + segmentIncrement - 1 ];
202         var v2 = vertices[ 0 ][ 0 ];
203         var v3 = this.vertices.length - 1;
204
205         var uv1 = uvs[ 0 ][ radialSegments + segmentIncrement - 1 ].clone
              ();
206         var uv2 = uvs[ 0 ][ 0 ].clone();
207         var uv3 = new THREE.Vector2( uv2.x, 0 );
208       }
209
210       this.faces.push( new THREE.Face3( v1, v2, v3, [ n1, n2, n3 ] ) );
211       this.faceVertexUvs[ 0 ].push( [ uv1, uv2, uv3 ] );
212
213     }
214
215     // bottom cap
216     this.vertices.push( new THREE.Vector3( 0, -heightHalf, 0 ) );
217
218     for ( x = 0; x < radialSegments + segmentIncrement; x++ ) {
219
220       var n1 = new THREE.Vector3( 0, -1, 0 );
```

```
221       var n2 = new THREE.Vector3( 0, -1, 0 );
222       var n3 = new THREE.Vector3( 0, -1, 0 );
223
224       if ( x !== radialSegments + segmentIncrement - 1 ) {
225         var v1 = vertices[ heightSegments ][ x + 1 ];
226         var v2 = vertices[ heightSegments ][ x ];
227         var v3 = this.vertices.length - 1;
228
229         var uv1 = uvs[ heightSegments ][ x + 1 ].clone();
230         var uv2 = uvs[ heightSegments ][ x ].clone();
231         var uv3 = new THREE.Vector2( uv2.x, 1 );
232       } else {
233         var uv1 = uvs[ heightSegments ][ 0 ].clone();
234         var uv2 = uvs[ heightSegments ][ radialSegments +
              segmentIncrement - 1 ].clone();
235         var uv3 = new THREE.Vector2( uv2.x, 1 );
236
237         var v1 = vertices[ heightSegments ][ 0 ];
238         var v2 = vertices[ heightSegments ][ radialSegments +
              segmentIncrement - 1 ];
239         var v3 = this.vertices.length - 1;
240       }
241
242       this.faces.push( new THREE.Face3( v1, v2, v3, [ n1, n2, n3 ] ) );
243       this.faceVertexUvs[ 0 ].push( [ uv1, uv2, uv3 ] );
244     }
245
246     this.computeFaceNormals();
247
248     function lerp ( start, end, percent ) {
249       return (start + percent * (end - start));
250     }
251   }
252
253   WCVaseGeometry.prototype = Object.create( THREE.Geometry.prototype );
254   WCVaseGeometry.prototype.constructor = WCVaseGeometry;
```

C User Study & Questionnaire

C.1 Questionnaire

Activity Sculpture Web-Configurator

This questionnaire helps us to better understand your experience while using the configurator.

* Erforderlich

1. **Gender** *
   Markieren Sie nur ein Oval.
   - ◯ Male
   - ◯ Female

2. **Age** *
   .....

Web Configurators

3. **Describe the type of experience you have with 3D modelling software** *
   Markieren Sie nur ein Oval.
   - ◯ None
   - ◯ Low
   - ◯ Intermediate
   - ◯ Advanced
   - ◯ Pro

4. **Have you ever used a web configurator before?** *
   Markieren Sie nur ein Oval.
   - ◯ Yes *Weiter mit Frage 5*
   - ◯ No *Weiter mit Frage 6*

Weiter mit Frage 6

Configurators in E-shops

5. **What describes best your experience while using web configurators?** *
   Markieren Sie nur ein Oval.
   - ◯ Frustrating
   - ◯ Complicated
   - ◯ Neutral
   - ◯ Interesting
   - ◯ Enjoyable

Product Value

6. **Do you think customising your product adds more value to it?**
Markieren Sie nur ein Oval.

- ◯ Yes
- ◯ No

7. **Do you think products that can be customised tend to be more expensive?** *
Markieren Sie nur ein Oval.

- ◯ Yes
- ◯ No

Tracking Devices

8. **Are you mindful/aware of your daily activity, i.e. have exercise goals like the recommended 10000 daily steps** *
Markieren Sie nur ein Oval.

- ◯ I've never thought about it
- ◯ From time to time, and try to do some excersie
- ◯ Yes, I exercise a few times a week
- ◯ It is really important to me, I exercise daily

9. **Do you use a tracking device?** *
Markieren Sie nur ein Oval.

- ◯ Yes *Weiter mit Frage 10*
- ◯ No *Weiter mit Frage 12*

10. **What tracking device do you use?** *
Markieren Sie nur ein Oval.

- ◯ Jawbone Up-2/3/4
- ◯ Fitbit Flex/Cahrge
- ◯ Withings Pulse X/Activité
- ◯ Polar A3000
- ◯ Sonstiges: .....

11. **Has the tracking device influenced your activity** *
Markieren Sie nur ein Oval.

- ◯ Yes
- ◯ No

Weiter mit Frage 13

12. **Have you considered using a tracking device** *
Markieren Sie nur ein Oval.

- ◯ Yes
- ◯ No

Activity Sculptures

13. **Which style did you find the most attractive?** *
Markieren Sie nur ein Oval.

- ◯ Normal
- ◯ Interpolated
- ◯ Wire-frame
- ◯ Wire-frame + Interpolated

14. **The sculpture was aesthetically appealing** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly desagree

15. **I felt attached to the sculpture in some way** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

16. **It was interesting to see my activity data visualized in a sculpture** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

17. **I would use it regularly to visualize my data** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

18. **If available, I would order a 3D printed sculpture of my activity** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

19. **I would wear my sculpture if it could be adapted to earrings, necklaces, rings, smart-phone cases, key chains,** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

20. **I would show my sculpture to friends and relatives**
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

Configurator Usability

21. **The configurator's interface was aesthetically appealing** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

22. **The configurator was easy to use** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

23. **The configurator was easy to learn** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

24. **I found it helpful to see a summary of all my activity in the dashboard** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly desagree

25. **A gallery of my sculptures is helpful as it allows me to collect my sculptures over time** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

26. **Each label clearly described the functionality of the slider or toggle control** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

27. **It was always clear how each slider and toggle button manipulated the sculpture** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

28. **The available configuration options were enough for me** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

29. **It was helpful to see the sculpture update instantly after changing a slider or toggle value** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

30. **I would have liked to see a visualization that represented the data more acurately** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

31. **I would have liked to have different sculptures to choose from** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

Final Remarks

32. **I find the configurator's functionality useful** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

33. **I would use a configurator like this again** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

34. **I am pleased with the sculpture I configured** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
I strongly agree	◯	◯	◯	◯	◯	I strongly disagree

35. **Any final thoughts you want to share?**

.....

.....

.....

.....

.....

C.2 Questionnaire Results

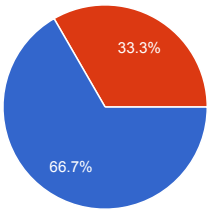
w.rempeningd@gmail.com
Dieses Formular bearbeiten

9 Antworten

Alle Antworten ansehen Analytics veröffentlichen

Zusammenfassung

Gender



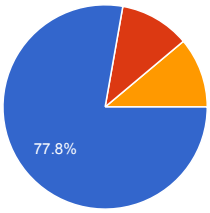
Male	**6**	66.7 %
Female	**3**	33.3 %

Age

35
22
23
24
26
28
19

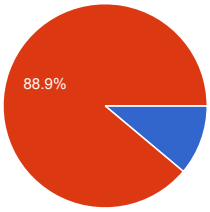
Web Configurators

Describe the type of experience you have with 3D modelling software



None	7	77.8 %
Low	1	11.1 %
Intermediate	1	11.1 %
Advanced	0	0 %
Pro	0	0 %

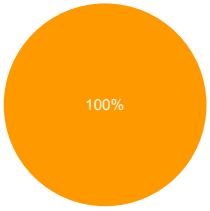
Have you ever used a web configurator before?



Yes	1	11.1 %
No	8	88.9 %

Configurators in E-shops

What describes best your experience while using web configurators?



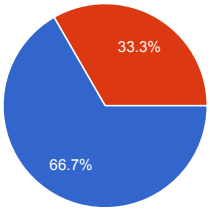
Frustrating	0	0 %
Complicated	0	0 %
Neutral	1	100 %
Interesting	0	0 %
Enjoyable	0	0 %

Product Value

Do you think customising your product adds more value to it?

Yes	**9**	100 %
No	**0**	0 %

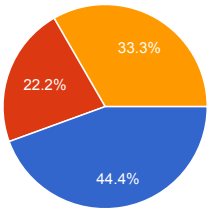
Do you think products that can be customised tend to be more expensive?



Yes	**6**	66.7 %
No	**3**	33.3 %

Tracking Devices

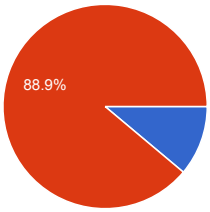
Are you mindful/aware of your daily activity, i.e. have exercise goals like the recommended 10000 daily steps



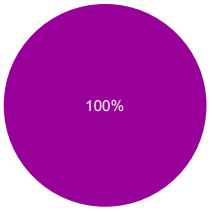
I've never thought about it	**4**	44.4 %
From time to time, and try to do some excersie	**2**	22.2 %
Yes, I exercise a few times a week	**3**	33.3 %
It is really important to me, I exercise daily	**0**	0 %

Do you use a tracking device?

Yes	**1**	11.1 %
No	**8**	88.9 %



What tracking device do you use?



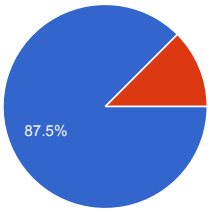
Jawbone Up-2/3/4	**0**	0 %
Fitbit Flex/Cahrge	**0**	0 %
Withings Pulse X/Activité	**0**	0 %
Polar A3000	**0**	0 %
Sonstige	**1**	100 %

Has the tracking device influenced your activity

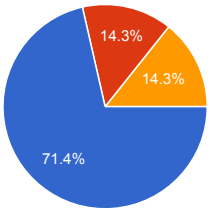
Yes	**1**	100 %
No	**0**	0 %

Have you considered using a tracking device

Yes	**7**	87.5 %
No	**1**	12.5 %

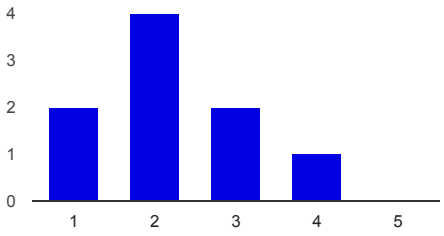


Which style did you find the most attractive?



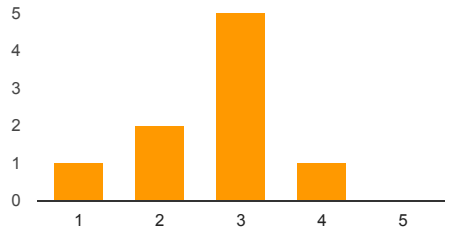
Normal	**5**	55.6 %
Interpolated	**1**	11.1 %
Wire-frame	**1**	11.1 %
Wire-frame + Interpolated	**0**	0 %

The sculpture was aesthetically appealing



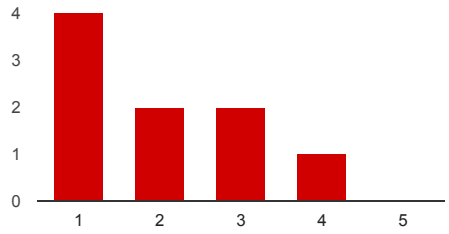
I strongly agree: 1	**2**	22.2 %
2	**4**	44.4 %
3	**2**	22.2 %
4	**1**	11.1 %
I strongly desagree: 5	**0**	0 %

I felt attached to the sculpture in some way



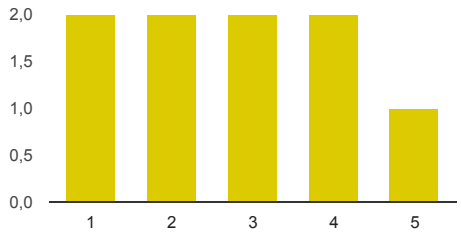
I strongly agree:	1	**1**	11.1 %
	2	**2**	22.2 %
	3	**5**	55.6 %
	4	**1**	11.1 %
I strongly disagree:	5	**0**	0 %

It was interesting to see my activity data visualized in a sculpture



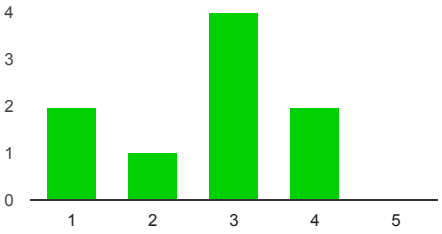
I strongly agree:	1	**4**	44.4 %
	2	**2**	22.2 %
	3	**2**	22.2 %
	4	**1**	11.1 %
I strongly disagree:	5	**0**	0 %

I would use it regularly to visualize my data



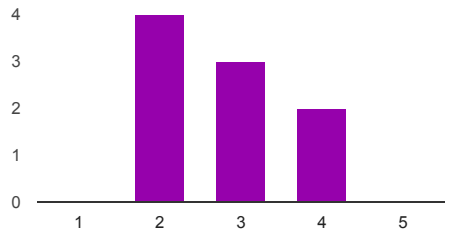
2	**2**	22.2 %
3	**2**	22.2 %
4	**2**	22.2 %
I strongly disagree: 5	**1**	11.1 %

If available, I would order a 3D printed sculpture of my activity



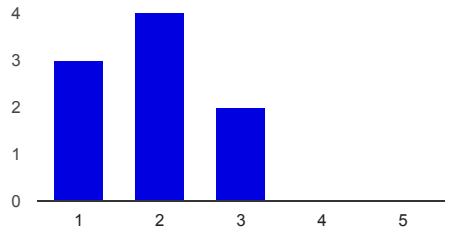
I strongly agree: 1	**2**	22.2 %
2	**1**	11.1 %
3	**4**	44.4 %
4	**2**	22.2 %
I strongly disagree: 5	**0**	0 %

I would wear my sculpture if it could be adapted to earrings, necklaces, rings, smart-phone cases, key chains,



I strongly disagree: 5 **0** 0 %

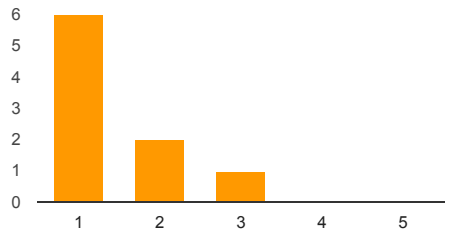
I would show my sculpture to friends and relatives



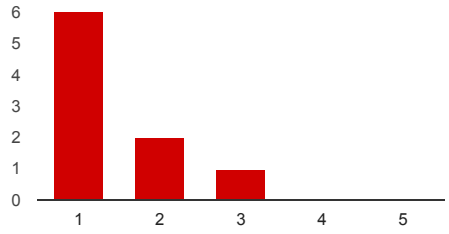
I strongly agree: 1	**3**	33.3 %
2	**4**	44.4 %
3	**2**	22.2 %
4	**0**	0 %
I strongly disagree: 5	**0**	0 %

Configurator Usability

The configurator's interface was aesthetically appealing

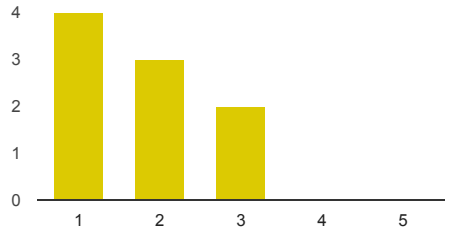


The configurator was easy to use



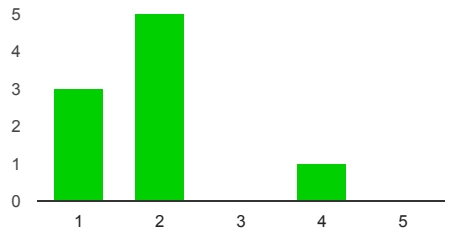
I strongly agree: 1	**6**	66.7 %
2	**2**	22.2 %
3	**1**	11.1 %
4	**0**	0 %
I strongly disagree: 5	**0**	0 %

The configurator was easy to learn



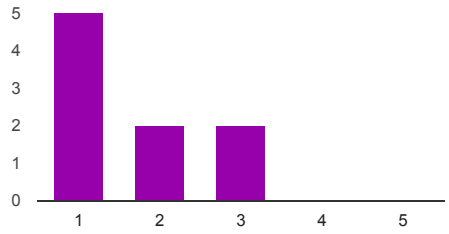
I strongly agree: 1	**4**	44.4 %
2	**3**	33.3 %
3	**2**	22.2 %
4	**0**	0 %
I strongly disagree: 5	**0**	0 %

I found it helpful to see a summary of all my activity in the dashboard



I strongly agree: 1	**3**	33.3 %
2	**5**	55.6 %
3	**0**	0 %
4	**1**	11.1 %
I strongly desagree: 5	**0**	0 %

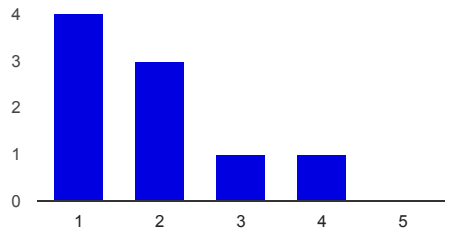
A gallery of my sculptures is helpful as it allows me to collect my sculptures over time



I strongly agree: 1	**5**	55.6 %
2	**2**	22.2 %
3	**2**	22.2 %

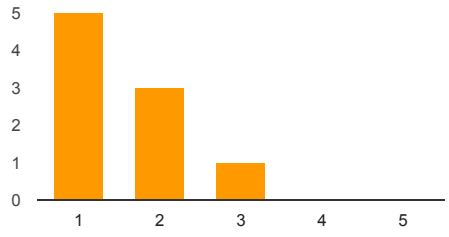
4	**0**	0 %
I strongly disagree: 5	**0**	0 %

Each label clearly described the functionality of the slider or toggle control



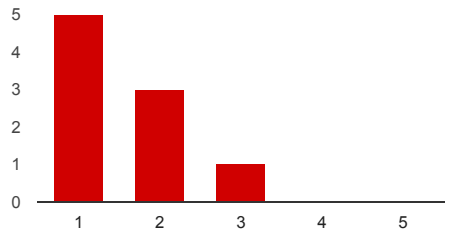
I strongly agree: 1	**4**	44.4 %
2	**3**	33.3 %
3	**1**	11.1 %
4	**1**	11.1 %
I strongly disagree: 5	**0**	0 %

It was always clear how each slider and toggle button manipulated the sculpture



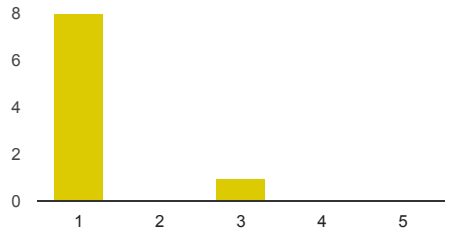
I strongly agree: 1	**5**	55.6 %
2	**3**	33.3 %
3	**1**	11.1 %
4	**0**	0 %
I strongly disagree: 5	**0**	0 %

The available configuration options were enough for me



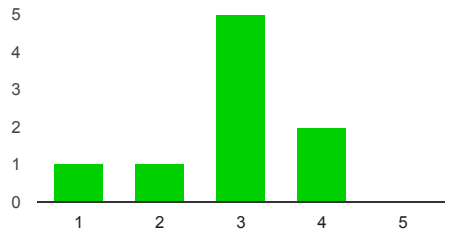
I strongly agree:	1	**5**	55.6 %
	2	**3**	33.3 %
	3	**1**	11.1 %
	4	**0**	0 %
I strongly disagree:	5	**0**	0 %

It was helpful to see the sculpture update instantly after changing a slider or toggle value



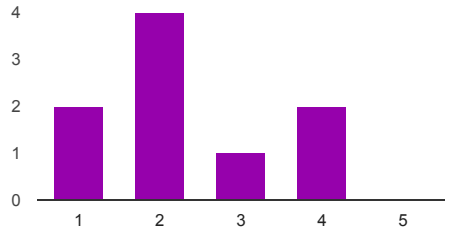
I strongly agree:	1	**8**	88.9 %
	2	**0**	0 %
	3	**1**	11.1 %
	4	**0**	0 %
I strongly disagree:	5	**0**	0 %

I would have liked to see a visualization that represented the data more acurately



I strongly agree: 1	**1**	11.1 %
2	**1**	11.1 %
3	**5**	55.6 %
4	**2**	22.2 %
I strongly disagree: 5	**0**	0 %

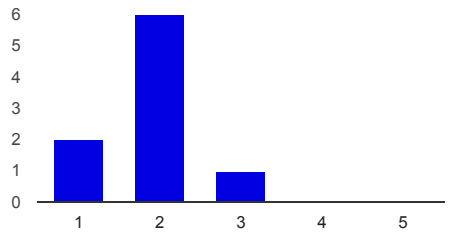
I would have liked to have different sculptures to choose from



I strongly agree: 1	**2**	22.2 %
2	**4**	44.4 %
3	**1**	11.1 %
4	**2**	22.2 %
I strongly disagree: 5	**0**	0 %

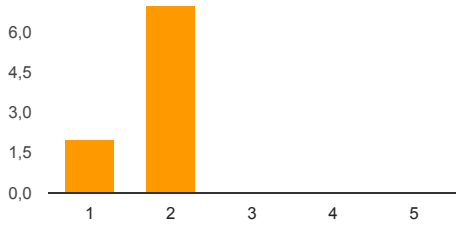
Final Remarks

I find the configurator's functionality useful



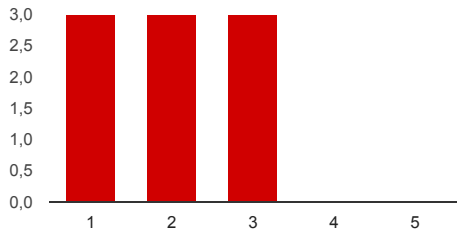
I strongly agree: 1	**2**	22.2 %
2	**6**	66.7 %
3	**1**	11.1 %
4	**0**	0 %
I strongly disagree: 5	**0**	0 %

I would use a configurator like this again



I strongly agree: 1	**2**	22.2 %
2	**7**	77.8 %
3	**0**	0 %
4	**0**	0 %
I strongly disagree: 5	**0**	0 %

I am pleased with the sculpture I configured

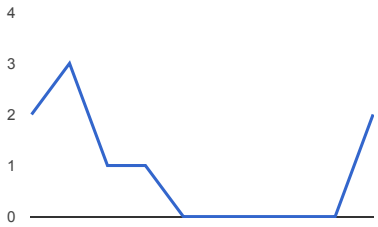


I strongly disagree: 5 **0** 0 %

Any final thoughts you want to share?

In ther diagramm you should not show the intensity on the same level as the burned calories.

Anzahl der täglichen Antworten

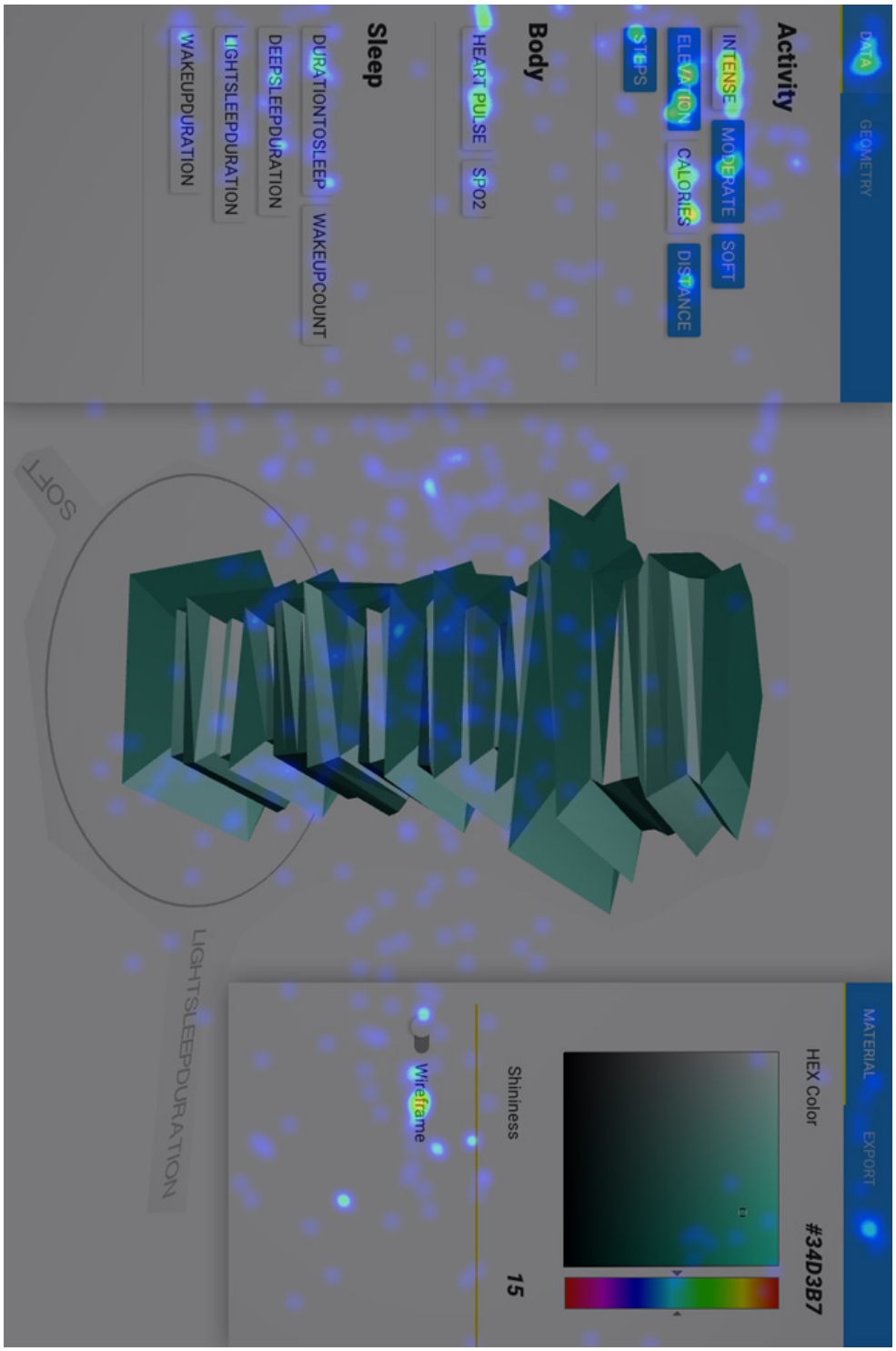


C.3 User Study Results

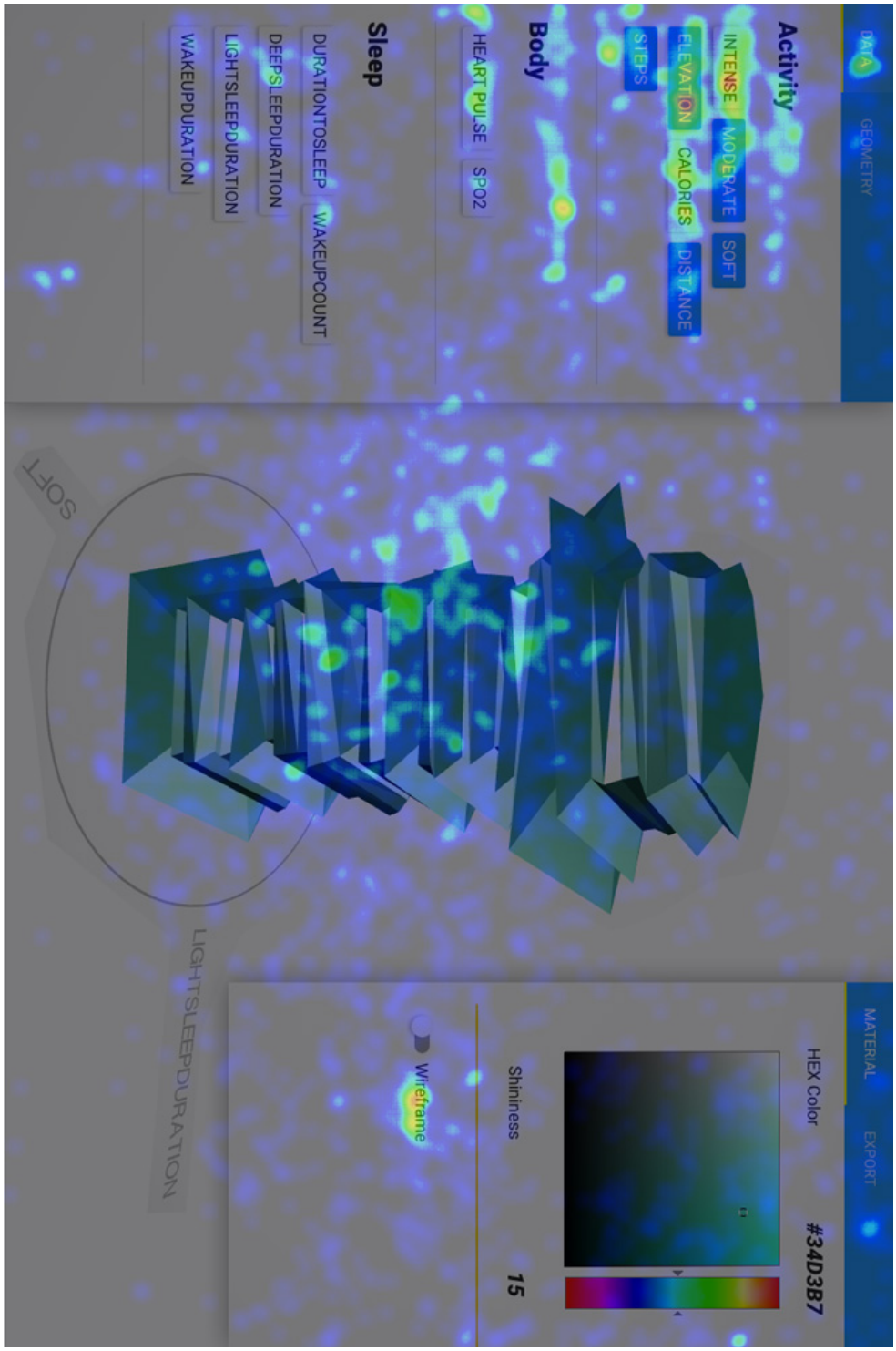
User Nr.	Missed Features	Widgets Problems	Remarks	Comments
1	Geometry Tab, Wireframe, turn off labels	Couldn't operate toggle button, tryed to slide it as if it would be a touch gesture	-Intuitively navigatet the sculpture -Not many comments about the dashboard	-Obvius to make a new sculpture with the button -Dashbaord Activity Graph wasn't logically grouped, remove calories -Interesting for comparing the relationship between variables -Found the normalization of the values illogical -Its ilogical that values equaly increse or decrease evenly -What tangible benefit has the sculpture? Dataviz should be precise -Didn't like the interpolation visualization
2	Turn off labels	Couldn't operate toggle button, tryed to slide it as if it would be a touch gesture	-Didn'd understand SPO2 -Intuitively navigatet the sculpture -Understood the idea of the stacked radar chart -Not many comments about the dashboard	-What is this shape supposed to tell me? -Understood that the data is visualized over time
3	Interpolated view with several variables, turn off labels		-Wanted to enter values, didn't quite get the idea of viewing data from your tracker -Not many comments about the dashboard	-I can add an achsis to the sculpture -Don't quite understand the interpolated mode -For the date range slider day 1 is the first day
4	turn off labels		-Navigation in space was later discvored -Not many comments about the dashboard	-Surprised of how much data one can gather about one self -Don't get the difference of adding more variables to the achsis -I don't know what my heart rate is -I add the values to the sculpture -My data was already tracked and it is visualized differently -With many variables it is difficult to see what difference it made -Really liked to see the different days -Don't understand what interpolated does
5	Wireframe toggle		-Kind of made a story out of the data, "this day I made a lot of intense activity" -Found the interpolation slider step bug	

C.4 Heat Map Images

Click Heatmap



Eye-tracking Heatmap



Contents of the enclosed CD

Thesis

- LaTeX Document
- PDF File

Presentations

- Initial presentation
- Final presentation

Activity Sculpture Web Configurator

- Prototype sketches
- Source code
- Gitlab and Github mirrors
- Instructions for deployment
- Login Data

Sculptures

- Prototype sketches
- .stl 3D print ready example files

User Study

- Questionnaire
- Results
- Heat map images