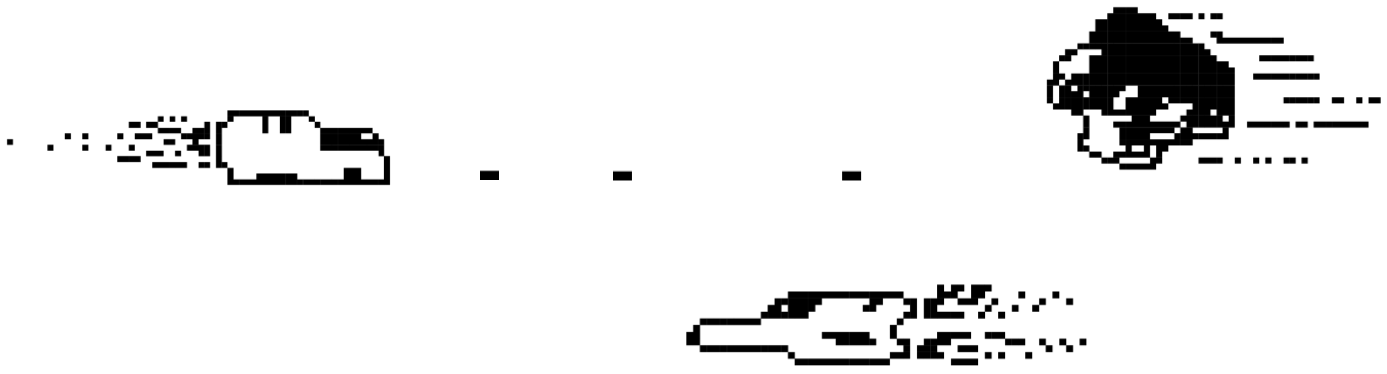
The Playfulness of Modsy.

The creation of a more playful music production controller



Graduation Project Thesis

Olivier Mathijssen, s2079607

09-07-2021

Supervisor: Wouter Eggink

Critical observer: Erik Faber

Bsc Creative Technology

University of Twente

Abstract

Play is an important aspect of life. Playful behaviour might be a vital element in human creativity, learning, development and much more. Music production and performance is an activity that is highly intertwined with play behaviour and play might be essential in the successful creation or reproduction of music. This thesis will describe the design optimization of a tool that can be used during this process, the Modsy controller, with focus on one specific design aspect, its playfulness. This thesis will assess whether playfulness can effectively be used in the design of the Modsy controller and how this can impact the user experience with this controller. In this thesis report, a variety of literature related to playfulness and product design will be analysed. Thereafter, a theoretically more playful Modsy controller will be created through the use of the Creative Technology design process and eventually evaluated on its playfulness and user experience. Results of this evaluation suggest that the created prototype was only slightly more playful in form, not more playful in interaction and not directly positive for the user experience compared to the current Modsy controller. However, the potential of designing for playfulness has been established and its implementation remains very flexible. Future research could help to assess how this playfulness could alternatively be implemented in the design of the Modsy controller and provide value for the customers of Modsy.

Acknowledgements

I would like to thank Wouter Eggink and Erik Faber for their support and feedback throughout this thesis project. I would also like to thank the Modsy boys Bram van Driel and Robbert-Jan Berkenbos, for their support and collaboration throughout this process. Lastly I would like to thank Lars Wintermans and Leon Groothaar for the occasional games of tennis to help keep the brain sane.

Glossary

This glossary describes terms relevant to this thesis. Reading from top to bottom is recommended, as some definitions make use of other definitions.

1. Digital Audio Workstation (DAW)

A software application used by producers and performers to record, arrange, compose, mix and master audio. This software application can be compared to a digital version of a physical studio filled with music gear.

2. MIDI (Musical Instrument Digital Interface)

Musical Instrument Digital Interface (MIDI) is a standard technical communication protocol for the communication between instruments, computers and other audio devices. The MIDI protocol can be used to communicate musical data like the note on/off messages or control changes.

3. MIDI Controller

A MIDI controller is a hardware or software device that can be used to control MIDI parameters. The controller uses the MIDI protocol for communication. A MIDI controller can take different shapes for different purposes. Within music production, the most well-known are keyboards, control surfaces or beat pads.

4. Ableton Live (Live)

Ableton Live is a popular DAW used by both music producers and performers. Ableton Live is relevant for Modsy since Modsy will exclusively function within this DAW during the first years of production.

5. Device (Ableton Live)

The term device is used as an umbrella term to describe all instruments, audio effects or MIDI effects that can be used in Ableton Live.

6. Plugin (Ableton Live)

A software component that enhances music production functionality. This component is “plugged in” to a DAW to add extra functionality, for instance for digital sound synthesis or processing.

7. Mapping

The connection between a software parameter in Ableton and the physical Modsy controller. The word ‘mapping’ can refer to a single parameter that is mapped or refer to a group of software parameters that is mapped to the Modsy controller.

8. Control surface (for a DAW)

A control surface is a hardware device (MIDI controller) that a musician can use to control his DAW environment. This controller concerns itself primarily with functionalities for DAW navigation, transport buttons and editing of notes and samples.

Index

List of figures

Chapter 1 – Introduction

1.1 - General Introduction

Play is a way for humans to solve problems, learn, and create. An activity that can bring joy and pleasure and can be seen throughout human life, from children running around in a playground to 80-year-old man pranking each other during a game of chess.

The benefits of play can be seen throughout the internet, research papers, and company strategies. Organisations like UNICEF or the LEGO group state play as one of the key aspects in the learning process of children (UNICEF, 2018; Zosh et al, 2017). And with good reason, research shows that play is a very important aspect in this process (Singer, Golinkoff & Hirsh-Pasek, 2006). Even for adults there seems to be a huge benefit that play can have. Play could help with worry, anxiety, strengthen relationships and make people more creative (Wignall, 2019; Bateson, 2015). What is this panacea called play and how could we use it in everyday practice?

Some activities seem to embody play more than others. Especially creative disciplines seem to rely heavily on the aspect of play. A painter, for example, might not find himself creating a new masterpiece if he does not have a playful attitude in the process. The discipline relevant for this paper, music production, also seems to draw great benefit from the act of play. The relevance of play in music production will be discussed shortly after the introduction, but we can state that playful behaviour might be a key element in music production and performance.

This thesis is aimed to improve the design of a music production controller called the Modsy controller. Since play is an important factor to consider during the music production process the tools used during this process should then also suit this playful behaviour. One important requirement is that the controller should provide musicians the right playful experience. Therefore it should be considered what a playful experience actually is and how the Modsy controller can be designed to enhance such an experience. This thesis will dive into the potential that playfulness can have in the design of the Modsy controller.

This first chapter will provide an introduction into the topic of playfulness within the context of the Modsy controller and provide an outline for the remainder of the report.

1.2 – Play and music

How important is play within music production? There will be an in-depth analysis of play and playfulness in the background research section, but before this formal analysis the importance of play in regards to music production and performance should be addressed. Because if play is of no importance, why even bother?

First of all, play is known to be important for creativity and innovation. Research from Bateson and Martin (2013) shows how both play and playfulness can facilitate creativity and innovation in both the natural world and human society. Further analysis of Bateson (2015) connects a playful attitude to the creativity seen in many well-known scientists, musicians, and other creatives. Other research supports this by stating that even for students gifted in more scientific disciplines like mathematics personal playfulness can have a positive influence on creativity (Chang, 2013). Creativity and music production go hand-in-hand, depending on who you ask creativity might be one of the most important characteristics of a good artist. Creative use of voice, instruments or effects could make or break an artist.

Furthermore, Csepregi (2013) ascribes play as one of the elements that brings life to music. In his research he states that play may be very important in the reproduction of music and musical composition. As stated by Csepregi:

> It is, in part at least, his sense of play that fills the music with life because, by “sharpening notes”, or “putting a little accent” or introducing “a slight change in tempo”, the performance is accomplished with a certain degree of unpredictability, individuality, and spontaneity. (Csepregi, 2013, p108)

Many artists state the importance of play in their music production process (Telekom Electronic Beats, n.d.). This might differ slightly for different genres of music, but overall it can be stated that play has large importance in the creation and performance of music. A playful attitude and mindset seem to be at the centre of creative music production. Without some form of playfulness music production and performance could become rather dull. This is the reason why it is so important to consider during the design of a product in this field. A music production controller should suit this playfulness, not disrupt it, and hopefully produce great value along the way.

1.3 - The current status of music production

Music production has changed a lot over the last 20 years. The rise of computers created a new way to create and manipulate music. Where musicians used to need whole rooms of equipment to create a song, musicians nowadays can record whole albums with the use of their phone (Pierce, 2017). While a phone might be slightly inconvenient, a laptop or computer is a perfect basis for music production and most modern musicians rely on software for music creation and manipulation. The software program at the core of this music production is called a Digital Audio Workstation, or DAW for short. An example of such a DAW called Ableton Live can be seen in figure 1. This software can be seen as the studio of a music producer. Inside a DAW a musician can use different digital instruments and effects that can be used within music production. A DAW also allows for the composition into a song and ways to mix, master and finalize your music.

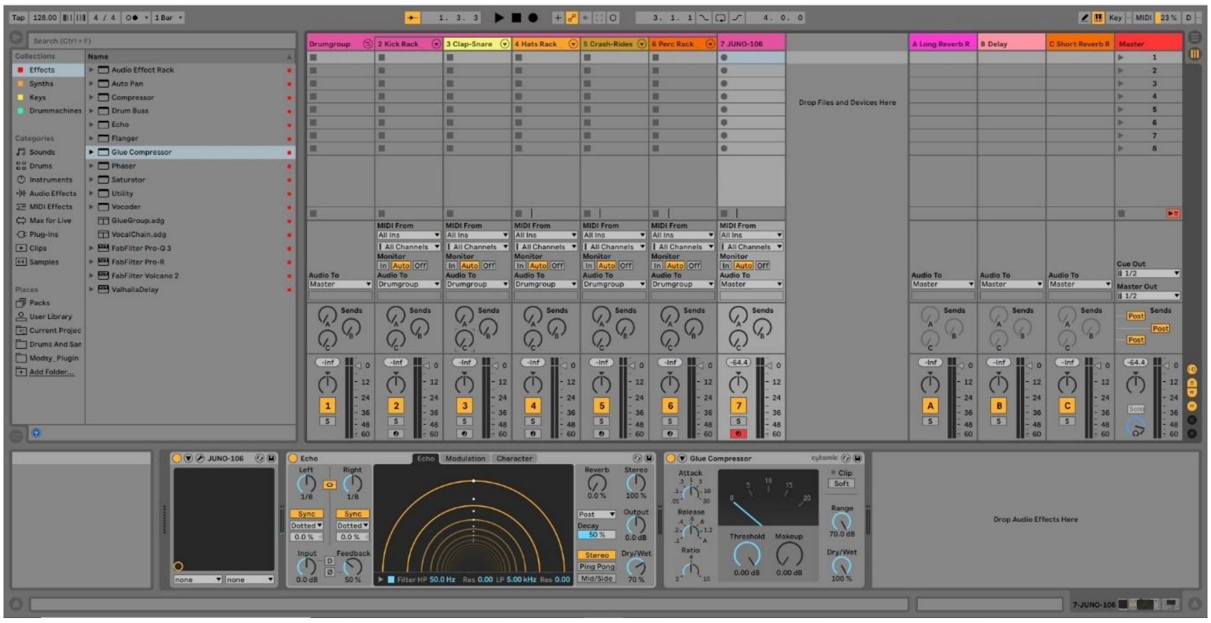


Figure 1 - Screenshot of DAW Ableton Live

Music production has become more accessible, flexible, and cheaper. However, there is also a downside to computer based music production. This is due to the fact that musicians can only use their mouse and computer keyboard to manipulate the sounds of instruments and effects. This is far from the original feeling of analog instruments and effects. Analog instruments and effects create a workflow that is more expressive, creative and it's a whole lot more fun. Tweaking multiple parameter at once, making mistakes that turn out to be masterpieces and collaborating with peers in musical jams are examples of the pros of this analog control.

1.4 - Modsy

This is where Modsy steps in. Modsy is a product by Weirdly Wired. A start-up founded by Olivier Mathijssen, Bram van Driel, and Robbert-Jan Berkenbos. All are Creative Technology students from the University of Twente. Modsy creates a way for musicians to get an analog feeling over their digital instruments and effects. Modsy is a hardware and software solution that creates an environment that allows musicians to instantly take physical control over any digital instrument or effect inside of their DAW. Figure 2 shows the current design of the Modsy controller, figure 3 on the next page shows how the Modsy controller works.

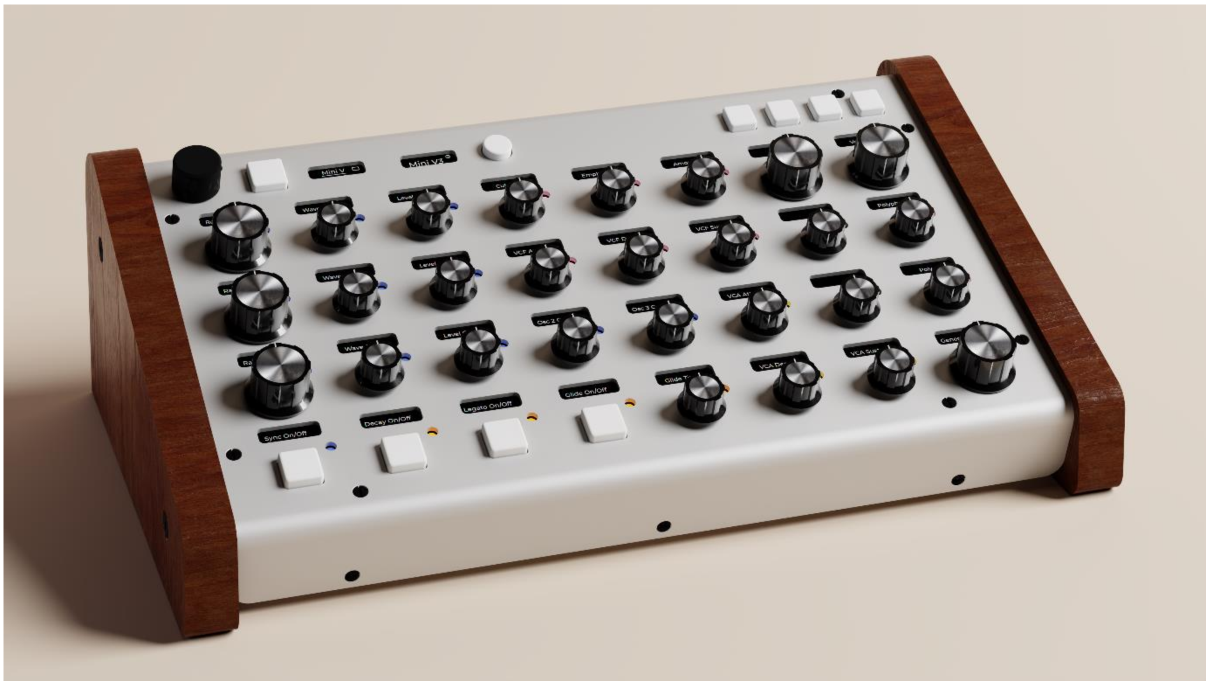


Figure 2 - Current Modsy controller design

The controller will be connected to the DAW of a user. The controller has 32 parameters to control an digital instrument or effect, with a display above each of these parameters. When a user selects a digital instrument or effect in his DAW (as can be seen in Image 3.1) and presses the mapping button on the controller (as seen in image 3.2), the controller automatically maps itself to the digital instrument or effect. All of the parameters will be linked to software parameters of the digital instrument or effect and the musician can start manipulating the sound right away. The displays above each parameter will then show the parameter's name and value. This is essentially what the Modsy controller does, it provides direct analog control to digital instruments and effects and ensures that the manipulation of these parameters is fully intuitive. This creates an analog workflow for any digital music production tool. The current prototype of Modsy can be seen in figure 4 on the next page.

The unique selling points or Modsy are:

- Automatic mapping to any digital instrument or effect within a DAW
- Display for each parameter with parameter name and value feedback
- Enough parameters for complete analog control

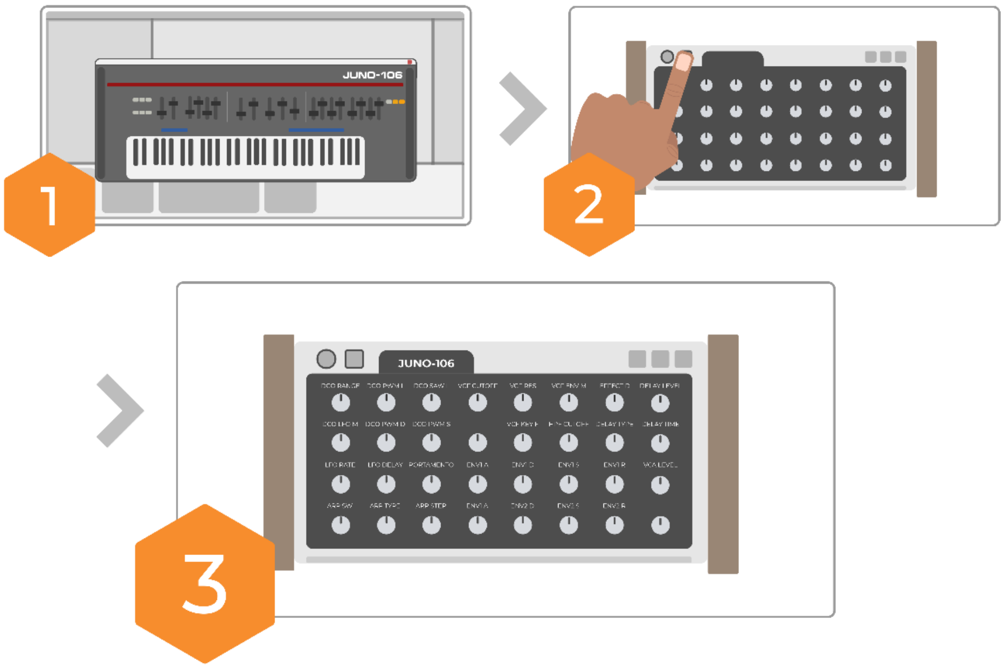


Figure 3 - Main Modsy functionality

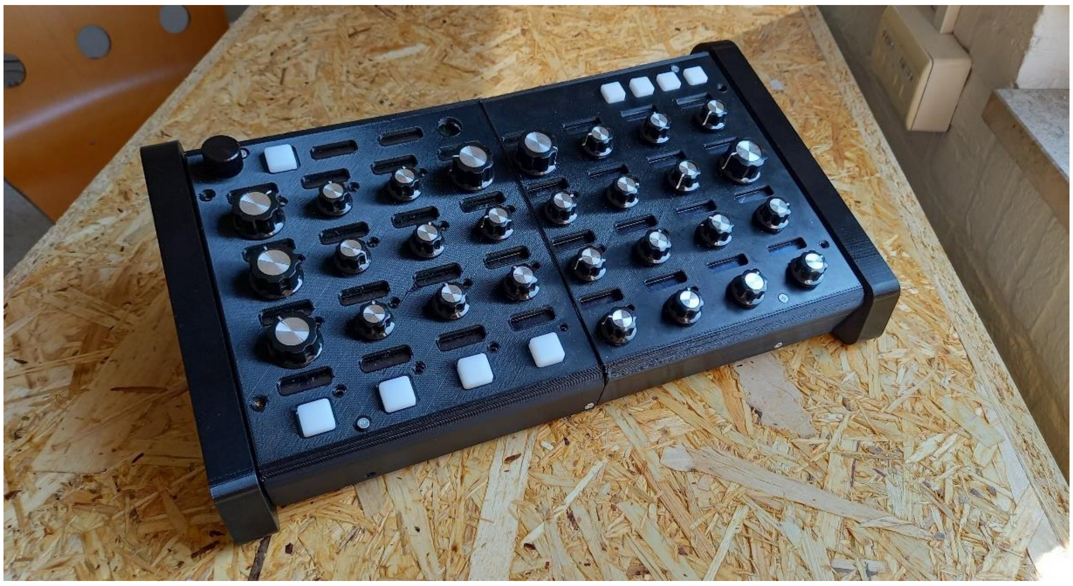


Figure 4 - Current Modsy prototype

1.5 - Project goal and research questions

The goal of the project is to determine how playfulness can be used in the design of the Modsy controller and develop a prototype that can be used to evaluate how playfulness affects the user experience with the Modsy controller. This will help to identify ways in which Weirdly Wired can utilize playfulness for future products. This thesis will try to answer the following main research question: **How can the playfulness of the Modsy controller be optimized to improve the overall user experience?**

Different sub questions have been created in order to answer this main research question. These questions will be answered with the use of literature research, prototyping and user testing.

- What is play and playfulness in the context of a music production controller?
- How can playfulness be used in the design of a music production controller?
- How can playfulness be evaluated?
- How does the playfulness of a product affect the user experience?

1.6 - Report structure

The report will be structured in the following way: Chapter 1 provided introduction to the problem and structure of the thesis. Chapter 2 includes an analysis of literature and state of the art. The literature review will cover play and playfulness within the context of a music production controller. Through literature analysis practical tools will be derived that can be used in the design of a more playful Modsy controller. In the state of the art section of chapter 2 both competitive and inspiring products will be analysed with tools derived from literature. Chapter 3 provides an overview of the methods and techniques that will be used in the remainder of the report. The ideation of a more playful Modsy controller is described in chapter 4, which will result in a preliminary concept that will be specified in chapter 5. Chapter 6 describes the realisation of this concept into a final prototype and chapter 7 the evaluation of this created prototype. Finally, the thesis will be concluded in chapter 8 and future work will be addressed. This report can be read in any way that suits the reader, however for the understanding of certain methods and techniques please see the method section.

Chapter 2 – Background research

The background research chapter is made up of two sections: literature research and state of the art. In the literature research section different topics will be analysed to frame play and playfulness in the context of product design and see how one can design for these concepts. The state of the art section will provide an understanding of the playfulness of products related to the Modsy controller and analyse inspirational products.

2.1 - literature research

The aim of this literature research is to analyse how playfulness can be used in product design and how this can affect the overall user experience. This literature research will look at the collective understanding of play and playfulness and determine how these elements can be applied in product design. This section will be structured in the following way: First, the concept of play and playfulness will be analysed. A definition will be derived that can be used in the remainder of the thesis. Thereafter, practical ways to analyse and design playful products will be discussed. Lastly, there will be an analysis of the effect of play and playfulness on individuals.

2.1.1 - Play

Due to different perspectives within the field of play, it has become a complex term that needs proper framing before usage. There are a lot of different definitions of play since play is ambiguous, complex, and context-specific (Shahri, 2014) Therefore, a clear distinction should be made in its meaning to prohibit confusion and conceptual problems (Starbuck & Webster, 1991). A good way to define play is to look at relevant literature within the context in which the term of play will be used. This method is applied by a variety of different researchers within the field of play (Gray, 2017; Starbuck & Webster, 1991; Arrasvuori et al, 2011).

Throughout history, there have been different ways to interpret play and describe its characteristics. One of the first theories about play was formed by Friedrich Schiller who explains play through the view that humans have certain abundant energy that they need to get rid of (Schiller, 1795). The first evolutionary theories of play were formed around 1893 by Karl Groos. His perception of play was that natural selection played a big part and play was seen as an activity that would help practice the skills needed to survive (Groos, 1898). In the years that followed play theories became more elaborate, Huizinga (1938) describes play as:

A free activity standing quite consciously outside "ordinary" life as being "not serious", but at the same time absorbing the player intensely and utterly. It is an activity connected with no material interest, and no profit can be gained by it. It proceeds within its own proper boundaries of time and space according to fixed rules and in an orderly manner. (Huizinga, 1938, p13)

Certain characteristics of play start to arise: absorbing the player, no profit can be gained, according to fixed rules. Huizinga also goes on to state that play plays an important role in the culture and construction of civilizations.

Modern-day play research builds further on these types of characteristics, with different researchers proposing different sets of characteristics to describe play behaviour. These characteristics are similar but differ slightly in framing and description. Most notable are the five characteristics proposed by Sandelands et al (1983) and the characteristics proposed by Peter Gray (2017), which were analysed but will not be discussed within the scope of this paper. William H. Starbuck and Jane Webster (1991) analysed different definitions of playful activities and found two elements to be consistent throughout all definitions, which is that playful interactions elicit involvement and provide pleasure. These two elements can also be found in more practical research that deals with play in design (Arrasvuori et al, 2011). Since there is only a need for a conceptual understanding of play within the scope of this thesis the analysis of Starbuck and Webster can be utilized. Within the scope of this thesis, play will be seen as a form of activity that elicits involvement and provides pleasure.

2.1.2 - Playfulness

Playfulness is a term that can be used to describe organisms, products or interactions, its meaning is dependent on the perspective that is used. Playfulness could be seen more like an attitude or modality, where we can take ‘the attitude of play’ without the activity (Sicart 2014). Playfulness can be used to describe organisms, or more specifically humans (Bateson, 2014; Proyer, 2013). In this context, the term can be used to describe a state of mind (Webster et al, 1993) or a character trait (Lieberman, 1977; Yager, Kappelman, Maples & Prybutok, 1997). Both perspectives could be valid approaches to playfulness (Webster, 1991). Playfulness could also be used in the context of products or interactions (Shahri, 2016; Arrasvuori, Korhonen & Boberg, 2010; Hong, 2012). Here, the term takes on a different role, it is used to describe the ability of these products or interactions to alter the state or trait of playfulness in humans.

Playfulness as a trait in children and adults can be described by different characteristics. Describing playfulness as a trait means that it can be seen as a distinguishing quality or characteristic of an individual. According to Nina J. Liebermann (1977) playfulness in children can be identified through the following 5 characteristics: physical spontaneity, manifest joy, sense of humour, social spontaneity, and cognitive spontaneity. These five character traits have been verified in research by Lynn A. Barnett (1990) and were supported by Susan E. Yager et al (1997). Playfulness in adults has been identified through slightly different characteristics. Glynn and Webster (1992) developed an adult playfulness scale that can be used to assess playfulness in adults. The scale uses five different facets of playfulness: spontaneous, expressive, fun, creative, and silly. This playfulness scale is used and supported in later research (Proyer, 2012).

Playfulness as a state of mind that can be described and measured with the use of flow theory. Many researchers have defined playfulness as a state of mind (Arrasvuori et al, 2011; Webster et al, 1993; Apter, 1989). Which can be described as a person's emotional state (Merriam-Webster, 2021). Within certain conditions, this state of playfulness can be enhanced (Proyer, Gander, Braeuer & Chick, 2021). The state of playfulness is highly related to flow theory by Csikszentmihalyi (1975). As described by Csikszentmihalyi: "Flow is a state in which people are so involved in an activity that nothing else seems to matter; the experience is so enjoyable that people will continue to do it even at great cost, for the sheer sake of doing it." (Csikszentmihalyi, 1990, p4) Csikszentmihalyi describes flow to be characterised by total focus, engagement and altered sense of time. Flow is an established concept that is used in a variety of game design and interaction research to describe optimal playful behaviour (Klarkowski, Johnson, Wyeth, Smith & Phillips, 2015). This claim is supported by Shahri (2016) who describes flow as the ultimate state of any sort of play. Flow and playfulness are highly related and have a positive relationship with one another (Reid, 2004). This means that the level of flow experienced during an interaction could provide good inside into the playfulness of that interaction (Webster et al, 1993). Measuring flow can be performed through interview questionnaires, experience sampling methods, and self-report questionnaires (Lonczak, 2020). Most notable are the Flow State Scale by Jackson & Eklund (2002) and Flow Short Scale by Rheinberg, Vollmeyer, and Engeser (2003).

Playfulness as a state or trait can both be relevant within the context of product design. In the next section of this paper, there will be more attention as to why certain products or interactions are seemingly more playful than others. For now, it can be stated that while using certain products or partaking in certain interactions humans can feel an enhanced sense of playfulness (Webster et al, 1993). During the usage of a product or interaction, this playfulness can be described as a state of mind (Arrasvuori et al, 2011), or the long term effect of playfulness intervention can be assessed on a trait level (Proyer et al, 2021). Since playfulness expresses itself in humans, it can be assumed that the experienced playfulness of a product is related to the change in the playfulness of an human due to product interaction. The way playfulness is used within product design is entirely dependent on the scope of the design problem and the preferred outcome. Within product design, it could be interesting to look at both playfulness as a state during product interaction or the effect on playfulness as a trait after product interaction.

2.1.3 - The playfulness of products

The playfulness of products can be described through form and function. Playfulness is an important attribute of products. Blijlevens, Creusen & Schoormans (2009) were able to show that playfulness was one of the main attributes that consumers used to distinguish between product appearances. Analytic tools can be used to better understand the root of this playfulness. Different products can, and should, be analysed, since many sources point out that playfulness can be best studied through examples (Shahri, 2016; King & Chang, 2015; Hong, 2012). Shahri (2016) proposes a taxonomy for this purpose, which can be seen as a way to group certain elements based on specific factors. In his taxonomy, two elements are used to describe the playfulness of a product: form and function. Products focussed around form can be described as having a “distinct/concrete quality of visual engagement” (Shahri, 2016, p84) that some people associate with being playful. Products focussed around function show their playfulness upon interaction with these product, as Shahri states: “This category is about creativity and play in which users can instigate” (Shahri, 2016, p95). This is most evident in products such as toys, but also a hair comb can have a certain level of functional playfulness when used to hold nails in place to drive into a wall. It is important to identify both elements since together they could provide the full scope of the playfulness of a product.

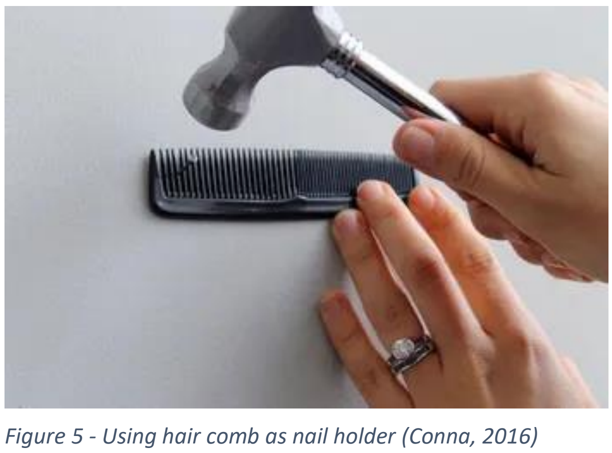


Figure 5 - Using hair comb as nail holder (Conna, 2016)

Analysis based on form and function is rather subjective. It is dependent on the type of user how both of these elements are interpreted (Townsend, Montoya & Calantone, 2011). Furthermore, it is dependent on the experience of the observer how the playfulness of a product is perceived. If a user discovers new functionality of a product his sense of playfulness about this product might increase. We can take the hair comb example of Sharhi (2016). If a user knows he can use a hair comb to hold a nail in place to drive into a wall, as can be seen in figure 5, he might judge the hair comb more playful than somebody who does not know this. Thus, one has to be careful when analysing products or ideating about functionality.

Underneath this perceived visual and function playfulness lie different aspects that contribute to the playfulness of a product. Design frameworks have been developed for identifying and designing for these playful aspects. One of these frameworks is the Playful Experiences framework (PLEX framework), which has been established through the analysis of different literature sources and performing empirical investigation (Korhonen, Montola, Arrasvuori, 2009). The PLEX framework is meant to provide an understanding of what kind of aspects play a role in playful products and interactions and how to apply these aspects in design.

The framework defines the following categories for playful interactions: Captivation, Challenge, Competition, Completion, Control, Cruelty, Discovery, Eroticism, Exploration, Expression, Fantasy, Fellowship, Nurture, Relaxation, Sensation, Simulation, Subversion, Suffering, Sympathy, Thrill. This framework was later expanded to make it more widely applicable, two more categories were added: Humour and Submission (Arrasvuori et al, 2010). The categories of this framework can be seen in table 1 below.

PLEX category	**Explanation**
Captivation	Forgetting one's surroundings
Challenge	Testing abilities in a demanding task
Competition	Contest with oneself or an opponent
Completion	Finishing a major task, closure
Control	Dominating, commanding, regulating
Cruelty	Causing mental or physical pain
Discovery	Finding something new or unknown
Eroticism	A sexually arousing experience
Exploration	Investigating an object or situation
Expression	Manifesting oneself creatively
Fantasy	An imagined experience
Fellowship	Friendship, communality or intimacy
Humour	Fun, joy, amusement, jokes, gags
Nurture	Taking care of oneself or others
Relaxation	Relief from bodily or mental work
Sensational	Excitement by stimulating senses

Simulation	An imitation of everyday life
Submission	Being part of a larger structure
Subversion	Breaking social rules and norms
Suffering	Experience of loss, frustration, anger
Sympathy	Sharing emotional feeling
Thrill	Excitement derived from risk, danger

Table 1 - PLEX category description

The framework was developed in relationship to digital games but can be applied to interactive product experiences as well (Arrasvuori et al, 2010). Further research provided insight into the practical implementation of the framework which included the creation of PLEX playing cards together with different methods to apply the PLEX framework in the design process (Arrasvuori et al, 2011). Other frameworks have been analysed, but show less relevance within the field of product design. Examples include the framework of Costello and Edmunds (2007) and the playfulness model developed by Garris, Ahlers & Driskell (2002).

The PLEX framework can be practically implemented in ideation or product analysis. In design ideation, the PLEX cards could serve as a source of inspiration (Lucero & Arrasvuori, 2010). The PLEX cards, and thus the PLEX categories, could serve as a way to come up with more playful concepts. The different PLEX categories can also assist in a heuristic evaluation of certain products or interactions (Lucero, Holopainen, Ollila, Suomela & Karapanos, 2013). The different categories can help to identify the amount in which playfulness is involved in a product or interaction. However, it should be noted that such an evaluation has reported flaws and should be done with caution (Lucero et al, 2013).

2.1.4 - The effect of play and playfulness

Play and playfulness seem to have a real effect on humans and their environment, but the root of this effect is difficult to scientifically determine. As Bekoff states “methodological insufficiencies do not allow an experimental manipulation that deprives an organism solely of play experience." (Bekoff, 1972, p. 422). Meaning that measuring the results of play deprivation is seemingly impossible since organisms cannot simply shut off their play factor. However, the effect of play seems to be undeniable. While observing playfulness in design we could derive many reasons why we would design for playfulness, including amusement, delight, evaluation, emotional boost and behaviour change (King & Chang, 2015). However, it is harder to scientifically proof that playfulness is the cause of these changes. Scientific sources have to be analysed to find the actual effect of playfulness.

Positive effects

Playfulness seems to have the most effect on the aspects of learning, pleasure, and involvement. Learning and play seem to be highly related to one another. Organizations like UNICEF and the LEGO group describe play as one of the major important aspects in the education of children (UNICEF, 2018, Zosh et al, 2017). Other research supports the importance of play-based learning, with Samuelsson & Johansson (2016) stating that play is a very important part of the life of children and their creation of meaning. Singer, Golinkoff & Hirsh-Pasek (2006) even go as far as to state that play is equal to learning, stressing the importance of play in diverse areas of child development. Even for adults, the importance of play behaviour in learning has been stated and has shown positive outcomes (Diaz-Varela & Wright, 2019; Tanis, 2012).

Pleasure is at the very root of playfulness. As stated earlier in the paper, play itself can be described with the characteristic that it provides pleasure. This is best reflected in research by Bridget Costello and Ernest Edmonds (2007) who constructed the pleasure framework around the subject of playfulness exploring the many ways in which playful interaction can lead to pleasure. Research by Guitard, Ferland & Dutil (2005) supports this by stating that pleasure is highly related to play behaviour.

Involvement is a strong effect of playful interaction. As both Csikszentmihalyi (1975) and Starbuck & Webster (1991) point out, playfulness can highly affect task involvement. Modern research by Chan & Ma (2014) support this by concluding that playfulness is a key determinant in the involvement and focussed attention with the use of social media. Next to these main positive effects, play has shown to affect productivity, product quality and job satisfaction (Starbuck & Webster, 1991). Furthermore, playful interaction can be a significant expressive motivational factor (Sandelands et al, 1983).

Negative effects

Playfulness may also lead to negative effects, most notably over-involvement and task completion. Strong involvement could create unpleasant tensions and worries that offset immediate pleasures (Webster et al, 1993). Webster states that due to over-involvement playful experiences could enhance short-sightedness, concentration and persistence which could negatively impact task completion. To support this, Pang (2012) points out that playful interactions which resemble a flow state could lead to over-involved behaviour like gambling-addiction or game addiction.

Task completion may also be affected when applying playful interactions. Sandelands et al (1988) found that merely labelling a task as play results in people taking longer to complete a task. Starbuck & Webster (1991) point out that the higher levels of involvement and pleasure during playful interaction might result in a longer continuation of these tasks. They point out that people that perform very playful tasks enjoy what they are doing and concentrate on their immediate activities and the pleasure those return. They feel less inclined to think about longer term goals and can underestimate the passing of time. Next to this, they might not want to switch tasks if deemed necessary. The severity of the consequence of these longer task completion times is then dependent on the task that needs to be completed (Starbuck & Webster, 1991).

2.1.5 - Conclusion literature research

The goal of this literature research is to analyse how playfulness can be used in product design and how this can affect the overall user experience of a product. In order to discuss the elements of play and playfulness both were analysed and their meaning discussed. This resulted in play being able to be expressed as a form of activity that elicits involvement and provides pleasure. Playfulness can be expressed as a term that can be used to describe a state or trait of an individual, or as a product or interaction that has the ability to alter the playfulness state or trait of an individual. Next to this, different sources were analysed to define methods for playful design. This resulted in a framework and taxonomy that can be used in both analysis and ideation. The framework that could be used is the PLEX framework with 22 categories that could enhance playfulness. The taxonomy allows distinction in form and functional aspects that could enhance playfulness. With the use of both methods, designers should be able to form a good conceptual idea of playfulness within their field of design and be able to ideate possible playful design solutions. Lastly, the analysis of literature provided possible positive and negative effects that playfulness could have on user experience. Possible positive effects could be an increase in involvement, pleasure and learning ability, of which all could have a strong effect. Possible negative effects include over-involvement and longer task completion, where the severity of the consequence of both is dependent on the task or interaction at hand.

The main limitations of this literary research are the interpretation of both play and playfulness and the analysis of playfulness frameworks. During the analysis of both play and playfulness many different perspectives could be derived within the field of play research. In the context of this research, the essence of both aspects was analysed and described, but a more structural analysis could have been performed to derive a more precise definition within the scope of this thesis. Next to this, the analysis of different playfulness frameworks for product design is dependent on personal interpretation of these frameworks, since there is limited literature on the application and comparison of these frameworks.

Future research could focus on the impact of products on playfulness as state or trait phenomenon. According to the analysis in this research there is serious potential in the use of playfulness within the field of product design. There is a need for a better understanding of both the state and trait of playfulness within the context of product design. Future research could investigate what products could alter the state or trait of playfulness and in what way this could be applied. Furthermore, there is a need for more research into the implementation of playfulness frameworks in product design.

Within the context of Modsy

The knowledge gained from this literature research can directly be applied to the Modsy controller. Playfulness as a state of mind is of most interest within the scope of this research since the controller should have the ability to trigger a certain playful state during usage. As discussed in the literature review playfulness as a state of mind can be assessed with the use of flow theory. Flow is positively correlated to the state of playfulness of an individual and has been used in research to determine the amount of playfulness experienced. Playfulness as a trait could be interesting for future research, but is of no interest within the scope of this thesis. Next to this, the PLEX framework and taxonomy proposed by Shari can be used in the ideation and analysis phase of the project. Both of these could be good tools to identify potential playful aspects of products and reasons why products are seemingly more playful. Lastly, the identified effects of playfulness on user experience might be useful when discussing certain outcomes of playfulness enhancement. Some of these effects might not be applicable for the Modsy controller since for instance the negative influence on task completion could be less troublesome within the activity of music production. This could depend on the type of producers and performers, while some producers and performers do not value efficiency and effectiveness other producers and performers do.

2.2 - State of the art

This section of the thesis will provide an analysis of different products. These products will be analysed with the use of two tools identified in literature research for the assessment of playfulness. This assessment of playfulness will concern the ability of these products to enhance the state of playfulness of a user.

The two tools that will be used are the PLEX framework and a taxonomy used by Shahri. The PLEX framework proposes 22 categories that could influence the playfulness of a product or interaction. These categories are listed in table 1 in chapter 2.1 together with their description. The taxonomy allows a distinction between playfulness in both form and function.

How will analysis with both items be performed?

The products will be analysed on both playfulness in form and function based on their features and appearance through tutorials, online documentation and reviews. This will provide a good sense of the product its playfulness in form and function.

Analysis based on PLEX categories will also be performed by an assessment of the feature and appearance of the products. During this analysis all 22 PLEX categories will be considered and their importance and presence within the product will be noted in an excel sheet. The most relevant and interesting PLEX categories have been described in detail in this report. An example of the full analysis has been included as item F in the appendix.

The use of tutorials, online documentation, and reviews of both analysis does allow the possibility of biased interpretation of the product its features and characteristics. Thus, the analysis should be interpreted with caution.

User experience

As stated in the literature research, analysis based on playfulness in form and function can be rather subjective. This is due to the experience certain users have with products. However, the user group of certain products is likely to have similar experience. This shared experience could lead to a similar view on the playfulness of these products. The user group of the Modsy controller is advanced and professional music producers and performers. The analysis of the playfulness of various products will be based on the experience of the intended user group of that product.

2.2.1 - Direct competitors

Some products can be seen as direct competitors of the Modsy controller. These direct competitors will also be analysed based on their competitive advantages in the market and possible weaknesses.

Ableton Push

Description

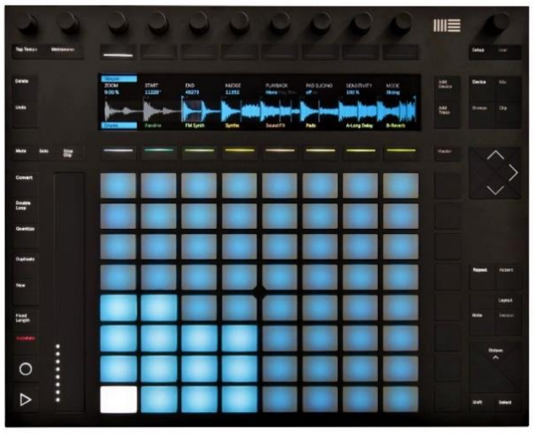


Figure 6 - Ableton Push controller (Kytary.nl, 2021)

Ableton Push (Ableton n.d.) is a control surface for the DAW Ableton Live. This means that this controller has both a hardware interface and software element that makes it well-integrated within a DAW, in this case, Ableton Live. The hardware controller can be seen in figure 6. This controller is meant to give direct tactile control over the Ableton Live environment. This control includes Ableton navigation, functions, and direct control over a variety of Ableton native instruments and effects. The controller is hard to learn due to the many functions that are available on the controller and the way that this controller can be operated. It is generally stated that the advantages of the controller only arise after a longer period of usage (Romanwave, 2020).

Competitive analysis

The Ableton Push has similar functionality to the Modsy controller. This is mainly related to the instrument and effect control inside of Ableton Live. Next to this, the Modsy will also be an Ableton exclusive product in first years of production. The price point of the Push, 600 euros, is very similar to that of the Modsy controller. The pros and cons compared to the Modsy controller have been listed in table 2 below.

Pros	**Cons**
Great Ableton integration	Steep learning curve
OLED display with detailed visualizations	Can only be used effectively with Ableton Live.
High quality build	Device mapping does not work well for third party devices.

Table 2 - Competitive analysis Ableton Push

Playfulness analysis

Playfulness analysis was performed for both the form and function of the product and included a general PLEX category analysis as stated in the beginning of this state of the art section. Both are described below.

- **Form**
  Playfulness in form can be seen in the OLED display and LEDs located on the controller. During the usage of different instruments or effects the user receives playful feedback through these elements. Both elements respond in relation to the music that is being created and could be seen as playful visual feedback. It should be noted that this only occurs during actual usage of the controller.
- **Function**
  The functionality of the Ableton Push provides greater and more direct control of music production tools. The control over these music production tools is directly related to play behaviour and thus the controller has strong functionally playful aspects. As will be explained through the use of PLEX categories the Push can function in a very playful way.

Analysis with PLEX categories indicates the following most prominent categories:

- **Captivation**, users can completely lose themselves in the environment of the Push and the control that is possible with the controller.
- **Discovery/ Exploration**, within the menus and functionalities available on the controller there is a lot of different functionality to discover and further explore. Possible software updates can further elaborate features, and has done so in the past (Ableton, 2017).
- **Sensational,** the visualizations, auditive feedback and led feedback trigger a lot of senses in the users of the controller.

Overall for the user group of the Ableton Push the perceived playfulness is **moderate to high**.

Native Instruments Komplete Kontrol

Description

Native instruments is one of the heavyweights within the digital music production industry. The company develops both hardware and software for the music production process. Most relevant within the scope of this research is the Komplete Kontrol line of products (Native Instruments, 2021).

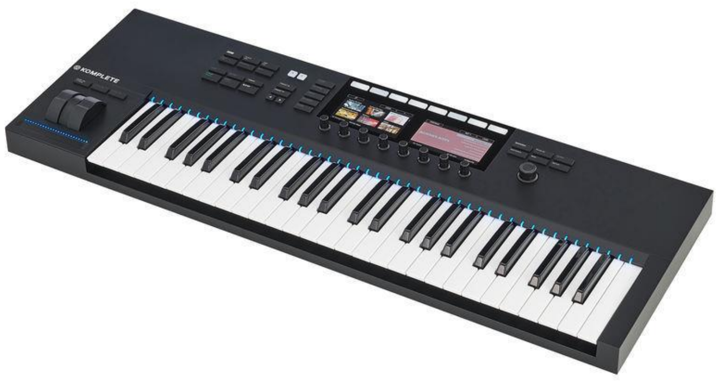


Figure 8 - Komplete Kontrol controller (Thomann, 2021-a)

One of these products named the Komplete Kontrol S49 Mk2 can be seen in figure 7. These products provide direct control over different digital instruments and effects within the Native instrument environment which can be seen in figure 8. This environment can be used inside different DAWs. The controller has relatively low costs and high functionality.

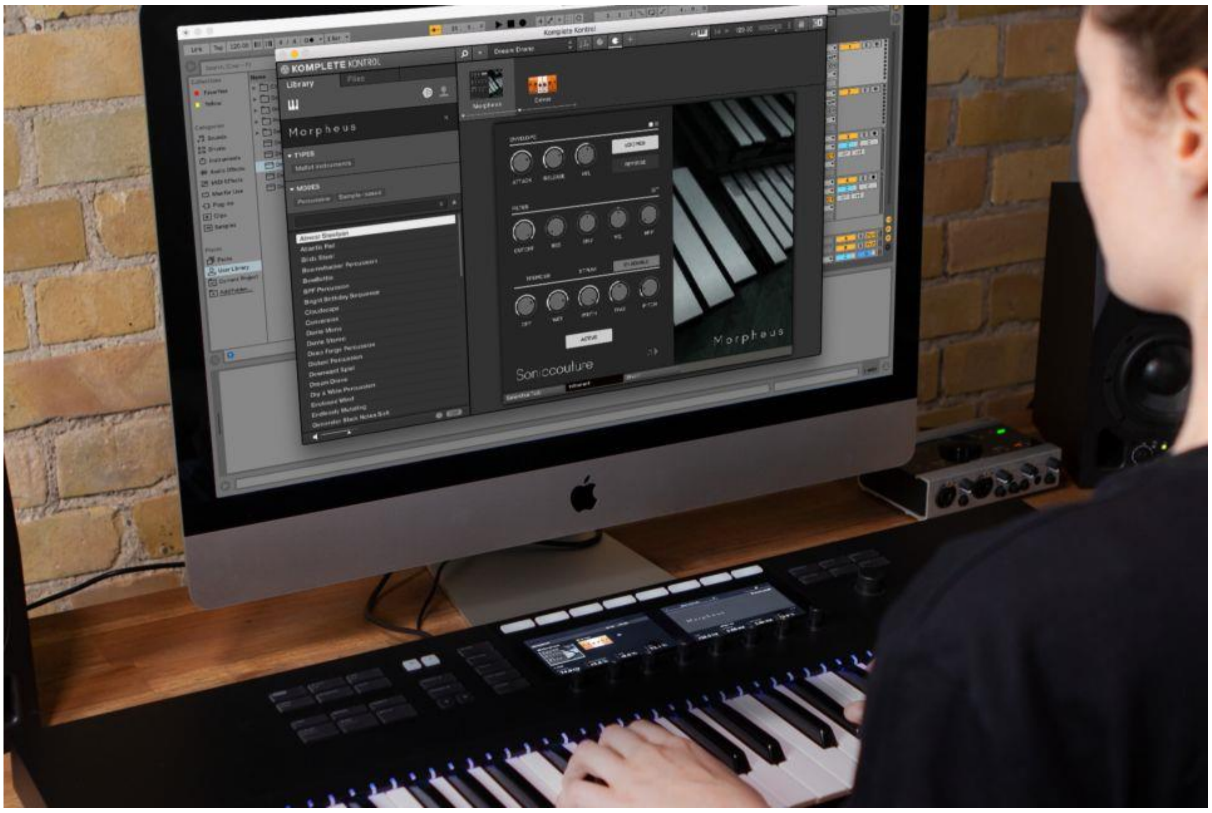


Figure 7 - Komplete Kontrol software environment (Native Instruments, 2021)

Competitive analysis

The Komplete Kontrol series can be seen as a direct competitor of the Modsy. However the controller addresses slightly different user needs. Instead of providing the user more analog control the Komplete Kontrol series is aimed at quick musical results and providing an efficient music production experience. The product line provides keyboard functionality next to device control, which broadens the scope of potential customers. The controller can automatically be linked to digital instruments and effects, however this only works within the Native Instruments environment. The pros and cons compared to the Modsy controller have been listed in table 3 below.

Pros	**Cons**
Good value for money	Device control only works well for NSK devices
Large amount of control available. Including keys, instrument control and DAW control.	A lot of extra control users might not need
Big display with colour for feedback.	Cheaper build quality
Works within all DAWs	

Table 3 - Competitive analysis Komplete Kontrol

Playfulness analysis

Playfulness analysis was performed for both the form and function of the product and included a general PLEX category analysis as stated in the beginning of this state of the art section. Both are described below.

- **Form**
  A visual playful aspect is less present in the design of the Komplete Kontrol controller. The visual cues are very straightforward. Some playfulness can be found in the LEDs related to the keyboard of the controller.
- **Function**
  A user can have full control over instruments and effects directly from the controller, this creates a very playful environment during usage. Due to Keys, DAW control, and parameter control a music producers should have a lot of functionality to discover and create sounds.

Analysis with PLEX categories indicates the following most prominent categories:

- **Captivation**, due to the integrated keyboard and extended functionality it is possible to fully control instruments and effects from the controller itself. This makes it easier to lose yourself in certain instruments or effects.

- **Expression**, Due to the extended control and integrated keyboard the controller is very suited for expression of your musical ideas.
- **Control,** there is high control due to the scope of the functionality directly available on the controller.

Overall for the user group of the Komplete Kontrol series the perceived playfulness is **moderate to high**.

Softube Console 1

Description

The Softube Console 1 aims to improve workflow, decision making, and save time (Softube, 2021). The controller can be seen in figure 9. The controller can be coupled with a software environment that allows all of the software parameters to be instantly mapped to the hardware controller, this software environment can be seen in figure 10. This can provide a more analogue feeling for the tools used by music producers. The controller is more focused on the mixing tasks of music production process, which could be seen as a more formal part of music production.

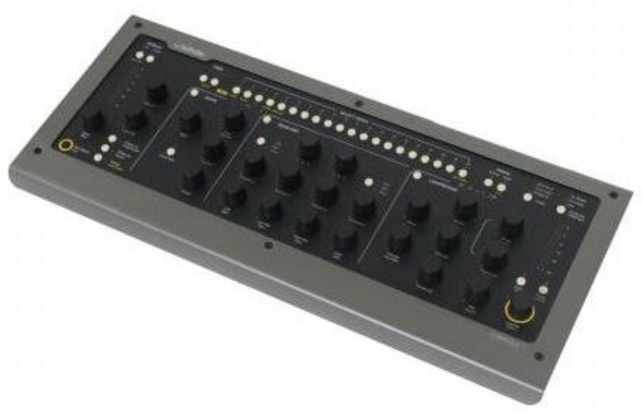


Figure 9 - Softube Console 1 (Baxmusic, 2021)

Competitive analysis

The Console 1 can be used with software from Softube. This provides a hardware-software ecosystem that works very well for people that use these tools. The controller can be seen as a direct competitor since the functionality is similar to that of Modsy. The pros and cons compared to the Modsy controller have been listed below in table 4.

Pros	**Cons**
Very intuitive control due to the fixed controller layout.	Device only works for the Softube environment, only with Softube plugins.

High quality build	Limited amount of control for plugins
	Not all digital parameters have a physical representation.

Table 4 - Competitive analysis Softube Console 1

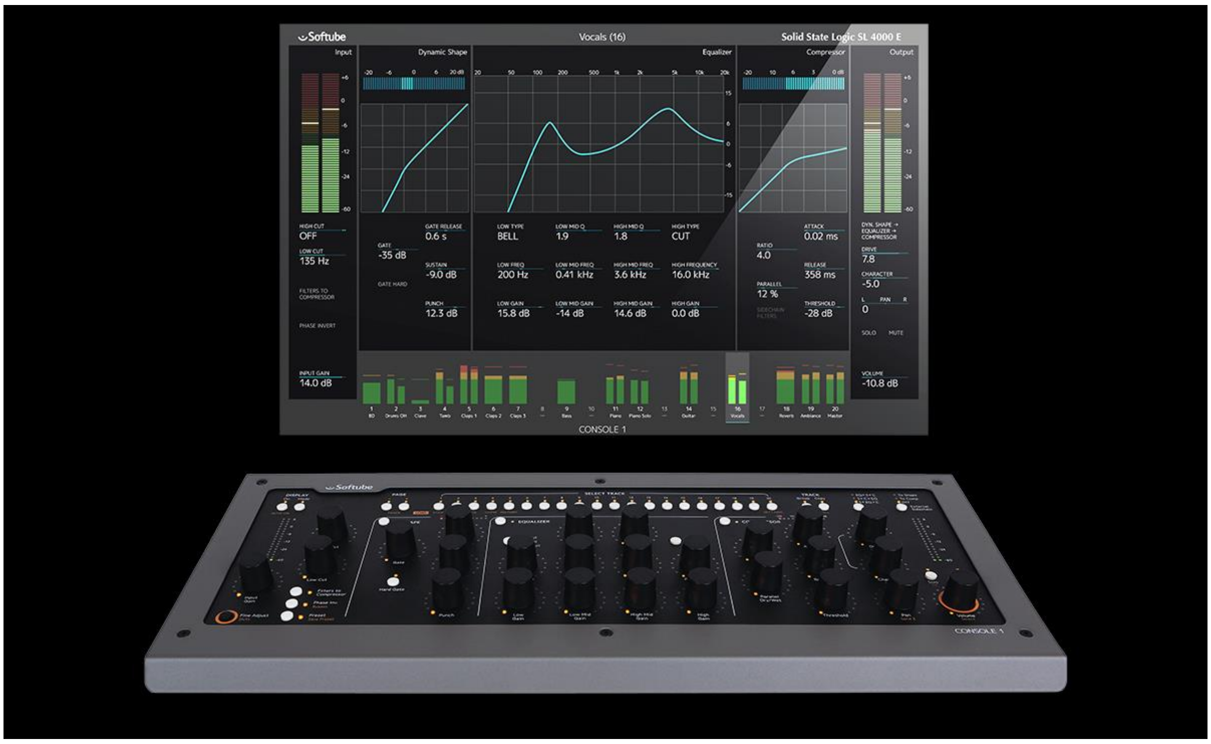


Figure 10 - Console 1 with Softube software (Softube, 2021)

Playfulness analysis

Playfulness analysis was performed for both the form and function of the product and included a general PLEX category analysis as stated in the beginning of this state of the art section. Both are described below.

- **Form**
  The form of the controller is very basic. The controller is not especially focussed on providing playful engagement upon observation. During usage the controller becomes more visually engaging, the LED on the controller will provide direct feedback for the user.
- **Function**
  The functionality of the controller is again related to music production, thus directly related to playful behaviour. However, the control with the Console 1 will be more focussed around

more formal aspects of the music production process. Since the controller will be used more in the mixing and mastering stage of music production and less in the instrumental / creation phase of music production. This means that the usage of the controller will most likely be used in less play based activities.

Analysis with PLEX categories indicates the following most prominent categories:

- **Control**, a major aspect of the controller is the high and direct amount of control that is provided over your environment. However, professional reviews of the controller state that this control is limited since not all functionality can be directly changed from the controller. (White Sea Studio, 2019)
- **Challenge**, there is certain challenge with the use of a hardware for music mixing and mastering which differs in feeling from solely digital music production. This is especially evident in the mixing and mastering stage and this controller suits challenge very well, it creates a more analog environment for the producer which the producer has to get used to.
- **Sensational**, A user will be provided with direct audible feedback upon his interactions with the controller.

Overall for the user group of the Console 1 the perceived playfulness is **moderate**.

MP MIDI

Description

The MP MIDI controller is a product from a start-up in Cyprus. The controller provides direct physical control over digital instruments and effects. These digital instruments and effects can be loaded directly onto the controller which can be seen in figure 11. The controller is made up of a large touch screen with 32 rotary encoders that can be used to control these digital instrument and effects.

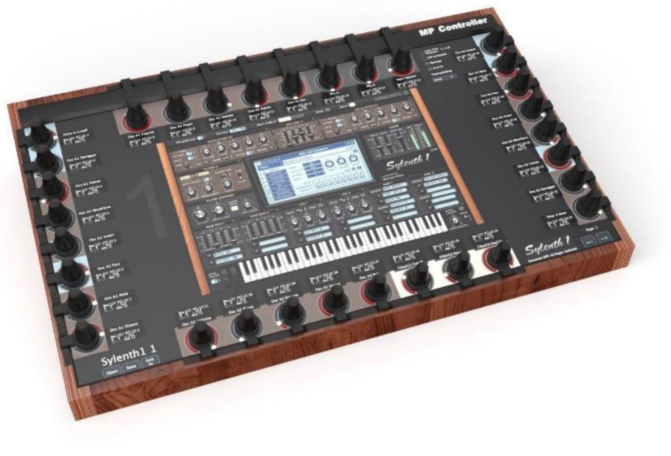


Figure 11 - MP Midi (Interface, 2020)

Competitive analysis

The MP MIDI controller is a direct competitor of Modsy since it provides direct analogue control over different digital instruments and effects. The software allows the usage of different software formats like VST, AU or AAX and can be used with a variety of different DAWs. The touchscreen functionality allows control over every element of a plugin in an instant, which could be a potential advantage over the Modsy. The price point of the controller is slightly higher than that of the Modsy, 720 euros. The pros and cons compared to the Modsy controller have been listed in table 5 below.

Pros	**Cons**
Works for different DAWs	Less of an analogue feeling due to the display and encoders.
Touchscreen for full control	Uses cumbersome solution for direct mapping of software.
Large amount of parameter control	Price is slightly higher than that of the Modsy controller.

Table 5 - Competitive analysis MP MIDI

Playfulness analysis

Playfulness analysis was performed for both the form and function of the product and included a general PLEX category analysis as stated in the beginning of this state of the art section. Both are described below.

- **Form**
  The form of the product is not very playfully engaging. Most elements are static and even with the display turned on there are no elements that can provide an engaging playful experience upon observation.
- **Function**
  The functionality of the controller is very playful. The product allows for direct playful manipulation of parameters with the use of the touch screen or physical knobs. This direct functionality allows a musician to fully focus on an instrument or effect and get fully engaged in its features.

Analysis with PLEX categories indicates the following most prominent categories:

- **Captivation**, the controller can provide a fully immersed experience with an instrument or effect

- **Control**, the controller provides high level of control with the use of both touch and encoder control. This creates an environment where the user feels he can playfully control all elements.
- **Exploration**, the different way of using instrument or effects provides a way to explore its features in an even more extensive way. The controller allows for better exploration of synths and effects inside of your DAW.

Overall for the user group of the MP MIDI the perceived playfulness is **moderate to high**.

2.2.2 - Inspirational products

As stated in the background research, playfulness can be best studied through examples. It could be very beneficial for a designer to assess the playfulness of inspirational products (King & Chang, 2015). This section of the state of the art is used for this purpose, an analysis of playfulness in interesting and inspirational products.

Teenage Engineering OP-1

Description

The OP-1 is a product by Teenage Engineering (Teenage Engineering, 2021). A company that seems to benefit from playful design and interactions in its products. A good example of this is the OP-1 controller, which can be seen in figure 12. This controller is a synthesizer, drum machine and overall music production tool that can be used to create music from the ground up. The product is respected and loved by musicians even though it has a higher price tag (1000 euros).

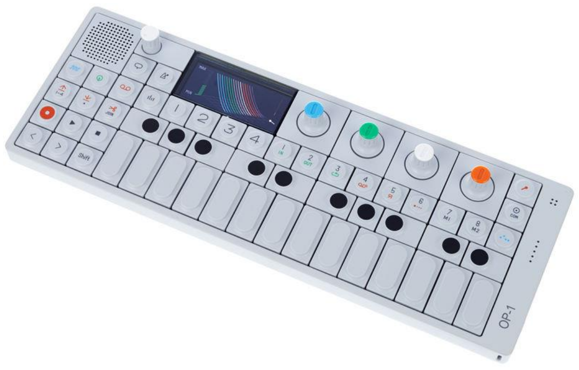


Figure 12 – OP-1 Controller (Thomann, 2021-b)

A particular interesting feature is the interface design of the OP-1 as can be seen to the right in figure 13. There are different visualizations of effect / parameter settings and these have been created in a seemingly ‘playful’ way. An example would be the boxer that can

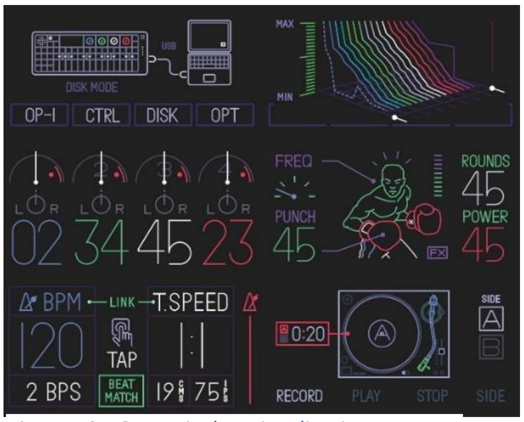


Figure 13 - OP-1 Display Visualizations (Tombolare, 2012)

be seen on the middle right of the image, twisting parameters would increase the punch of the sound and this is made clear through this seemingly playful visualization.

Playfulness analysis

Playfulness analysis was performed for both the form and function of the product and included a general PLEX category analysis as stated in the beginning of this state of the art section. Both are described below.

- **Form**
  The form of the controller is very playfully engaging. The controller creates this visual engagement through the use of different types of colours, buttons and keyboard layout the controller. Next to this, the visualisations throughout the use of different effects, sounds and menus within the controller is highly playful.
- **Function**
  The functionality of the controller is very playful as well. The controller can be used as a stand-alone music production station. This means that users can create sound on the controller, alter these with on board manipulations and effects, and put them into an arrangement. This enables users to quickly ideate and create musical ideas and loops while experimenting with their sounds.

Analysis with PLEX categories indicates the following most prominent categories:

- **Captivation**, Since the controller can be used as a stand-alone music station an artist can completely loose him or herself in the process for a long period of time.
- **Discovery / Exploration**, there is a lot of different functionality to discover and further explore upon discovery. This includes different instruments, effects and ways of song creation.
- **Humour / Fantasy**, the controller is unique in the way that it deals with user feedback for setting changes. This can be seen in the display interface from image 13. Humour and fantasy are used to convey a message, this is unlike many other controllers.
- **Sensational**, with the use of both audible and visible feedback the controller can provide a very sensational experience.

Overall for the user group of the OP-1 the perceived playfulness is **high**.

Moog Minimoog Model D

Description

This analogue synthesizer is one of the most iconic electrical instruments (Moog, n.d.). Many artists can recognize this machine and feel an urge to start playing with its parameters. Even though the design of the controller is very straightforward, as can be seen in figure 14, there are many ways in which the settings of the machine can be changed in other to generate new sounds.

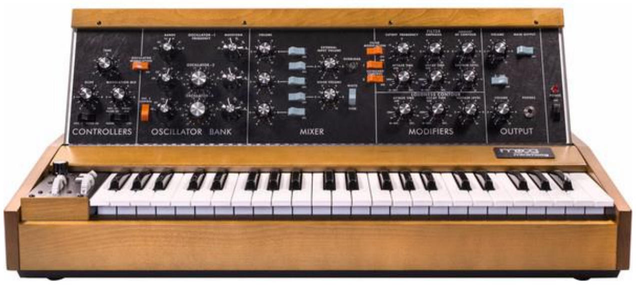


Figure 14 - Moog Minimoog Model D (Music Store, 2021)

What is particularly interesting about the Model D is the fact that the Model D was one of the first portable synthesizers and has been used since the 1970s. This resulted in the Model D being used by a lot of different artists over the years including Herbie Hancock, Kraftwerk, and Funkadelic. The users of the Model D can create the sounds of these artists (Reverb, 2019).

Playfulness analysis

Playfulness analysis was performed for both the form and function of the product and included a general PLEX category analysis as stated in the beginning of this state of the art section. Both are described below.

- **Form**
  The form of the Model D is not particularly playful. Due to the layout of the knobs and design of the controller there is some visual playful engagement into how these button relate, but there are no specific items that are highly visually playfully engaging.
- **Function**
  The functionality of the Model D is very playfully engaging. The machine can be used to create a variety of sounds and by twisting different parameters the relationship to other parameters changes, thus creating a never ending array of sounds. As stated in the description the synthesizer can also be used to create sounds of artists that used the machine in their recorded music. This creates another layer of functionality and could add additional challenge to the use of the product.

Analysis with PLEX categories indicates the following most prominent categories:

- **Captivation**, the Model D can be very captivating during usage. The controller can be used to create a variety of sound and has enough functionality to fully captivate users during usage.
- **Discovery / Exploration**, the Model D has different functionalities that can be discovered during usage, each of these functionalities changes the behaviour of the parameters on the instruments. This creates a wide variety of possibilities and options to explore.
- **Challenge**, as stated in the playful functionality of the Model D, the controller can be used to create sounds of artists that have used the instrument before. This could create the challenge to mimic or discover ways to create these sounds.
- **Thrill**, one element that comes with the territory of old analogue synthesizers is the change of the controller breaking or developing certain quirks during usage. It can provide a certain risk and danger during music production that could add to the playfulness of the experience of using the instrument.

Overall for the user group of the Minimoog the perceived playfulness is **moderate to high**.

LEGO

Description

Every kid has touched a Lego block at some point in their life. The company behind the Lego blocks, the LEGO Group, finds play extremely important and has stated play as a key role in the development of children (LEGO, 2021; Zosh et al, 2017). Lego is commonly understood as being a playful product, so it is interesting to see where this playfulness comes from and find ways to apply it within the design process. An example of a LEGO creation can be seen in figure 15.

Figure 15 – LEGO (IKEA, 2021)

Playfulness analysis

Playfulness analysis was performed for both the form and function of the product and included a general PLEX category analysis as stated in the beginning of this state of the art section. Both are described below.

- **Form**

  The form of Lego is very visually engaging. When gazing upon a pile of Lego blocks one can imagine what can be build or altered to improve or change a Lego creation. The different colours, options and styles allow for many ways one can get visually engaged with the Lego blocks.
- **Function**

  The function of the Lego blocks is highly playful as well. The act of playing with Lego allows for exploration, discovery and changes as we will describe with the use of PLEX categories. The act of building something new from existing blocks and being able to change and swap at any time is very free and playfully engagement.

Analysis with PLEX categories indicates the following most prominent categories:

- **Captivation**, users can get completely captivated in the building process of Lego or the play that can be performed with the Lego.
- **Discovery**, users can keep discovering new ways to use the Legos and different ways to apply these Lego creations.
- **Exploration**, after finding new ways to use the Legos, users can explore different colours, shapes, or types of characters to use for similar projects.
- **Fantasy**, users can get lost in a fantasy Lego world due to the different product lines of Lego. They can create whole worlds of Lego to get lost in for hours. One glance at the Lego world event will provide understanding into the possibility of this (LEGO world, n.d.).

What is very interesting about Lego is the versatility of the product, since there are many different products under the Lego name and they can be used in different ways, the following categories are also of interest

- **Competition**, Lego can be used for competitive purposes. There are different competitions around the world based around Lego (LEGO education, n.d.). There is even a Dutch TV show centred around the building of Lego structures (Endemol Shine Group, 2021).
- **Completion**, there can be a real sense of completion after finishing a Lego build. It depends on the type of build and the user if this is seen as true completion.
- **Humour**, Lego uses humorous pieces and constructions in their products. This can support the playfulness of these products.
- **Simulation**, users can choose to simulate scenarios of daily life, movies or their fantasy worlds. They can buy Lego versions of elements of these world and simulate this environment.

Overall for the user group of Lego the perceived playfulness is **high**.

Blipblox

Description

The Blipbox is a synthesiser designed for children (Playtime Engineering, 2021). The controller is made of plastic and has a variety of differently coloured knobs, handles and triggers to create sounds. The controller can be seen in figure 16. Since the parameter names are not displayed on the product itself it is up to the user to find out what each knob does. Even though the controller is meant for children, a lot of older music producers that want something different end up buying this controller (O'Brien, 2020). This is very interesting behaviour since this would normally not be considered a qualitative good instrument. The playfulness of the controller might play a key part in the acceptance of musicians.

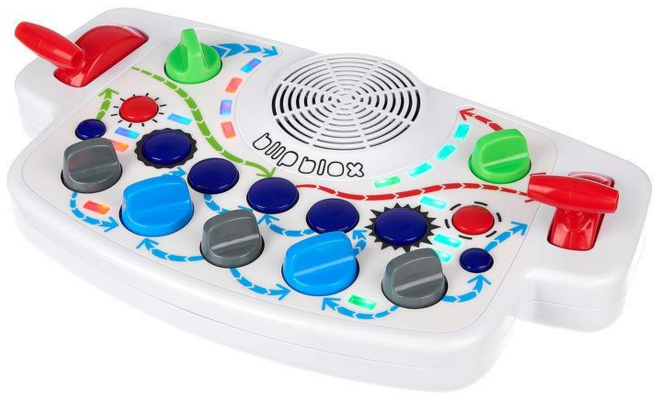


Figure 16 - Blipblox synthesizer (Thomann, 2021-c)

Playfulness analysis

Playfulness analysis was performed for both the form and function of the product and included a general PLEX category analysis as stated in the beginning of this state of the art section. Both are described below.

- **Form**
  The form of the Blipblox is very playful. This is due to the colours, arrows and shape of the device, these elements create some playful visual engagement into the meaning and functionality of the controller.
- **Function**
  The functionality of the controller is highly playful as well. As will be pointed out in the PLEX analysis, the discovery and exploration during usage result in a high level of playful interaction with the device. Users can only hear the change of audio when twisting a knob.

Analysis with PLEX categories indicates the following most prominent categories:

- **Discovery**, this controller has a high amount of discovery upon first usage, this is due to the lack of parameter names and the strange layout of the controller itself. It can take a user quite some time to discover all features of the controller.
- **Exploration**, through usage of the controller the user explores all of the features present on the controller itself and can find new combinations when exploring all parameters.
- **Challenge**, it can be a real challenge to figure out what functionality knobs are linked to. During usage it can be a good challenge to find out more and more what the synth does and remind this for future usage.
- **Sensational**, due to audible feedback, Bright lights and movements the controller is very sensational in usage. Since the feedback might also be unexpected this further strengthens the sensation of using the controller.

Overall for the user group of Blipbox the perceived playfulness is **high**.

2.2.3 - Lessons learned

Different things can be learned from the analysis of both competitors and inspirational products. The main takeaways are listed below.

- **Baseline for music production controllers**
  Some PLEX categories seem to be related to most of the music production controllers. These are Captivation, Discovery and Exploration. This makes sense since this might be at the core of a good music production controller. A good controller may have to captivate a musician in its usage and allow him to discover and explore new elements that might lead him to the production of new music. When designing a music production controller these elements seem to be important in the playfulness of the controller.
- **Design for unique categories**
  Some categories where less frequently identified but provided huge playful value for a variety of the products. These include Humour, Challenge, Fantasy. These categories could for instance be seen in the OP-1 controller. Challenge could be seen in the Moog controller, which is an often overlooked aspect of the appeal of this instrument.
- **Untapped markets**
  Some categories of the PLEX framework where never identified during the analysis of direct competitors of the Modsy, these could be potential categories that can be used in the ideation phase of the Modsy controller to create a unique design. They included: Fellowship, Submission, Subversion, Suffering, Sympathy, Completion, Eroticism. Some of these categories may be very difficult to effectively include in the design of a controller without the disruption of functionality or focus of the product. But these categories do pose unique design aspects and will be taken into account for the ideation phase.
- **Form and function**
  Functionality and form seem to be well reflected in the analysis with PLEX categories. Analysis based on form and function does make some of the categories somewhat more concrete and allows the distinction of precise design features. We can look at the example of the OP-1 controller. Plex categories identifies Humour and Fantasy to be important, the form analysis proposes the Oled Display to be one major visual aspect that contributes to this playfulness. So, thinking about form and function could be useful to make plex categories more concrete and transform these to actual design features. This could be useful in the ideation phase.

Chapter 3 – Methods and Techniques

This chapter describes the methods and techniques that will be used throughout the remainder of the report.

3.1 - Creative Technology Design process

The Creative Technology Design process can help with the development of new and innovatie products, applications and services (Mader, Eggink, 2014). This method describes 4 different phases that can be used during the design process: ideation, specification, realisation, evaluation. The phases follow after eachother and end with a definied set of intermediate results.

- **Ideation**
  The ideation phase establishes new concepts or ideas. This is done through the gathering of user data through, e.g. user interviews or observation, and the creation of concepts, e.g. sketching, storyboarding and prototyping. The ideation phase allows for the diverging and converging of ideas. At the end of the ideation phase one or more preliminary concepts have been created, these concepts form the starting point of the specification phase.
- **Specification**
  The specification phase elaborates and explores the preliminary concepts carried over from the ideation phase. The specification of these concepts relates to the use of different methods and techniques to further establish the concept and make it more concrete in its form and function. At the end of the specification phase the final concept is proposed that can be used in the next phase of the design process.
- **Realisation**
  In the realisation phase the final concept carried over from the specification phase is realised. This includes the creation of hardware components, software development and wiring of electrical components. At the end of the realisation phase the final concept has been realised to a functional and testable prototype.
- **Evaluation**
  In the evaluation phase the prototype is tested. Both functional and non-functional testing could take place during this phase. User testing is most likely to be involved during this phase in order to evaluate and reflect on the realised design.

3.2 - Stakeholder analysis

A stakeholder analysis is the process of identifying the individuals, groups or organisations that have a certain interest or influence within a project (Smith, 2000). Throughout the stakeholder analysis these groups can be describes and have both their influence and interest analysed within the scope of the project. A stakeholder analysis can help to understand the impact that a certain project can have.

3.3 - PACT analysis

PACT analysis is a framework that can help and think about human centred design. The framework allows for the analysis of people carrying out certain activities in context using certain technologies (Nayanathara, 2020). The PACT analysis can be used to derive requirements for a system. In the PACT analysis four elements can be analysed and are listed below.

- **People**
  During analysis of people all possible physical and metal characteristics of the users of the product will be taken into account. Within the scope of this thesis people are explored and explained through user interviews, user research and personas. Personas provide detailed descriptions of the users of a product.
- **Activities**
  Analysis of activities allows for the distinction between different tasks or interactions that a user performs with a system. These tasks can differ in length, complexity, or any other relevant factor.
- **Context**
  The context is related to the situation in which the product will be used. This can for instance describe the environment, social context or organisational context in which the product is used.
- **Technologies**
  The technologies relate to all the technology that is used in the operation of the system. This could relate to certain input devices that are needed such as keyboards or touchpads, or certain output devices like displays or speakers. Next to these two elements the communication of this technology should also be considered.

3.4 - PLEX brainstorm

The PLEX brainstorm technique stems from the Playful Experiences framework (Lucero, Arrasvuori, 2010). The PLEX framework has been discussed during the literature research in chapter 2. The PLEX brainstorming technique can be used to quickly come up with large amounts of playful concepts. The method can be performed by groups of two. This technique uses the PLEX cards, these cards hold the names and visualization of different PLEX categories. An example of these PLEX cards can be seen in figure 17. As discussed in chapter 2, these PLEX categories are related to playful interaction. PLEX brainstorming can be performed in the following way:

1. First, one card is drawn randomly form the deck of PLEX cards and placed face-up on the table. This card is called the Seed card.
2. Then both players pick three extra cards from the deck of PLEX cards.
3. Thereafter, the user which placed the Seed card should explore the product or idea using the PLEX category on the Seed card.
4. Both players listen and consider the cards in their hand and when they feel that they can elaborate on the idea placed on the table they put it on the table. They explain how the initial idea is changed or evolved.
5. The idea is developed further by picking and placing cards from the hands of the participants.
6. After all cards have been used the players can conclude their PLEX brainstorm session.

During the PLEX brainstorm the ideas can be recorded on paper.

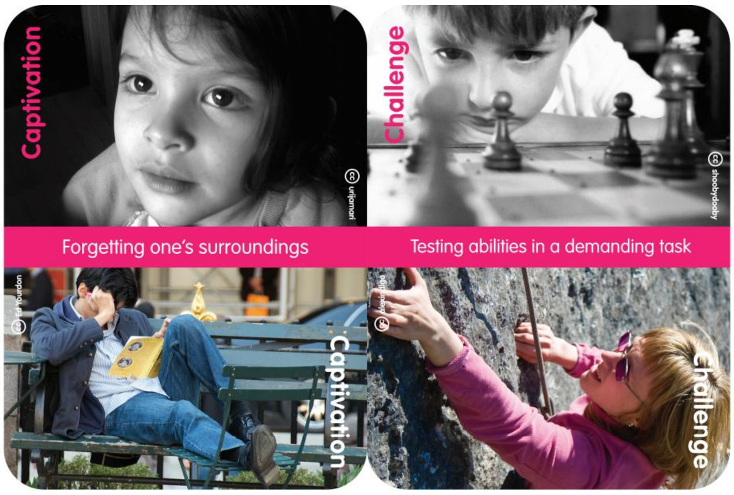

Captivation
Challenge
Forgetting one's surroundings
Testing abilities in a demanding task
Captivation
Challenge

Figure 17 - Example of PLEX cards, captivation and challenge (funkydesignspaces, n.d.)

3.5 - PLEX scenario

The PLEX scenario technique is also related to the PLEX framework. This technique can be used to create more elaborate ideas using the PLEX cards (Lucero, Arrasvuori, 2010). This technique can be performed by two participants. One deck of PLEX cards is used.

1. One pair of participants draw 3 PLEX cards. These cards are placed face up on the table.
2. Thereafter a scenario is formed by the participants using a PLEX scenario template as can be seen in figure 18 below. There are 3 stages to the scenario, one card can trigger the beginning of the scenario, another card the continuation, and the third card should provide the ending to the scenario.

The ideas can be written or sketched on the PLEX scenario template.

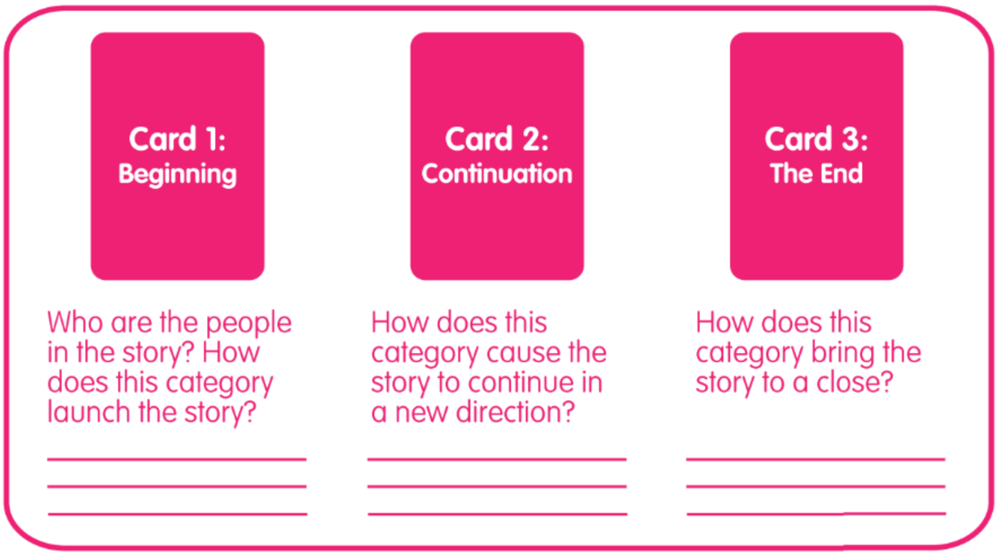

Card 1: Beginning
Card 2: Continuation
Card 3: The End
Who are the people in the story? How does this category launch the story?
How does this category cause the story to continue in a new direction?
How does this category bring the story to a close?

Figure 18 - Plex Scenario cards (funkydesignspaces, n.d.)

3.6 – Mind Mapping

Mind mapping is a technique that can be used to map out different ideas or options in a structured manner (University of Adelaine, 2014). The method is mainly used to analyse associations that certain elements or topics have with each other. Within the scope of this thesis mind mapping will be used to provide structure in a brainstorming exercise. A central theme can be placed in the middle of a page or view and other elements can be connected to this theme using lines or branches.

3.7 - Rapid ideation

Rapid ideation is an brainstorming method that can be performed by a group or individual to quickly come up with a lot of different ideas (Belyh, 2019). During this method a group or individual has to come up with as many ideas as possible within a certain time limit. The rapid ideation performed during this thesis does not include no direct reflection on feasibility. Within the scope of this thesis this method is only used individually.

3.8 - MoSCow Method

The MoSCoW method is a prioritization technique that can be used to manage requirements of a product (ProductPlan, 2021). The method uses four different categories to define difference in priority: Must haves, should haves, could haves, and will not haves. These categories are shortly described below:

3. **Must have**, non-negotiable product needs that are mandatory for the successful realisation of the product.
4. **Should have**, Important needs that are not vital, but can provide significant value to the product.
5. **Could have**, Nice to have initiatives that will have a small impact if left out.
6. **Will not have**, Initiatives that are not a priority for the time frame of this thesis.

The MoSCoW method will be used at the end of the ideation and specification phase to form the requirements of the final product.

3.9 - Sketch ideation

Sketch ideation is a method to discover and explore ideas (Patel, 2020). The method uses sketching on paper, whiteboard or other material to materialize and give form to ideas. The method can be performed by an individual as well as a group.

3.10 - Mood boarding

A mood board is a physical or digital collage of ideas that can help with the ideation of a new product (Canva, 2021). During the method a user can use different images, materials, text or other design elements to ideate about a product. Within the scope of this thesis a digital mood board will be created using a variety of digital images.

3.11 - SUS analysis

The System Usability Scale (SUS) provides a quick way to measure the usability of a system (Usability.gov, 2020). The SUS consist of 10 different questionnaire items with 5 response options. The SUS can be analysed in a specific manner which will result in a SUS score on a scale from 0 to 100. This score then relates to the overall usability of the system. A score above 68 would indicate an above average usability score, a SUS score of above 80.3 would indicate that people are very positive about the usability of the system (Thomas, n.d.).). The SUS questionnaire as used during the evaluation phase of this thesis has been included in the appendix as item A. The SUS score can be calculated in the following way:

- For all odd numbered questions 1 should be subtracted
- For all the even numbered questions their value should be subtracted from 5
- All the new values should be added up and multiplied by 2.5

3.12 - Flow Short Scale (FSS)

The Flow Short Scale has been developed by Rheinberg, Vollmeyer, and Engeser (2003). The Flow Short Scale can be used to analyse flow after completed activities or during experience sampling method based assessments. The Flow Short Scale consists of 13 items, with items 1 – 10 measuring the components of flow experience. Items 11, 12 and 13 measure perceived importance or perceived outcome importance (Engeser, 2012). The Flow Short scale as used during the evaluation phase of this thesis has been included in the appendix as item B.

Chapter 4 – Ideation

This chapter describes the ideation phase of the creative technology process. It will include the analysis of stakeholders, analysis of the Modsy controller, PACT analysis and finally, the actual ideation of more playful Modsy controller concepts. At the end of this chapter the preliminary concept will be described together with a list of preliminary requirements.

4.1 - Stakeholder analysis

This stakeholder analysis will discuss individuals or groups that could have interest or influence in the graduation project.

4.1.1 - Potential users of Modsy

The users of the Modsy controller could have significant influence and interest in this graduation project. The users of Modsy are producers and performers that use a DAW. We can segment these potential users into 2 potential user groups: advanced and professional producers and performers and beginner producers and performers. It is important to segment the user group of Modsy since research shows that advanced and professional producers and performers are most likely to be interested in the product and could benefit most from the Modsy controller (Mathijssen, Driel, Berkenbos, Wintermans, 2020). This resulted in a focus on advanced and professional producers and performers for Modsy its features and design. Both groups are described below with their interest and influence.

Advanced and professional producers and performers

Advanced and professional producers & performers are individuals that have significant skill within music creation or performance with the use of a DAW. These users are familiar with DAW functionality and know the relevant terms when it comes to music production or performance.

Interest	**Influence**
<u>High</u> These users will derive high benefit from the Modsy system, thus the interest in the Modsy controller will most likely be high.	<u>Moderate</u> These users are the target audience of the Modsy. The Modsy design will be fitted to this user group and feedback of the user group will directly be used in design choices. Thus, the influence that these users have is moderate.

Table 6 - Influence / Interest advanced and professional producers and performers

Beginner producers and performers

Beginner producers & performers that use a DAW can be defined as: Individuals who are relatively new to music creation and performance with the use of a DAW and still have a lot to learn in this process.

Interest	**Influence**
<u>Low</u> Since these users will derive less benefit from the Modsy controller it will be most likely that the interest in the controller will be lower.	<u>Low</u> These users are not the direct target audience of Modsy, thus the influence that these users will have is low.

Table 7 - Influence / Interest beginner producers and performers

4.1.2 - Weirdly Wired

Weirdly Wired is the start-up behind the Modsy controller. This group consists of Bram van Driel, Robbert-Jan Berkenbos, and Olivier Mathijssen. Decisions about the Modsy controller are influenced by this team as design problems or company strategy is discussed between team members.

Interest	**Influence**
<u>High</u> Since there is a high interest in successful development of the product there is high interest in the design of the Modsy controller.	<u>High</u> All of the individuals within the Weirdly wired team have the influence to drastically change the design of the Modsy controller. So all have significant influence in this graduation project.

Table 8 - Influence / Interest Weirdly Wired

4.1.3 - DAWs

The Modsy controller is used in combination with a DAW (Digital audio workspace). Without a DAW the Modsy cannot function. This means that the design of a DAW and their actions can have a significant amount of influence on the graduation project. The software structure of this DAW could limit functionality and possible features of the Modsy controller.

Ableton Live

The DAW that is initially of most importance is Ableton Live since the first iteration of Modsy's will only be functional in combination with this DAW. Through the course of the graduation project Ableton Live will be used in combination with the Modsy controller.

Interest	**Influence**
Low There is no direct benefit that the Modsy controller will have for a DAW. However, there could be some interested for the design of the Modsy controller since this could provide benefit for users of a DAW.	Moderate The structure of Ableton Live could influence the development of the Modsy controller, but this does not apply for all aspects of the development. It merely concerns the digital features of the Modsy controller, e.g. the mapping of the Modsy controller to digital instruments and effects. This will be further explained in the product analysis during this ideation chapter.

Table 9 - Influence / Interest DAW

4.1.4 - Music production controller manufacturers

Other music production manufacturers could have influence in the design process. Decisions made by this group could influence the current status of the music production and performance controller market. The most important parties to be considered are the direct competitors of the Modsy controller as described in the state of the art section, namely Native Instruments, Ableton Push, Softube and MP MIDI. To differentiate the product from these controller the Modsy design should differ from that of these controller, to some extent this has to be considered throughout this graduation project.

Interest	**Influence**
Low This group is not very interested in the development phase, but could be interested in the outcome of the project.	Low This group could have influence during the converging of design concepts, however within the scope of this thesis competitive advantage is not the highest priority.

Table 10 - Influence / Interest music production controller manufacturers

4.1.5 - Production partners

For the actual realization of the design into a real-world product it could be important to consider the different production partners needed in this phase. Even a good design could pose potential design problems in a later stage due to production complications. Thus the influence and interest of this group should be assessed.

Interest	**Influence**
<u>Low</u> There is a low interest into project course and outcome. The interest is mostly financial.	<u>Low</u> For the graduation project the influence is low. This could increase when implementing potential design solution for a new product since elements might need to be altered in order to suit production.

Table 11 - Influence / Interest production partners

4.1.6 - Project supervision

Possible decision influencers include Erik Faber and Wouter Eggink. Both are supervisors of this graduation project. They will provide feedback throughout the project and will eventually assess and grade the work. In the course of the graduation project these two individuals can have significant influence in the steps that are taken.

Interest	**Influence**
<u>Moderate</u> There is no high interested in the outcome of the graduation project itself, this interest is in correct execution of the project steps. This creates an moderate element of interest in the project.	<u>Moderate</u> Both supervisors have the ability to change the course of the graduation project. This is mostly through feedback and advice on the course of the graduation project.

Table 12 - Influence / Interest project supervision

4.2 - Modsy product analysis

To ideate about a potentially more playful Modsy controller this controller should be analysed on essential features and potential requirements for design improvement. The users and context of the Modsy controller will be analysed using a PACT analysis, however specific product features will be analysed beforehand to get a sense of the goals, purpose and essential features of the controller. This will help with the development of design requirements.

4.2.1 - Essential Modsy aspects

As stated in the first chapter, the Modsy controller will be able to provide an analogue feeling to digital tools. The controller can be used to instantly take full control over different digital instruments or effects. There are different elements in the current design of Modsy that form the essential functionality of the product, these elements have been analysed together with Bram van Driel and Robbert-Jan Berkenbos of the Weirdly Wired team and are listed below:

- **32 parameter control elements**
  The current Modsy controller has 32 parameter control elements. These control elements include both potentiometers and buttons. They are used to take control over different digital instruments and effects and bring the analogue feeling to these digital devices. A choice has been made for direct control over 32 parameter since this gives users the same amount of control they would have over a real analog instrument or effect. This has been derived from synthesizers like the JUNO-106 and Moog Minimoog as described in the state of the art section. The control elements of the Modsy can be seen in figure 19 on the next page.
- **Parameter feedback**
  It is essential for a user to receive feedback of the currently mapped instrument or effect. When providing analog control it is essential that the user can instantly recognize what parameter he or she is controlling and how the value changes when he or she alters a parameter. This parameter feedback will be provided through little displays above each parameter, these displays can also be seen in figure 19 on the next page.
- **Modsy plugin**
  There is a digital element that is always used together with the hardware controller. This is the Modsy plugin. The Modsy plugin can be loaded inside of Ableton live and allows for the creation and editing of mappings that are used to link the controller to different instruments and effects inside of Ableton. This plugin can be seen on the next page in figure 20.

- **Ableton control**

  Lastly, an essential element of the Modsy controller is the Ableton control. Ableton control relates to the triggering of certain functionality within the Ableton Live environment such as: start/stop, records, and quantize. The controller has an encoder, 2 displays, and 6 buttons that can be used for Ableton control. These elements can be seen at the top of the controller in figure 19.

All these features are essentials for the proper functioning of the Modsy system and should be included in a future design as well.

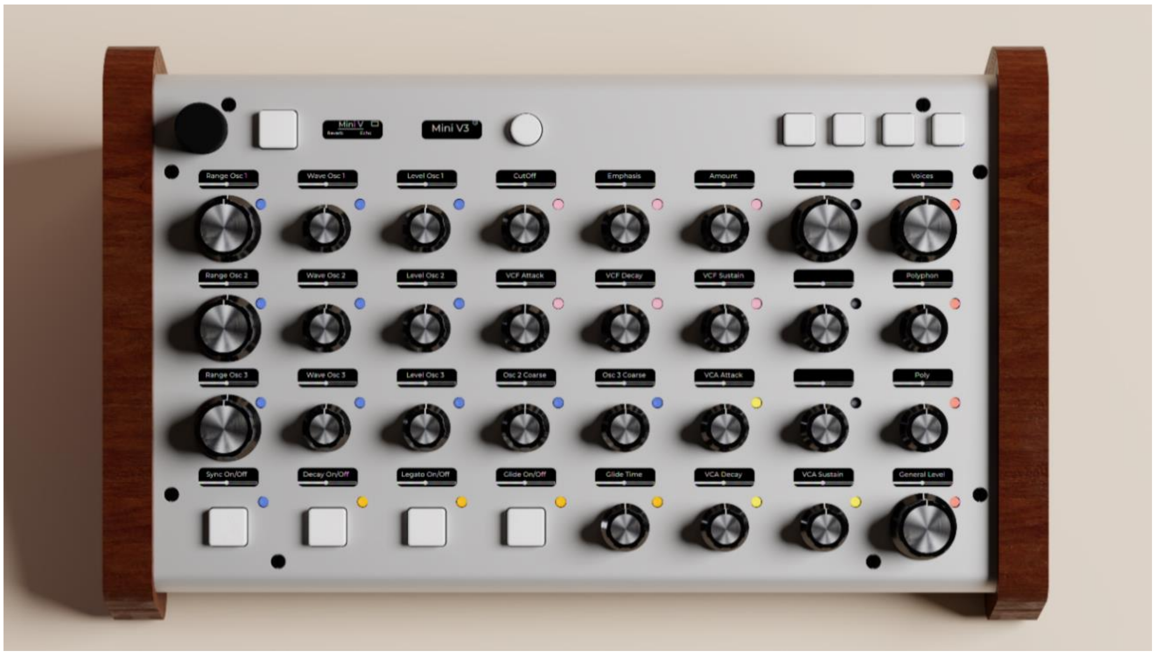


Figure 20 - Current Modsy controller design

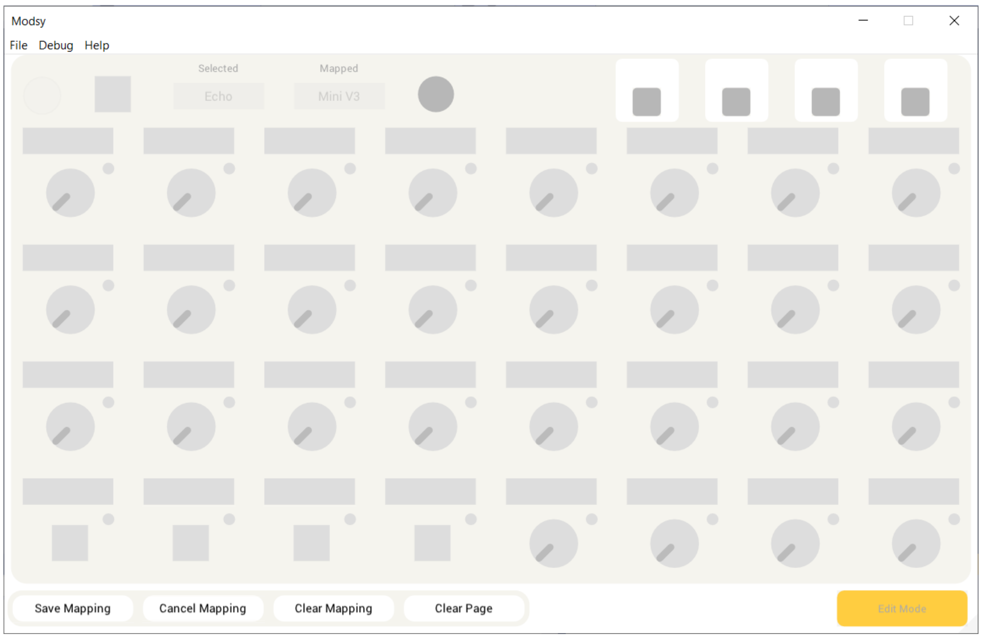


Figure 19 - Modsy software plugin

4.3 - PACT analysis

A PACT analysis was performed to analyse the context in which the Modsy controller will be used. As explained in chapter 3, the PACT-analysis is a useful framework to think about a design problem by considering people, activities, contexts and technologies. The purpose of such an analysis in the context of this thesis is to derive requirements for the design of a Modsy controller. This analysis has been performed together with members of the Weirdly Wired team.

4.3.1 - People

The people aspect of the PACT analysis will analyse the characteristics of the users of Modsy. Both user interviews and personas will be used to analyse Modsy users.

User interviews

At an earlier stage of development Weirdly Wired has performed research into user requirements (Mathijssen, Driel, Berkenbos, Wintermans, 2020). Different user interviews and tests were conducted to explore the desires of the potential customers of Modsy. This resulted in an assessment of the main benefit of the Modsy system, the following aspects were stated to be most important:

- A way to explore digital instruments and effects
- A way to gain more creative control
- A way to gain more expression
- A way to automate parameters, this will be explained in the activity section of the PACT analysis.

All these potential benefits are essential for the Modsy system and should be present in a future design of Modsy. In the context of a playful Modsy controller 3 unstructured user interviews were conducted with users familiar with the Modsy controller. During these interviews the goal was to identify user requirements that would be important if design improvements were to be made related to the Modsy system. The following two requirements could be derived:

- **The parameter control should be non-disruptive for the music production process.**
  This requirements relates to the way that parameters can be altered on the Modsy controller. This should not disrupt the music in unexpected ways. When a music producers alters a

parameter he or she would like to know what is changing and this should be adjustable if the change is disruptive to the music that is created.

- **The controller should keep the ability to fully focus on music production process.**
  This means that when a user is using the Modsy system the controller should still allow complete focus on the music production process. Within the scope of this thesis that is relevant since some playful solutions might make the music production process more complex or distracting and this is, at least not always, what a music producer or performer wants.

Personas

Within the scope of this thesis personas are used to explore and describe the different characteristics of a variety of Modsy users. In total four personas have been described. These personas were created together with the other members of the Weirdly Wired team.

Boris, male, professional producer, 28 years old, Amsterdam, DJ/Producer

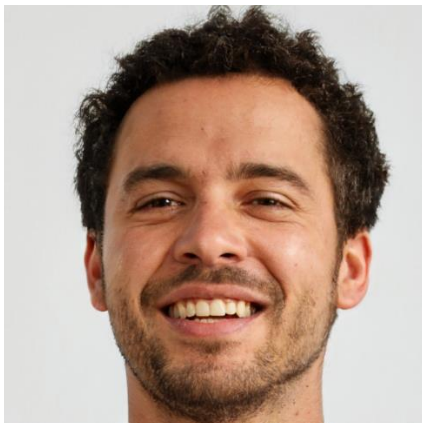


Figure 21 - Image of Boris

Boris is a 28 year old DJ and producer. He has an apartment in Amsterdam together with this girlfriend. Boris is signed to a electronic music label based in Amsterdam and can utilize a studio owned by this label. An impression of Boris can be seen in figure 21, created with thispersondoesnotexist.com (Karras et al. (2019). Next to this Boris has different music production equipment in his apartment. This equipment consists of good speakers, a synthesizer and a keyboard. In the studio he has different synthesizers, drum-computers and mastering equipment at his disposal. Boris has been making music for a long time and has always been a digital music producer. Musical concepts are mostly created in his apartment. Finalizing these concepts, so the mixing and mastering stage of his music production, is performed in the studio of this label. Boris his main drivers for music production are income, enjoyment, and emotional release. Boris has signed a deal with his label which states that he has to release at least 1 album each year, and he been able to deliver on this promise. Boris is very active on social media, mostly for the promotion of this own music. Furthermore Boris has a large social circle within Amsterdam with many friends active in the music industry, this results in a lot of local collaboration and support.

Josh, male, advanced producer, 32 years old, Berlin, software developer

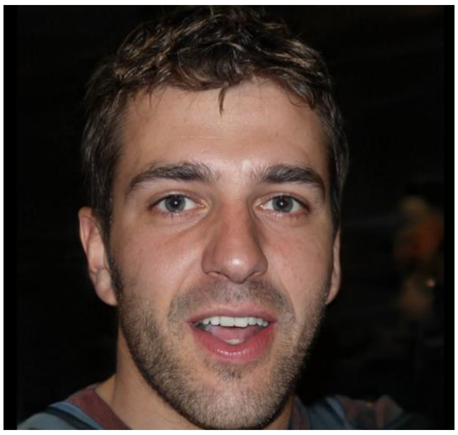


Figure 22 - Image of Josh

Josh is 32 years old and has been producing music since his childhood years. An impression of Josh can be seen in figure 22, created with thispersondoesnotexist.com (Karras et al. (2019). He started out playing guitar, but later moved on to digital music production in Ableton. He has studied computer science at the university of Delft and now works a full-time job as a software developer in Berlin. He lives alone in a moderate apartment in the city of Berlin. Josh mostly partakes in music production during evenings and in the weekend. Josh his main drivers for music production are enjoyment, emotional release and personal development. Josh only occasionally releases his music on Soundcloud or his Spotify page. Josh is a member of a Facebook group where he talks with other like-minded musicians. Within this group he is not very active, but will engage occasionally. Noah can finance his musical activities well, since he has a stable income. His setup is minimalistic but of good quality, with a proper mixer, good speakers, a MIDI keyboard, and a few instruments.

Nina, female, advanced producer, 21 years old, Utrecht, Sound Design student.

Figure 23 - Image of Nina

Nina is a student sound design at the HKU in Utrecht. In this program she specialises in sound for video content. She works part time in a sound design agency and makes her own music as well. An impression of Nina can be seen in figure 23, created with thispersondoesnotexist.com (Karras et al. (2019). She has a lot of contact with young producers and DJ's and is trying to build a portfolio for herself, which includes being active on social media and doing promotional work. Since she wants to distinguish herself from the crowd she is always looking for creative ways to make new music. This includes utilizing old instruments or sampling old records. Her budget is not big which means she is very particular about what she invests her money in.

Robert, male, advanced producer and performer, 56 years old, Hogeveen, Electrical engineer

Figure 24 - Image of Robert

Robert is 56 years old and has been performing music for a long time. An impression of Robert can be seen in figure 24, created with thispersondoesnotexist.com (Karras et al. (2019). He started out in bands around the age of 16 and has been making music ever since. This passion has progressed parallel to this professional career in engineering. The music Robert produces is mostly psychedelic rock and synth wave music. Roberts main drivers for music production and performance are emotional release and enjoyment. He does make a small amount of money from his music performances, but this is not a main drive. Robert records music with a band and uses different musical instrument in this process. Robert has always been very interested in the mixing and mastering of the audio and has a small mixing studio in his garden. This studio holds a variety of guitars, effects, mixing board, and some synthesizers. Robert mostly uses Ableton for the finalizing of his recordings and occasionally uses software instruments or effects in the process.

4.3.2 - Activities

Analysis of activities provides the essential functionality that the Modsy will have. These activities should all be well supported in the design of a more playful Modsy controller.

Sound design

Sound design is an activity in which new sounds are being created using different instruments or effects. Within the context of Modsy this will be the creation of new music by altering the parameters of a digital instrument or effect. This activity could occur often during the music production process. It is an active activity and is more often performed individually, but this can also be done collectively. Sound design can take place for any music related project, from dance music to soundscapes and audio visual projects.

Sound automation

Within the scope of this thesis sound automation relates to the occurrence of automated parameter changes by the DAW of a user (Bawiec, 2018). What this means is that the DAW will change certain parameters of instruments of effects over the course of a song. The way that the DAW changes these parameters can be recorded or edited by a user themselves. A form of automation is the slow increase of a cut-off filter or volume throughout a musical piece. An example of this automation, a volume drop, can be seen in figure 25 below. The Modsy controller can be used during the musical arrangement to create these automations in the music.

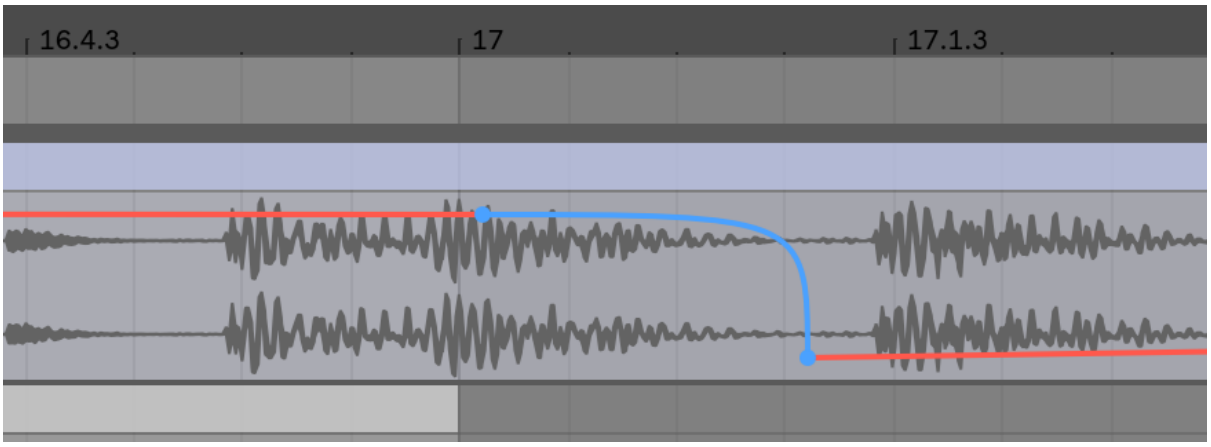


Figure 25 - Automation in Ableton Live

Ableton navigation and control

Ableton navigation and control refers to the function buttons and mapping section on the Modsy controller. The function buttons can be used to trigger certain functions inside of Ableton Live. This can be start/stop, record, quantize, and other relevant functionality. The encoder can be used to navigate between your digital instruments and effects within Ableton and thus enabling to fully control you DAW environment straight from the controllers interface.

These three elements from the core functionality of Modsy controller and must be kept intact for a good functioning of the Modsy controller.

4.3.3 - Context

There are two main environments in which the Modsy controller can be used: a Studio environment and Live environment.

Studio environment

A studio environment in the context of the Modsy controller refers to a music production studio environment that can be used for the recording, creation, mixing or mastering of audio. The type of music studio can differ based on the type of music producer of performer. Some studios are full of analogue machines while other studios are more simple and have only good speakers, a computer system and some mixing equipment. Beginner musicians are more likely to have a simple studio while more advanced and professional musicians are more likely to have a more advanced studio. A studio environment can relate to any place that is used for creation of audio, this also includes bedroom and office spaces with only a desk, chair and laptop.

Live performance

The Modsy can also be used during a Live performance, in this environment the Modsy controller can be used together with other instruments or effects to provide a real performance. Such a performance is normally performed in front of an audience. The user has different needs in such a context. Following user interviews performed by Mathijssen, Driel, Berkenbos, Wintermans (2020) musicians that perform live need simple layout, a sturdy fixed controller, and clear control visible in low light. This is due to the fast pace, rowdy and dark environment in which live performance takes place.

The primary focus of the Modsy controller is on the studio environment since the Modsy controller seems to have highest potential in this context (Mathijssen, Driel, Berkenbos, Wintermans, 2020). However, the live environment is still important to consider since artists have indicated the potential during live performance.

4.3.4 - Technologies

The primary technology that is needed for the functioning of the Modsy controller is a computer with Ableton Live. This computer should have basic peripherals for normal operation of Ableton Live. The Modsy controller itself has different input and output types that create its interaction with the user.

Input

These are the main input technologies that are used within the Modsy system.

- **Potentiometer**
  These will be used to control the parameters of different digital instruments or effects.

- **Buttons**
    - Function buttons

      The function buttons are used for triggering different functions within the Ableton Live environment like start/stop, quantise, record etc.
    - Parameter buttons

      The parameter buttons can be coupled to different digital instruments and effects like the potentiometers.
    - Mapping / Page button

      These buttons are located on the top left of the controller and are used for Mapping specific functions.
- **Encoder**

  The encoder can be used to select different instrument or effects within Ableton Live.
- **Mouse and keyboard**

  The user can also select and control Modsy functionality straight from the Modsy plugin on his computer.

Output

These are the main output technologies that are used within the Modsy system.

- **34 OLED displays**
    - Above potentiometers and parameter buttons

      Above each potentiometer and parameter button will be a display that will tell a user what this parameter is mapped to and display the current status of this parameter. This could be the parameter level or setting.
    - For mapped and selected instrument or effect

      Two displays are located at the top of the controller which will display the instrument or effect name that the controller is mapped to and the selected instrument or effect. This selected device can be changed using the encoder, the display will update automatically.
- **Digital interface – The Modsy plugin**

  The Modsy controller has a software interface within Ableton Live that is used to the creation and manipulation of mappings. These mappings are what map the controller to certain digital instruments or effects.

Communication

- **MIDI communication**
  The controller will communicate to the computer using MIDI protocol over USB.

Content

- **Updated content**
  The content displayed on the controller itself is embedded software and is not likely to be updated. The content within the software plugin is more likely to receive updates in content and graphical user interface. This will be both updates graphical user interface and digital features.

4.4 - Playfulness Ideation

This section of the Ideation phase will discuss the actual ideation of more playful concepts regarding the Modsy controller. According to literature research the PLEX framework is well-suited to be used in ideation of concepts related to playfulness. Thus, the PLEX categories will serve as the driving force behind this ideation phase. The order of the described ideation methods describes the order of events during this ideation phase and is structured in two separate diverging and converging sections. At the end of the ideation phase a preliminary concept was created that was carried over into the specification phase.

4.4.1 - Group brainstorm #1 - Diverging

A group brainstorm was chosen as the first Ideation method to be used since group ideation can lead creative solutions that look at a problem from different perspectives (George, 2007). For this brainstorming session PLEX brainstorming and PLEX scenario methods were used as described in chapter 3 of this report. In total two group session has been performed using these two methods, one with the Weirdly Wired team, and one with potential customers. Both groups group sessions have been performed in different stages of the ideation phase.

The first group brainstorming was performed with the Weirdly Wired team and had the purpose to identify the potential of the different PLEX methods and categories within the context of the design of the Modsy controller. The Weirdly Wired team was chosen for this group brainstorm since this group understands the controller the best and can more easily understand potential features and application of the Modsy controller using different categories.

First PLEX brainstorming was performed. This method, as described in chapter 3, can be used to come up with different playful concepts using PLEX categories. The execution of this method can be seen in figure 26. The outcome of this method was reported on paper. Since the Weirdly Wired team has 3 team members a slight alteration of the PLEX brainstorming method was performed. This alteration used 3 participants of which each picked 3 PLEX cards, however only 6 cards could be added to the Seed card to remain same level of depth in ideated concept. An example of a concept created during this PLEX brainstorm can be seen in table 13 below:

Seed card	Expression
Added PLEX cards	Control, Captivation, Submission, Competition, Sympathy, Thrill
Description	The music made with the Modsy is directly visualized in a VR setting. One can completely be emerged in his musical experience since the changes made with the use of the Modsy controller are directly visualized in this environment. This VR experience allows other users to join and potentially compete with each other. Other people have the possibility of tuning in and listening to these musical creation where they can show how they feel about it. The worse you perform the worse your visualization is.

Table 13 - Example of concept created to PLEX brainstorming

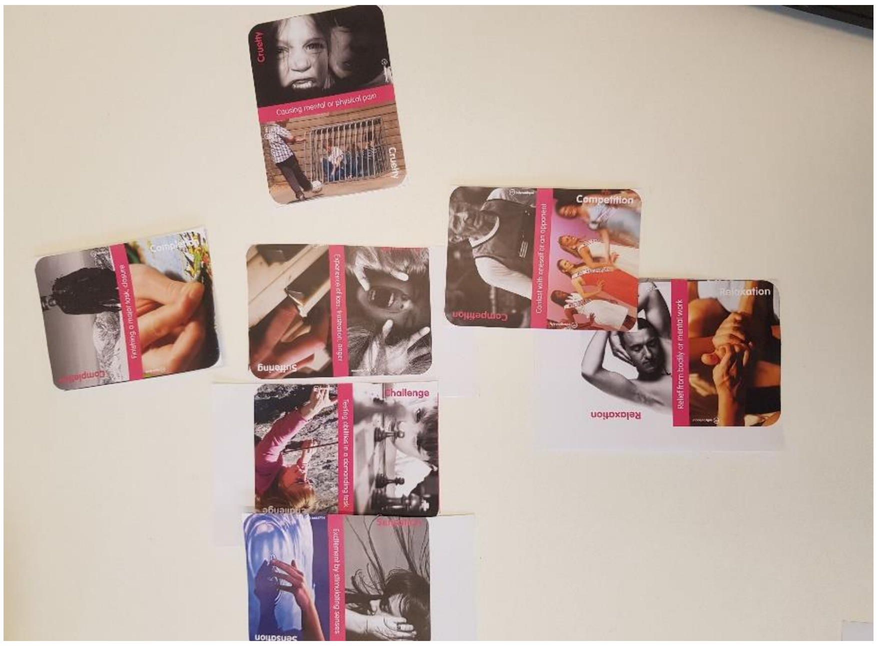


Figure 26 – Execution of PLEX brainstorm method

Next, the PLEX scenario method was performed. This method results in more complete product scenarios using 3 plex categories. Again, 3 participants were included in this methods who could collectively come up with a scenario. One of these created scenarios can be seen in figure 27 below, the storyline was documented digitally. An example of a concept created through PLEX scenario forming can be read in table 14.

Beginning	Nurture	You want to improve and develop yourself by optimizing you mappings and getting better at the control created through these mappings.
Continuation	Simulation	Through the creation and optimizing of mappings you create a digital studio where you can see all of you gear like you would in a real analog studio.
Ending	Sensation	You see visually what you have made and is standing in your studio of mappings. This can be made in very sensational way, as a real experience.

Table 14 - Example of concept created through PLEX scenario forming

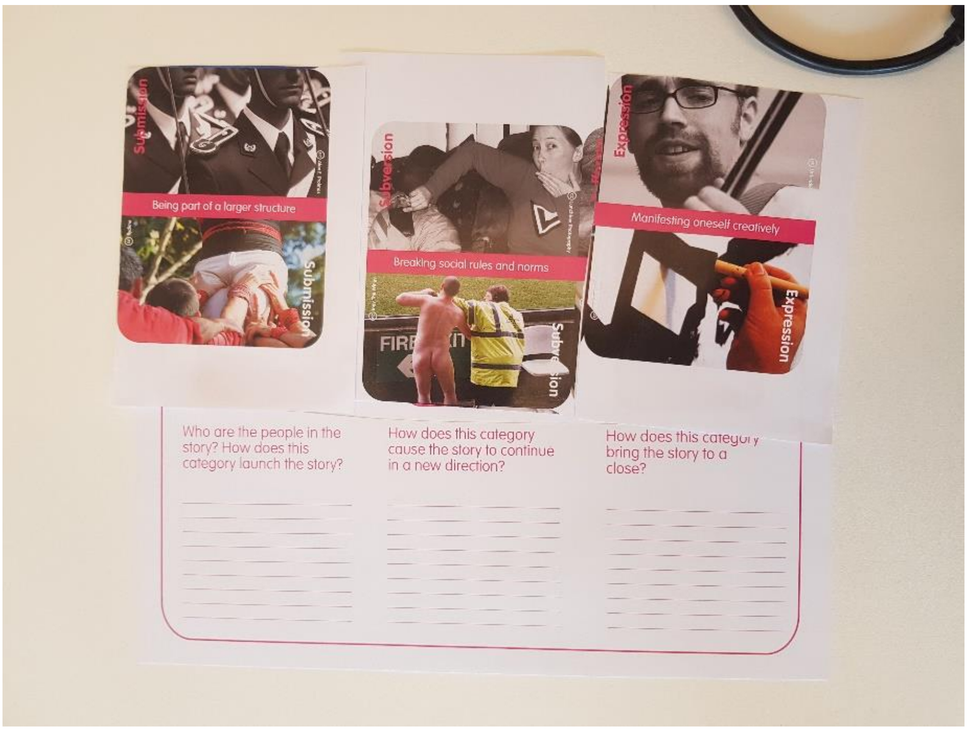

Submission
Being part of a larger structure
Submission
Subversion
Breaking social rules and norms
Subversion
Expression
Manifesting oneself creatively
Expression
Who are the people in the story? How does this category launch the story?
How does this category cause the story to continue in a new direction?
How does this category bring the story to a close?

Figure 27 – Execution of PLEX scenario method

Conclusion group brainstorm

The outcome of the sessions was a variety of concepts that varied highly in feasibility and acceptability. It seemed that there was a need for a more structured and focussed ideation while using the PLEX framework. The creative potential of the use of PLEX methods could be seen, but the effectiveness varied highly throughout the brainstorm.

4.4.2 - Mind mapping - Diverging

The group brainstorm identified the usefulness of some categories but showed certain disruption in concepts as well. This might be due to the nature of these categories and the mismatch in the purpose of the Modsy controller. To further explore these categories and their influence on concepts a mind mapping exercise was used as described in chapter 3, this was performed individually.

Mind mapping was used to ideate about each category of PLEX framework and potential concepts that would derive from these categories. A part of the resulting mind map can be seen in the figure 28 on the next page. The full mind map can be viewed in the appendix, item C.

Mind mapping process

The procedure used during the mind mapping exercise was very straightforward. This is described below.

1. Place main theme (Modsy controller) and all PLEX categories.
2. Place potential sub-categories of these categories on green cards.
   This could for instance be spectacular visuals and spectacular audio for the category of sensation.
3. Place practical ideas of these categories in purple.
4. Possibly add additional comments in pink.

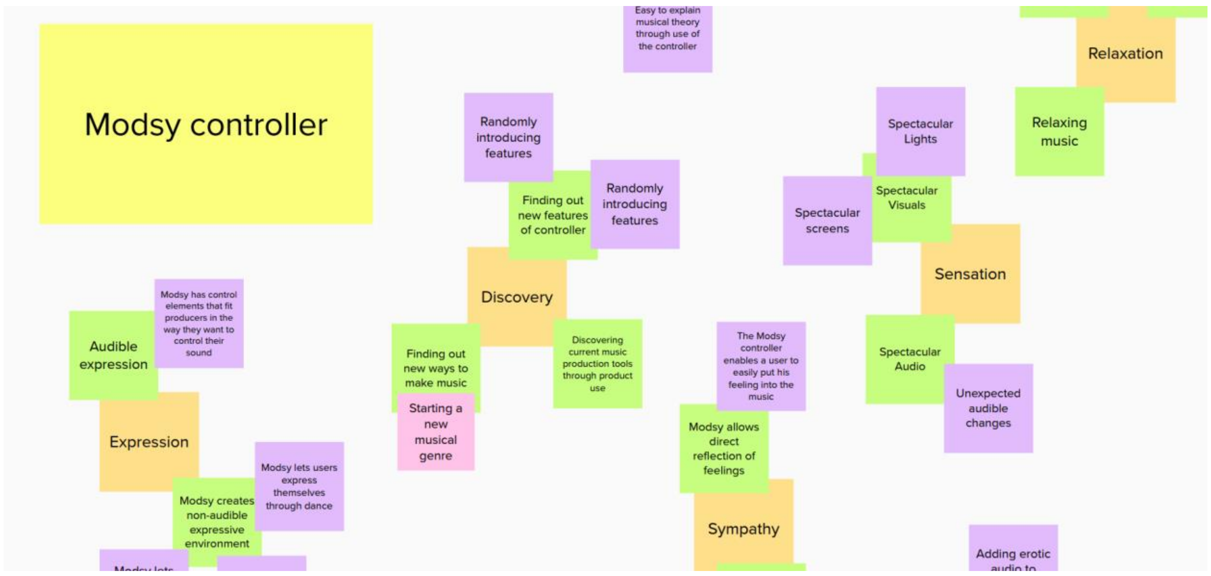

Modsy controller
Easy to explain musical theory through use of the controller
Randomly introducing features
Finding out new features of controller
Randomly introducing features
Discovery
Finding out new ways to make music
Starting a new musical genre
Discovering current music production tools through product use
Spectacular Lights
Spectacular Visuals
Spectacular screens
Relaxation
Relaxing music
Sensation
Spectacular Audio
Unexpected audible changes
The Modsy controller enables a user to easily put his feeling into the music
Modsy allows direct reflection of feelings
Sympathy
Adding erotic
Audible expression
Modsy has control elements that fit producers in the way they want to control their sound
Expression
Modsy lets users express themselves through dance
Modsy creates non-audible expressive environment

Figure 28 - Part of mind map Modsy with PLEX categories

Conclusion mind mapping

The mind mapping exercise helped to further identify the potential of certain categories. Some categories failed to bring forward feasible concepts, this will be further discussed in the next section. Some concepts that spark interest during this mind mapping exercise were concepts related to category humour and fantasy, the use of humorous audio or visuals during the music production process could be very feasible and original for a music production controller. The use of fantasy aspects could also provide an original take on a music production controller design.

4.4.3 - Assessment of PLEX categories - Converging

Through the first brainstorm and mind mapping exercise the potential of the use of PLEX categories in ideation phase could be assessed. There were certain categories which seem to result in interesting concepts and other categories which seem to be rather disruptive. These disruptive categories had a tendency to negatively influence ideas during the PLEX brainstorming and scenario methods. These categories also did not pose great potential following the individual mind mapping exercise and could sometimes be seen as questionable regarding Modsy's overall goal and vision. Taking into account the nature of the PLEX framework and its application this is no surprise, the framework was not primarily focussed on improving product design. So, for the purpose of focussed ideation the framework could be evaluated.

Disruptive categories

First, the more disruptive categories have to be addressed. These should not be seen as disruptive for the playfulness of Modsy but more so disruptive for the purpose and vision of Modsy. As described earlier in the product analysis, there is a very clear emphasis on certain features and vision for the product. These disruptive categories seemed to undermine the vision that Weirdly Wired had for the product and are thus less suited for a future design.

Cruelty, Suffering, Thrill, Eroticism, Subversion

Concepts created through, or altered by, these categories seem to be conflicting with the overall goal and purpose of Modsy. The goal and purpose of Modsy is to create joy, expression and creativity. Adding erotic elements might improve the playfulness in a way, but this could send very confusing signals to the users of Modsy.

Completion, Competition

These two categories are conflicting with the vision of Weirdly Wired on music production. Even though these categories could potentially include playfulness Weirdly wired does not wat to make its product competitive based. It could be interesting to play with the element of competition or completion for oneself, this is why the category Challenge is still labelled as an Interesting category. However, a focus on completion or competition will most likely not be the best approach.

Nurture

Concepts created with this category resulted in features that improved learning or skill. A thesis by Robbert-Jan Berkenbos, which is executed in parallel to this thesis, covers these aspects already (Berkenbos, 2021). Thus, it was less interesting within the scope of this thesis to use the element of Nurture.

Interesting categories

Interesting categories include categories that bring forward interesting concepts that would be well suited in a playful enhancement of the Modsy controller.

Discovery, Exploration

Literature research shows these elements to be important factors for music production controllers. The individual brainstorm confirms that these could be well implemented within the Modsy system. However, these elements are already at the core of standard Modsy functionality. A user gets to play with and discover his digital instruments and effects.

Captivation, Expression, Sympathy, Challenge, Sensation, Control

All of the above mentioned categories provided interesting concepts that could be applicable for the Modsy system. These categories are grouped since they provide features that can be seen in different music production controllers already. Some improved sense of captivation, expression or challenge has been ways in which controllers have been designed. It could still be interesting to further explore, but might not be as unique of a solution.

Fantasy, Humour, Simulation, Fellowship, Submission

These categories not only provided interesting concepts, but also differentiated from current solutions seen in different music production controllers. A playful solution resulting from these categories could be a new unique selling point of the Modsy controller and truly set it apart from the competition.

Conclusion converging

What we can state from this converging exercise is that we could potentially exclude certain categories from the PLEX brainstorm and scenario since these seem to reduce the feasibility and application of the created concepts. These categories are listed as the disruptive categories. The categories that create more interesting concepts can be further explored in future brainstorms and ideation with special focus on the categories of Fantasy, Humour, Simulation, Fellowship, Submission since these could provide a new unique selling point for the Modsy controller. These categories will be the focus of future brainstorming and concept creation sessions.

4.4.4 - Group brainstorm #2 - Diverging

The second brainstorm was also executed with the use of PLEX brainstorming and PLEX scenario method, only now with reduced amount of PLEX categories. The following PLEX categories were included: Discovery, Exploration, Captivation, Expression, Sympathy, Challenge, Sensation, Control, Fantasy, Humour, Simulation, Fellowship, Submission as a result of the converging exercise that was performed. The PLEX brainstorm was executed in an alternative fashion, only 4 PLEX cards could be used to form a concept, this resulted in somewhat more focussed ideas and seemed to positively influence concept creation.

Both PLEX brainstorming and scenario forming showed interesting results, the created concepts seemed to result in more feasible concepts that were in-line with Modsy's overall goal and vision. An example of a created solution during the brainstorm is the following:

There is a small creature living inside of the controller that has to be maintained and changes throughout your music production process. Whatever music you make this influence this creature in the controller. The creature can also begin to alter parameter and take a life on its own.

4.4.5 - Rapid ideation - Diverging

Individual rapid ideation was performed to explore all categories that could create unique playful solution for the Modsy controller. As stated these categories are: Fantasy, Humour, Simulation, Fellowship, Submission. Since these categories were not seen in many other music production controller it could be very interesting if these categories are used in a playful solution. To give an idea of the concepts created during this rapid ideation phase some examples are listed in table 15 below.

Category	**Description**
Fantasy	Controller that can be made custom like a Warhammer figure
Fantasy	Transformer controller that can be transformed to an action figure and alter the sound during play behaviour.
Humour	A user can enter a CD into the controller which will alter the control of the controller.
Humour	There is an animal or person living in the controller that has to be maintained and kept happy (like a Tamagotchi)
Simulation	The controller can be wind up before usage to simulate the power it consumes.
Fellowship / Submission	Controller can be used to communicate to other elements. This can be other Modsy controllers, other instruments, or to an audience.
Fellowship / Submission	The controller reacts to its environment. This could be a response to the sound created, the elements that are in the room, or the other instruments that are connected.

Table 15 - Concepts created during rapid ideation

4.4.6 - Individual grouping - Converging

The concepts created during the group brainstorm and individual session could be reduced to a selection that were most feasible to pursue. This reduction was performed individually and based on feasibility within the context of this thesis, the overall vision and goals of Weirdly Wired and user acceptance based on the user information gathered through PACT analysis in section 4.3 of this chapter. The following concepts remained:

- **The game controller**
  A controller that allows users to simultaneously play a certain game during usage, for instance a classic game of pong. The knobs, buttons and displays could function as input and feedback for this game. This game can be played during the music production process resulting in more creative usage of parameters. This concept is mostly related to the categories humour, fantasy, and challenge of the PLEX framework.
- **The Warhammer controller**
  A fully customizable controller that can be shaped to the specific fantasy of the controller, complete with backstory, artifacts and additional items. The controller can be put together by users themselves. This concept is mostly related to the fantasy category of the PLEX framework.
- **Feel the room controller**
  The controller will respond to its environment. This can be a response to the music that is created, to the usage of other Instruments, or other Modsy controllers. This is related to the PLEX categories submission and fellowship.
- **The musical Tamagotchi controller**
  Lastly, there is the concept of some entity being stuck inside of the controller. This entity can develop itself by the way that the controller is being used. If a music producers uses the controller in a certain way this will affect the entity living inside of the controller. The entity can also alter certain parameters randomly if it feels the need to do this. This concept is related to the fantasy and humour categories of the PLEX framework.

4.4.7 - Weirdly Wired / User feedback - Converging

The 4 concepts were discussed with two different stakeholder groups: the Weirdly Wired team and 2 potential users of the Modsy controller. The following feedback was received about the concepts:

Concept	Feedback
The game controller / Musician Tamagotchi	After feedback it seemed like the game controller and musical Tamagotchi concept could be combined into one concept where there are external influences that alter the Modsy controller parameters, this could be games, some virus or an unknown entity. At the core this

	seemed to be what created an interesting playful interaction while also increasing creativity in the music production process.
Warhammer controller	The Warhammer concept seems to be playful in form but the interaction with the controller lacks in playfulness, after verbal explanation this concept was perceived as less playful as the other concepts.
Feel the room controller	The feel the room controller was stated to be an interesting concept, especially the factor of collaboration was seen as interesting in the feel the room concept. However, it might not be the case that two Modsy's are seen together very often so it was somewhat unclear how extensive the interaction could be with only one Modsy. Next to this, using communication to device other than Modsy controllers might not be feasible.

Table 16 - Feedback on final set of concepts

The feedback was used to decide on a final concept direction. As can be seen in the feedback section the game controller / musical Tamagotchi seemed to have most prospect. This concept was chosen for last section of ideation to define the final concept.

4.5 – Defining the final concept

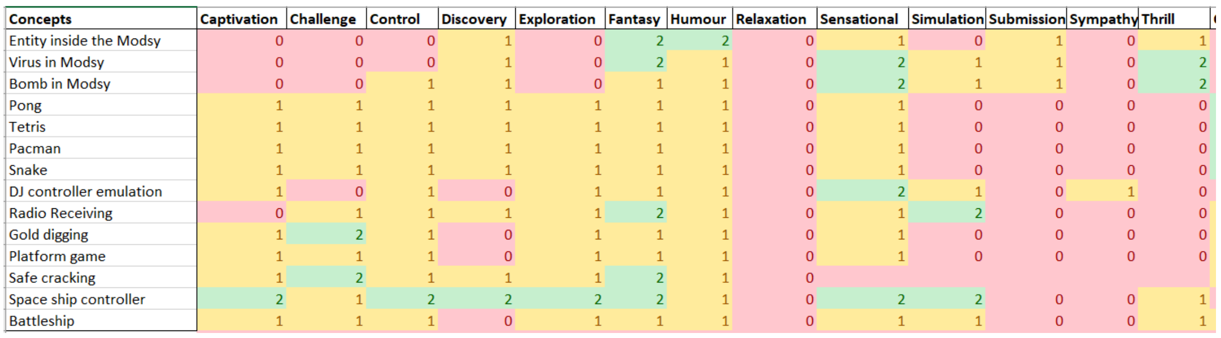
From the game controller, musical Tamagotchi direction several different possible sub-concepts could be ideated for implementation. These sub-concepts then related to the type of game or interaction a user will have with the Modsy controller. Individual rapid brainstorming was used to explore possible sub-concepts. Each ideated sub-concept is mentioned in table 19 with a short description, it should be noted that these are still conceptual and leave some room for interpretation.

Sub-concept	**Description**
Entity inside the Modsy	An unknown entity lives inside the Modsy controller and can manipulate / control parameters. The user can interact with this entity in some way. This can be compared to a Tamagotchi.
Virus	There is a virus that takes over the Modsy controller and changes parameters / settings.
Bomb	There is a bomb that goes off and changes the parameters.

Pong	Classic game of Pong using Modsy for feedback / controls
Tetris	Classic game of Tetris using Modsy for feedback / controls
Pacman	Classic game of Pacman using Modsy for feedback / controls
Snake	Classic game of Snake using Modsy for feedback / controls
DJ controller emulation	The controller emulates a DJ controller that uses the Modsy controls
Radio receiving	The controller turns into an old radio that can be used to tune into certain frequencies and communicate with other entities or possibly controllers.
Gold digging game	The Modsy controller can be used for a classic gold digging game. This will include some sort of grappling hook that is used to grab gold from parts of the controller.
Platform game	A platform game that includes the running to a certain point avoiding obstacles etc. This can be compared to Mario or Sonic like games.
Safe cracking	The Modsy controller turns into a safe that has to be cracked, this can be done by twisting certain parameters of the Modsy controller.
Spaceship controller	The Modsy can be used to control a spaceship and navigate around obstacles like blocks of rubble. Also certain alarms can go off that a user has to take care of. This can be compared to concepts like space-invaders and among us.
Battleship	The Modsy controller can be used to play a game of battleship.

Table 17 - Possible sub-concepts preliminary concept

One of these concepts one had to be chosen for specification. This should be the concept that best serves the factor of playfulness. In the context of playfulness we should consider how well each concept serves the PLEX categories. In this way it could be derived which concept had most potential for playful interaction. The analysis was performed using Excel where all concepts were rated per PLEX category. All PLEX categories were included in this analysis, since all of the current concepts could be feasible and it is interesting to see where there was most playful potential. The concepts were rated from a 0 for no relevance, 1 for some relevance, and 2 for high relevance. This analysis was performed by myself with own interpretation of these concepts. A part of the resulting table can be seen in figure 29 below. The full excel table has been included in the appendix as item D.



Concepts	Captivation	Challenge	Control	Discovery	Exploration	Fantasy	Humour	Relaxation	Sensational	Simulation	Submission	Sympathy	Thrill
Entity inside the Modsy	0	0	0	1	0	2	2	0	1	0	1	0	1
Virus in Modsy	0	0	0	1	0	2	1	0	2	1	1	0	2
Bomb in Modsy	0	0	1	1	0	1	1	0	2	1	1	0	2
Pong	1	1	1	1	1	1	1	0	1	0	0	0	0
Tetris	1	1	1	1	1	1	1	0	1	0	0	0	0
Pacman	1	1	1	1	1	1	1	0	1	0	0	0	0
Snake	1	1	1	1	1	1	1	0	1	0	0	0	0
DJ controller emulation	1	0	1	0	1	1	1	0	2	1	0	1	0
Radio Receiving	0	1	1	1	1	2	1	0	1	2	0	0	0
Gold digging	1	2	1	0	1	1	1	0	1	0	0	0	0
Platform game	1	1	1	0	1	1	1	0	1	0	0	0	0
Safe cracking	1	2	1	1	1	2	1	0					
Space ship controller	2	1	2	2	2	2	1	0	2	2	0	0	1
Battleship	1	1	1	0	1	1	1	0	1	1	0	0	1

Figure 29 - Part of the PLEX analysis for sub-concepts

After this PLEX analysis the space ship control game was seen as potentially most playful. This was mainly due to elements of exploration, fantasy and captivation, but many other categories showed high potential as well. This concept was also very flexible, it could be combined with concepts like the virus or bomb concepts, since there could be occasion's that the spaceship gets hit or gets infected by a virus. Thus the spaceship control concept will be chosen as current concept and further specified in the next chapter.

4.6 - Preliminary concept

The preliminary concept is a spaceship game that can be played during the music production process. A user can interact with this spaceship game with the same Modsy parameters that are used to control an instrument or effect. This means that while playing the game a user is simultaneously altering the sound. This should provide a playful experience according to the following PLEX categories: Exploration, Fantasy, Captivation and Discovery. This concept will be further specified in the next chapter.

4.7 - Use scenario

To clarify how a user of the Modsy controller could interact with the current preliminary concept a short use scenario will be described including one of the personas mentioned in section 4.3.1 of this ideation chapter. Table 17 provides an overview of the use case, table 18 describes the actual interaction with the current preliminary concept, this is somewhat abstract since the concept has not been specified yet.

Use case 1	Interaction with more playful Modsy controller
Actor	Boris, from persona section PACT analysis, 4.3.1
Use Case overview	Boris is using the Modsy controller in his standard musical production process. He is creating a song on his computer in his home studio.
Subject Area	Home studio in apartment.
Trigger	Boris is in need of something that will spark his musical creativity.
Precondition 1	Modsy is connected to a digital instrument within Ableton Live.

Table 18 - Overview of use case

Description	This scenario describes a standard interaction with the more playful Modsy controller. The Modsy controller is connected to a digital instrument within Ableton Live.
1	Boris is controlling a digital instrument or effect with the Modsy controller and presses a button that starts playful interaction.
2	The controller then starts the playful interaction. The Modsy controller turns into a spaceship game. Different Modsy elements change to game elements.
3	Boris sees enemy spaceships and aliens flying towards him, and locates the controls to avoid these elements or destroy them. These elements can be seen on the displays of the Modsy controller or another external display.
4	Boris uses a variety of the Modsy control elements to take control over the spaceship and make sure that the ship stays alive, all the while the controls are connected to a digital instrument in Ableton which means that his control changes the sound as well.
5	When Boris is satisfied with his playful interaction and the sound that has been created through this interaction, or the game reaches its logical conclusion, Boris can return to this normal music production process.

Table 19 - Use case of preliminary concept

4.8 - Preliminary requirements

Different requirements could be derived from the ideation phase. These are the result of user analysis and research, product analysis, and ideation about a more playful Modsy controller. These were structured using the MoSCow method into a list of preliminary requirements split between non-functional and functional requirements.

Non-Functional

Must have

- The controller must be more playful in interaction compared to the current Modsy controller.
- The controller must be more playful in form compared to the current Modsy controller.
- The controller must be able to provide an analogue feeling
- The controller must be able to provide the same, or more expression during music production process compared to the current Modsy controller.
- The controller must be able to provide the same, or more creative control during music production process compared to the current Modsy controller.
- The controller must be the same or more fun to use compared to the current Modsy controller.

Should have

- The control of parameters should be non-disruptive for the music production process.
- The controller should keep the ability to fully focus on music production process
- The controller should be able to be used by all potential users of the Modsy.

Could have

No could haves were derived from this ideation phase.

Won't have

No won't haves were derived from this ideation phase.

Functional

Must have

- The controller must be able to control 32 parameters.
- The controller must be able to provide feedback on the names of currently mapped parameters.
- The controller must be able to provide feedback on the value of currently mapped parameters.
- The controller must allow parameters to be automated.
- The controller must be able to be used in a studio environment.

Should have

- The controller should be able to be used during a Live performance
- The controller should be easy to transport.

Could have

No could haves were derived from this ideation phase

Won't have

- The controller won't use a different way of parameter control compared to the current Modsy controller.

Chapter 5 – Specification

This specification phase will define the preliminary concept in both its form and function. The specification phase will be structured in the following way: First, there will be some general ideation about the form and function through sketching and mood board creation. Thereafter a brainstorm was performed that look at different possible mechanics for the spaceship game, which are then placed in a storyboard to provide overview of the concept. Next, the current Modsy system is analysed and the implementation of the concept is discussed. Lastly, the way in which the concept will interact with the parameters of the Modsy system will be addressed.

5.1 - Spaceship control

To ideate about potential implementation of this spaceship game concept, sketch based ideation was used. This method showed different ways in which the concept could be implemented and helps to ideate about the potential form of the concept. A sketch from this exercise can be seen in image 30.

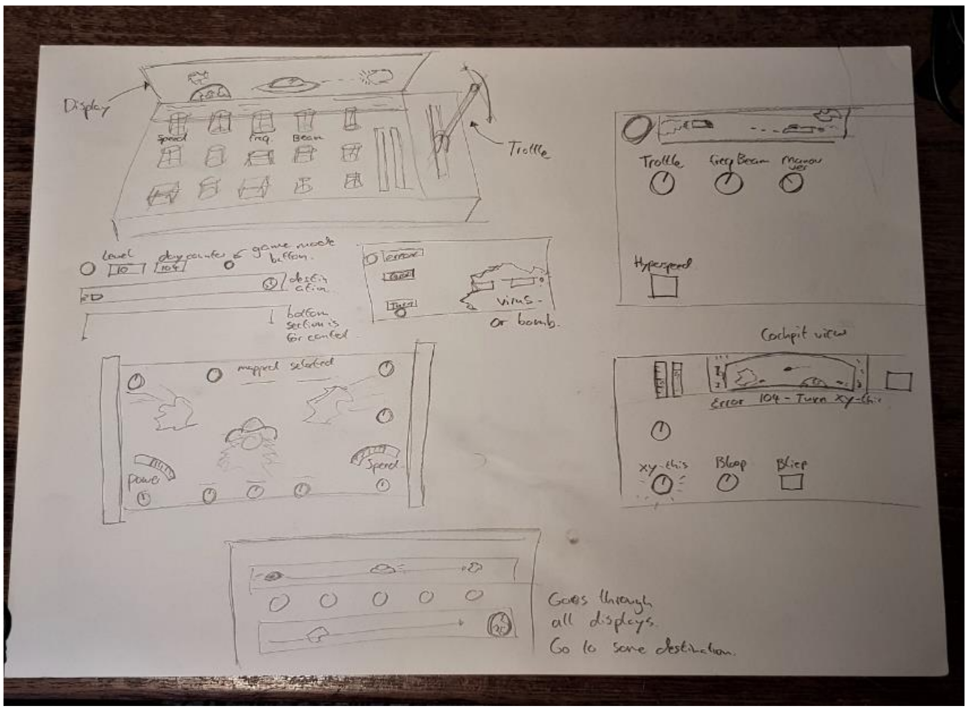


Figure 30 - Sketch ideation

Next to this, a Mood board was created which was used to conceptualize and communicate the possible form of the Modsy controller. This mood board can be viewed in figure 31. With special attention to the bottom right of the mood board, an mission control table for children, very playful in form since it invites an observer to investigate the meaning of each of these elements in the mission control table. This is a explorative and fantasy-rich interaction, which playfully engages the viewer.

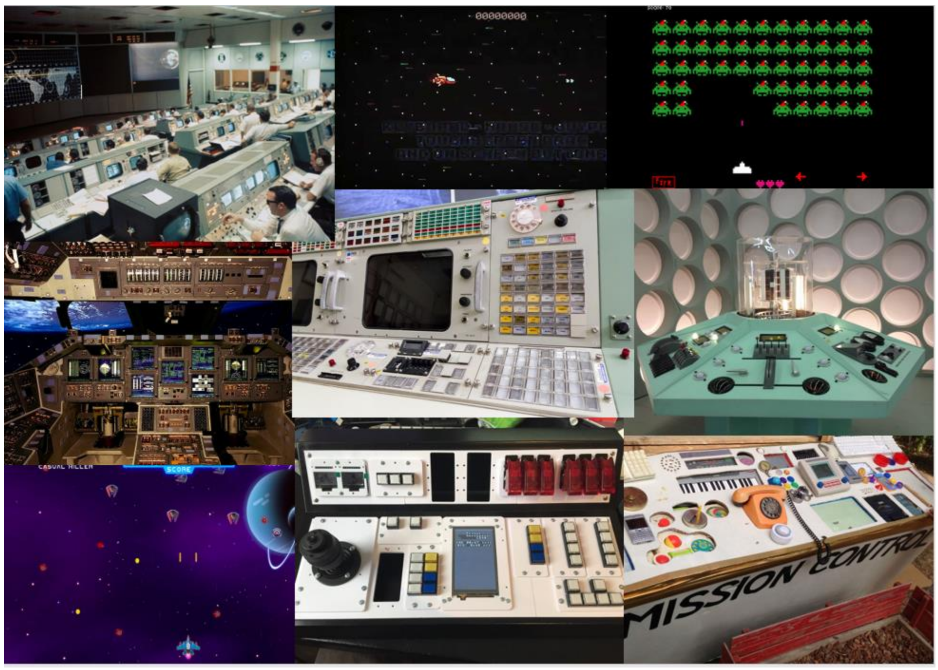


Figure 31 - Mood board spaceship game

5.2 - Game mechanics ideation

For the specific game mechanics of the spaceship game an individual rapid ideation and group brainstorm was performed. The group brainstorm was performed with the Weirdly Wired team. The goal was to identify different feasible game mechanics that could be included in the spaceship game.

The following mechanics resulted from this brainstorm: Manoeuvre spaceship, shoot gun spaceship, point gun of spaceship, manoeuvre around enemy ships, manoeuvre around asteroids, powerups, activate a shield, point the shield, avoid floating aliens, completion of mission, virus that takes over spaceship control, reload action, XY-control for manoeuvres, fix the ship, message to other ships, end point that the ship reaches (end planet), Indication of day or mission that you are on.

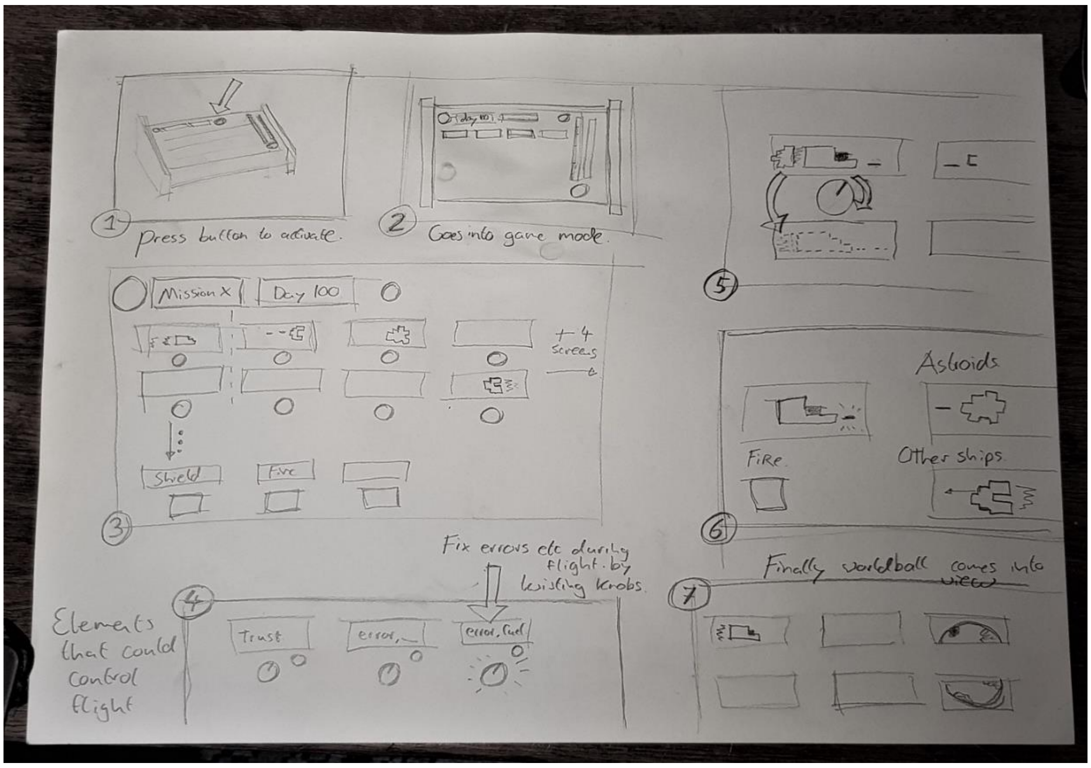
Within the scope of this thesis a limited amount of game mechanics could be included and tested. Next to this, there is danger in adding too many game mechanics which could potentially take over the music production process. It might distract users from the actual purpose of the Modsy controller and as stated in the requirements: The controller should keep the ability to fully focus on music production process. Thus, to test the potential of the spaceship concept a limited amount of functionality will be used:

- Manoeuvre ship
- Shoot gun
- Activate shield
- Avoid asteroids
- Avoid enemy ships
- Reload action
- Completion of mission
- Fix errors

These game mechanics will be addressed in the realisation phase.

5.3 - Storyboarding

After game mechanics had been specified a storyboarding exercise was used to place all mechanics in logical order and see how they could interact with each other. In this storyboard a simple and straightforward implementation of the concept with the current Modsy controller can be seen. This allowed ideation about a feasible implementation of the concept. This storyboard can be seen in figure 32, the next page explains the different elements in the storyboard.



1
Press button to activate.
2
Goes into game mode.
Mission X
Day 100
+4 screens
Shield
Fire
3
4
Fix errors etc during flight by twisting knobs.
Elements that could control flight
Trust
error,_
error, fuel
5
Asteroids
Fire
Other ships
6
7
Finally worldball comes into view

Figure 32 - Storyboarding of spaceship concept

The descriptions listed below are related to the illustrations that can be seen in the storyboard. The numbers relate to the numbers in the storyboard.

1. A user can press a button which will initiate the playful interaction.
2. User enters the game mode. The spaceship and elements of the game can be viewed on the Modsy controller.
3. This illustration shows a potential implementation of the spaceship interaction. The screens of the Modsy controller have been used as feedback of the game.
4. Different elements of the Modsy controller can relate to different spaceship functionality, such as error fixing or trusting the ship forward as can be seen in this illustration.
5. Parameters of the Modsy controller can be used for spaceship control. This could relate to the twisting of a potentiometer to manoeuvre the spaceship.
6. This illustration shows potential elements that the game could include, e.g. friendly spaceship, enemy spaceships and asteroids. It also shows a fire button that the user can use to shoot at the enemy elements.
7. This illustrations shows a world that comes into view. This will mark the end of the game interaction and adds an element of completion to the entire interaction.

5.4 - Modsy system

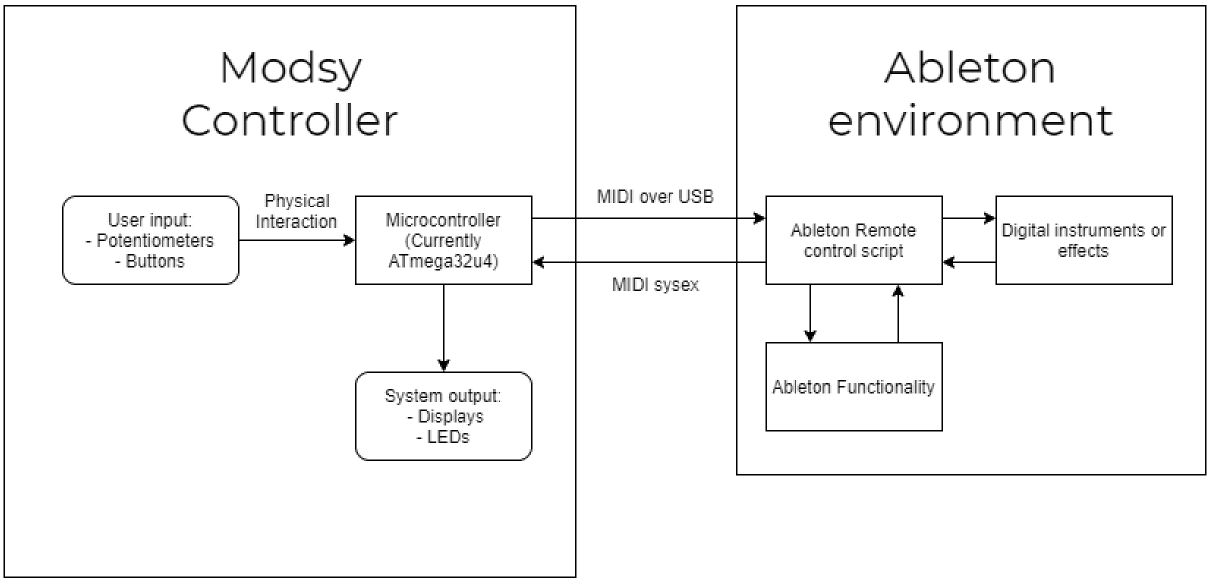
For the feasibility and realization of this concept it is good to consider the current Modsy system and analyse what should be changed in order to implement the current concept. Changes to the design of the current Modsy hardware could weaken the feasibility and application of the concept. However, if a different system architecture could significantly improve the playfulness of the Modsy controller this is important to consider.

System analysis

The current system of the Modsy controller consists of 2 main elements: The hardware controller compatible with Arduino environment, and the Ableton environment with Python scripts and VST plugin that connect the Modsy controller to the Ableton Live environment. An overview of this system can be seen in image 33 on the next page.

Currently, the Modsy controller has 2 input types that enable control over the system: potentiometers and buttons, these are connected to an ATmega32u4 controller using a selection circuit with demultiplexing chips (74HC238D) and switching elements (74LVC1G66GV), these elements can be seen in figure 34. The Modsy controller also has 2 output types: 34 0,91 inch OLED displays using SSD1306 IC controllers and addressable RGB LEDs. These elements can also be seen in figure 34. The Modsy controller uses MIDI protocol over USB to communicate back and forward with the Ableton environment. It can receive information about the Ableton environment through MIDI Sysex messages.

The Ableton environment has 2 essential Modsy software elements: A remote control script running in the back-end of Ableton and a Modsy VST plugin which can be used for user interaction. As can be seen in figure 35 the Remote control script is the brain of the operation and this talks to the Modsy controller. The Modsy VST plugin has a graphical user interface that communicates to this Remote control script, but also creates a way to easily map to third party devices, this will not be explained in more detail since this is not relevant within the scope of this thesis.



Modsy Controller
User input: - Potentiometers - Buttons
Physical Interaction
Microcontroller (Currently ATmega32u4)
System output: - Displays - LEDs
MIDI over USB
MIDI sysex
Ableton environment
Ableton Remote control script
Digital instruments or effects
Ableton Functionality

Figure 33 - Overview of Modsy system architecture

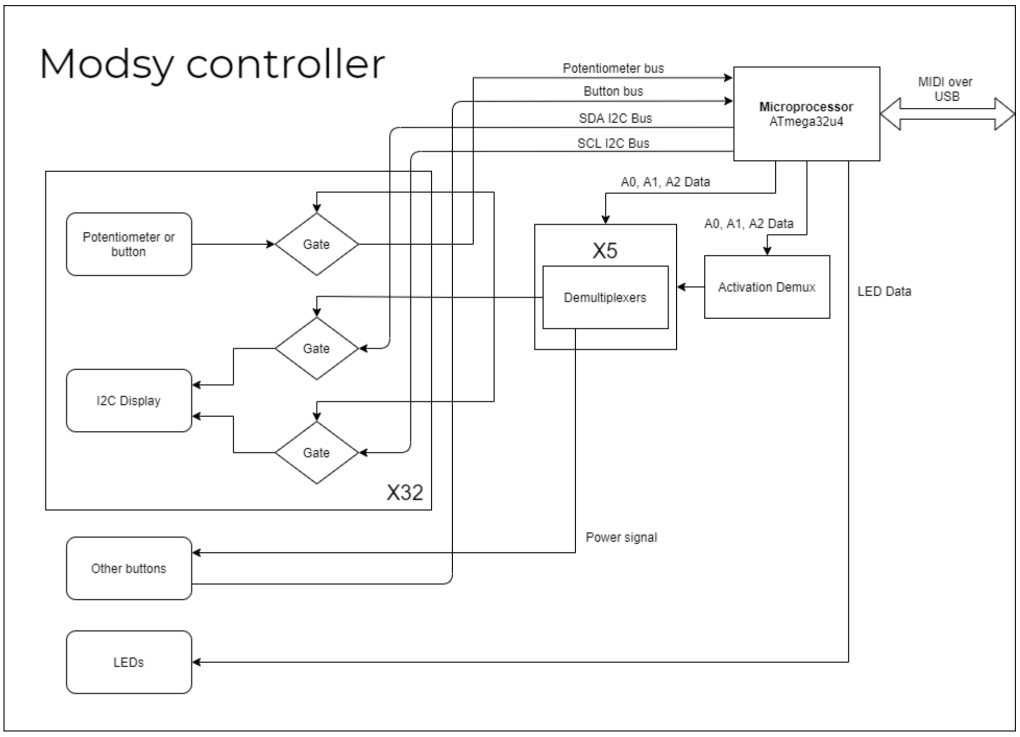


Figure 34 - Overview of Modsy controller architecture

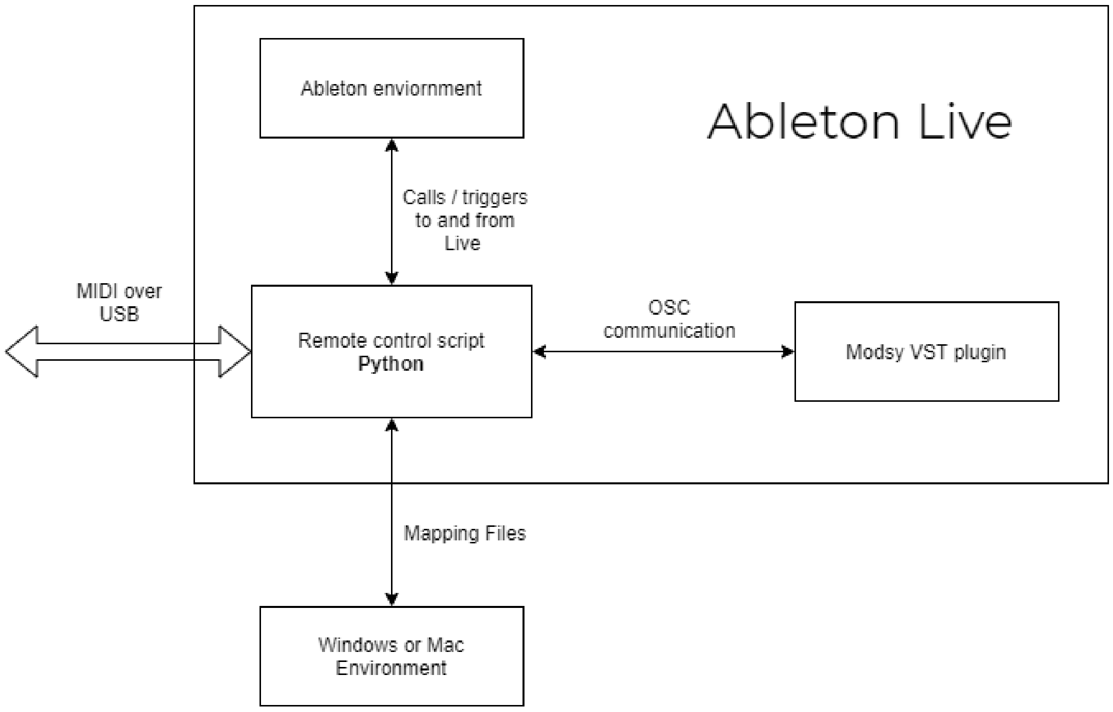


Figure 35 - Ableton Live system overview

If the same system architecture is used for the current concept it might be important to consider if the microprocessor of Modsy can handle the visualisations and game mechanics of the concept. It should be considered what the capabilities of the current Modsy controller are.

Modsy capabilities

Since the current Modsy controller works with the Arduino environment it would be beneficial for a more playful solution to be compatible with the Arduino environment. This means that the microcontroller or chip used in the concept has to be able to be programmed with the Arduino IDE. The same chip could be used as in the current Modsy controller: an ATmega32u4 as seen on the Arduino Leonardo. The capabilities of this microprocessor are listed below:

Digital I/O	20
Analog Input Channels	12
Flash memory	32KB
SRAM	2.5KB
EEPROM	1KB
Clock speed	16MHz

Table 20 - Technical specifications Atmega32u4

Both the analog/digital input, output and memory could be expended if necessary. The clock speed cannot be increased, if improvement is needed a different chip should be used. To assess whether the current Modsy controller could handle the current concept reference projects were investigated. A project like the Arduboy for instance is fully based on an Arduino chip and has dynamic visualizations (https://arduboy.com/). This project is a Gameboy controller based around an Arduino microprocessor. Another project showing the performance of an Arduino chip is a project like Arduino Tetris (Wisesa, L., Dirakit community, 2017). This project uses an Arduino uno, which has a clock speed of 16MHz, just like the current Modsy controller. Based on these examples it seems feasible to build the current concept with the use of the ATmega32u4 chip. However its limitation should be noted.

5.5 - Controller Design

The current concept could directly be implemented on the Modsy controller, using the same microcontroller, same displays, same LEDs and same potentiometers and buttons. However, this would mean that the Modsy in its natural state would not seem more playful in form or function. Only upon triggering the spaceship game will the controller appear to be more playful in function (through the gameplay) and more playful in form (by the visualisation on the displays).

As stated in literature research, playfulness is one of the main attributes that consumers use to distinguish between product appearances (Blijlevens, Creusen & Schoormans, 2009). Thus, it is important that users directly see the playful potential of the product. As stated in chapter 2 of this report an import factor to consider is the playfulness in form. Playfulness in form can be defined as: Distinct quality of visual engagement that people associate with being playful. Many ways can be derived to create this playful visual engagement. However, within the scope of this thesis not all possible formfactors can be evaluated. So, it was decided that a conceptual evaluation of the use of a playful formfactor would be executed. A simple formfactor would be added to the design, that could determine whether it was effective to design for playfulness in form.

Then, what playful formfactor could be added? In order to create playful visual engagement the PLEX categories of the current concept were used. The most dominant PLEX categories in the current concept are fantasy, captivation, exploration and discovery. Of these categories, fantasy seems to be the most logical to implement in the Modsy controller design. As can be seen on the mood board in figure 31, there are many elements that could add to the fantasy of a spaceship controller. Mainly handles, joysticks, lights and displays.

Now there is a certain balancing act that needs to be performed, while it might be more playful to add all kinds of handles, switches and colours this might not be positive for the music production process and the overall experience of a user. This is also stated as one of the requirements: The controller should keep the ability to fully focus on music production process.

Since this thesis aims to improve overall user experience with the use of playfulness, it has to be considered whether some of these potentially more playful elements won’t reduce the overall user experience. An important design aspect of the Modsy can be used during this balancing act which is the fact that the Modsy is somewhat modular as can be seen in figure 36, its legs can be taken off. This could be used in the design of a more playful Modsy controller. Playful elements could be attached or removed when deemed necessary, which could restore full focus on music production process.

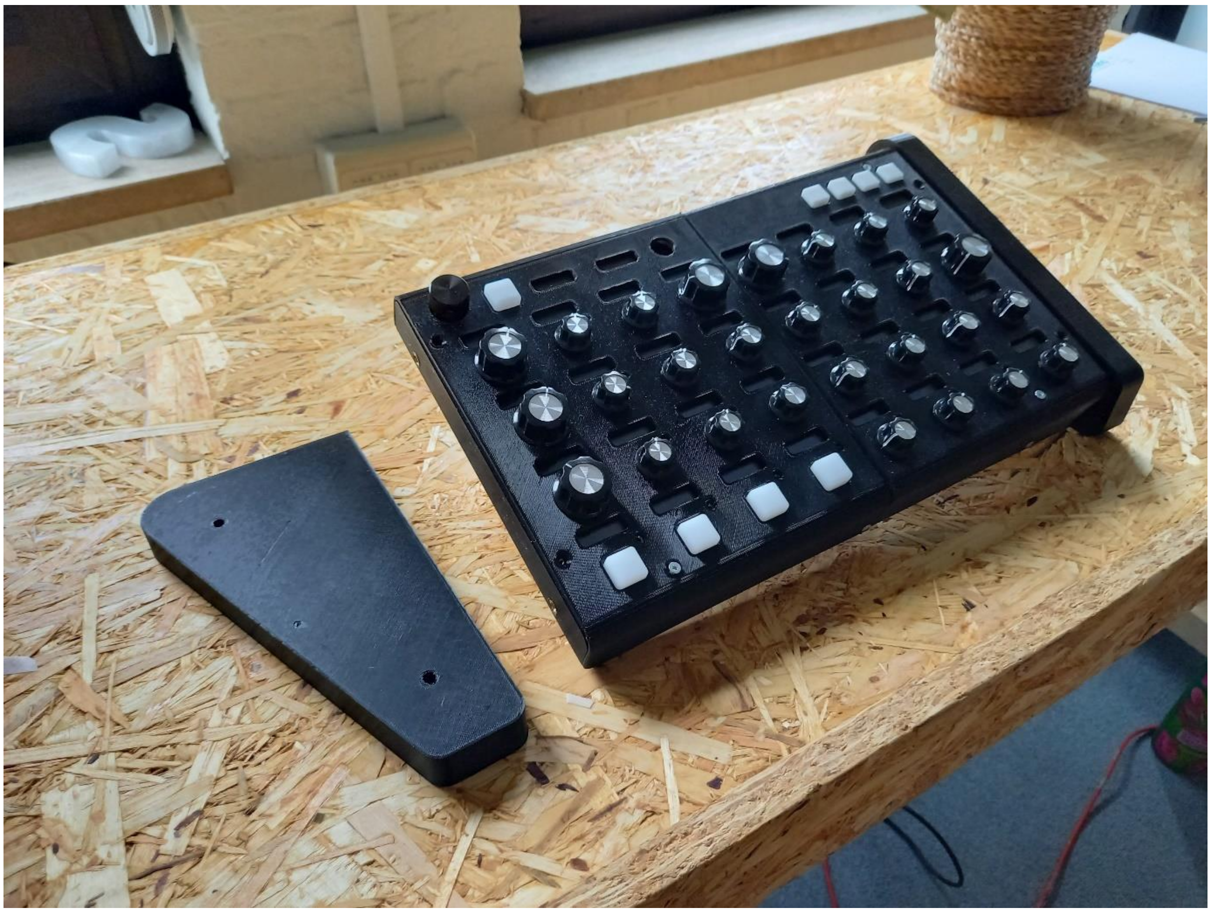


Figure 36 - Modularity of the Modsy controller

The decision was made to use a simple attachment that would be more playful in form. This playfulness will come from the use of two LED meters that indicate the status of the spaceship during the game and a joystick that could be used for spaceship control. This was expected to bring more playful visual engagement for the users of the Modsy controller. This attachment should make the Modsy controller more playful in form.

5.6 - User interaction

The core of the current concept is for the user to interact differently with the Modsy controller due to the game that he is partaking in. In this interaction the user should be able to use the Modsy parameters for game input and could retrieved game feedback via the Modsy elements as well.

This means that for input the following elements could be used:

- 32 Control elements (28 potentiometers, 4 silicone buttons)

- 4 function buttons
- Mapping control elements
- Joystick on playful attachment

Of these input types the function buttons and mapping control elements will not be used for playful interaction since these parameters do not related to a mapped instrument or effect but keep the same functionality throughout the use of the Modsy controller. Only the control elements and joystick will be used for game interaction.

For system output the following elements could be used:

- 34 OLED displays
- 42 LEDs
- Spaceship meters

Of these output elements the 34 displays are very well suited for this game concept and could perfectly be used for system feedback. These displays could already been spotted in the storyboards that were created and are one of the reasons why this concept would be very promising. To develop the game interaction for these 34 displays in the current grid these screens and their resolution were analysed. The layout of these displays can be seen in figure 37 below.

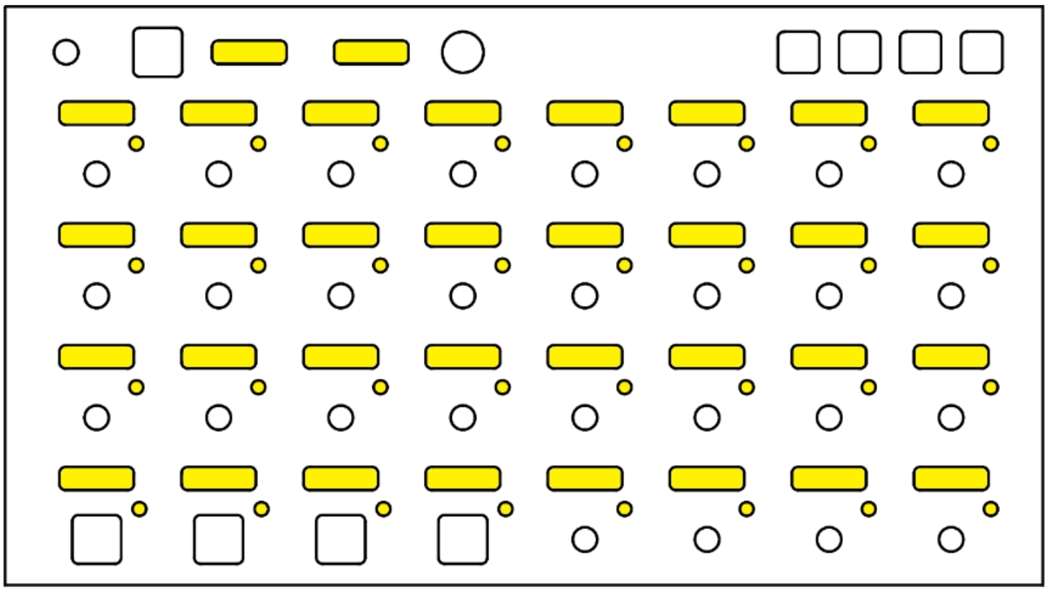


Figure 37 - Modsy display and LED layout for user feedback

The displays used in the Modsy controller are 0.91" OLED displays using a 128 * 32 pixel grid. To assess whether proper feedback could be provided throughout the game potential visualization were made.

These visualizations can be seen in figure 38, 39 and 40 below. The illustrations where made using the 128 * 32 pixel grid.

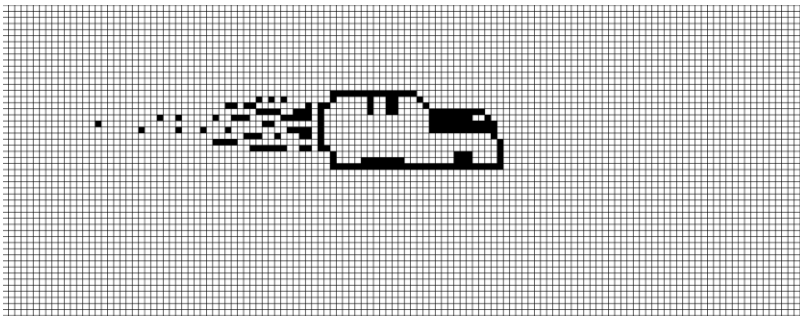


Figure 38 - Possible spaceship visualization

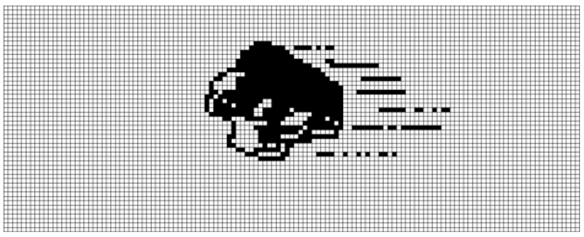


Figure 40 - Possible asteroid visualization

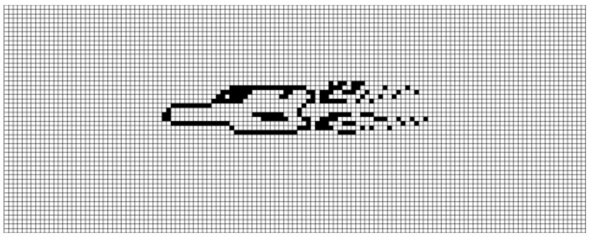


Figure 39 - Possible enemy ship visualization

In order for users to interact will all parameters of an instrument or effect users should have an incentive to alter a variety of different parameters. This is where specific game mechanics like "fix the ship" come in. If a "fix-ship" message arises at one of the screens on the Modsy controller, as can be seen in figure 41 below, the user has to twist the related parameter in order to solve this issue (to fix the ship). This would provide a way for the game to indicate parameters that the user would then feel urged to alter. This will help with the exploration and discovery of new parameters. Next to "fix parameters" there will also be "fuel parameters" to provide a more diverse way for the user to interact with the game. Both these elements will be implemented during the realisation phase.

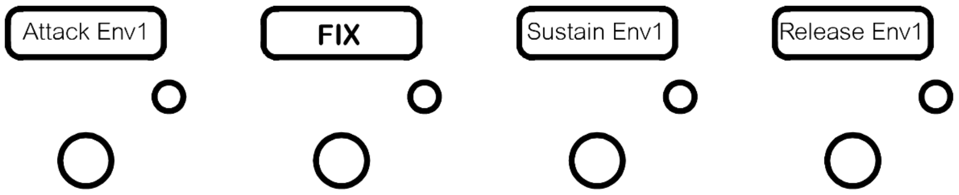


Figure 41 - Fix visualization

Now, to combine all system input and output the full controller with playful attachment and possible visualizations was illustrated. This was used to analyse the potential of the concept and assess whether this would be feasible to realize. This illustration can be seen in figure 42 below. In figure 43 a close up of the illustration of friendly spaceship and asteroid can be seen.

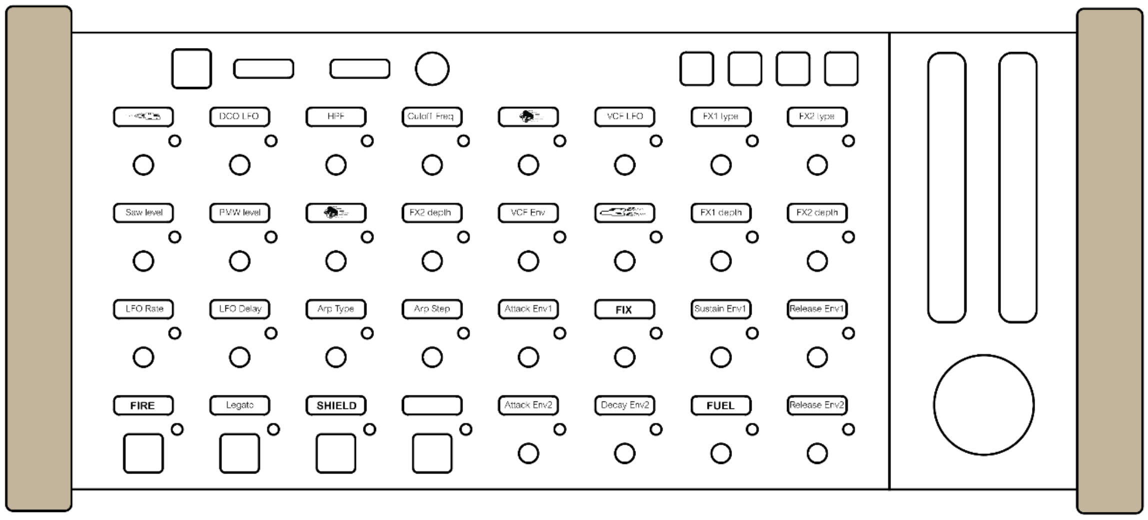


Figure 42 - Illustration of the final concept

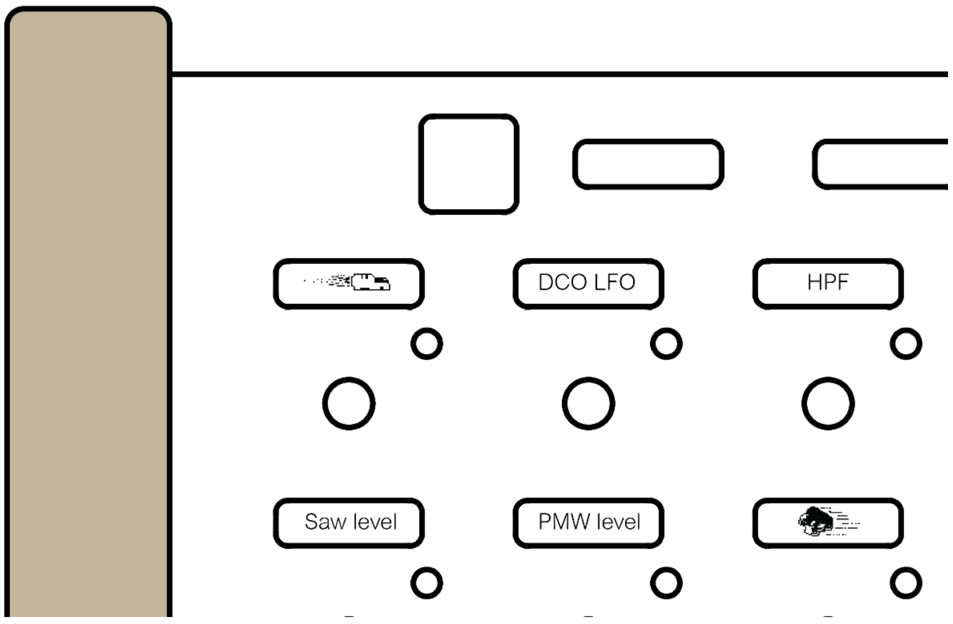


Figure 43 - Closeup of spaceship visualizations

5.7 - Instrument / effect parameter control

As stated in the preliminary requirements of the ideation phase: The control of parameters should be non-disruptive for music production process. This means that the parameters that a user has control over and the way that the user controls these parameters should not create irritating or annoying sound, but help to inspire during the creation of music. With this in mind, the influence of the current concept on the music production process should be discussed. The current concept might not always be positive for general quality of the music. This is due to the following aspects:

- **Control over specific parameters might disrupt the sound**

  All digital instruments and effects have different parameters that can influence the sound. These parameters vary from filter control to oscillator control to dry/wet knobs. Some of these parameters are fun to change drastically, but some parameters are more precise and are normally only altered within certain context. Take for instance tuning, if you create a song in a certain key and then change the tuning of one of these instruments it will sound "out-of-key", which is typically not what a musician would want.

  In order to create a playful solution that is non-disruptive for the music production process it has to be considered what parameters would be suitable for interaction. For this purpose, an analysis has been performed of 3 different digital instruments and effects. This analysis had the purpose of identifying parameters that could be disruptive within the music production process if altered during music production process. The digital instruments analysed in this exercise are: Roland Juno-106 VST, Arturia Mini-V, Sylenth-1 VST, the digital effects analysed in this exercise are: Ableton Echo, Fabfilter Pro-R, and Ableton Glue Compressor. This analysis was done through a sound design exercise in Ableton Live, all knobs were randomly changed during music production process and the parameters that badly influenced the sound were noted. The parameters that could have disruptive effect on the music are listed in table 21 on the next page. In the first row of this table the general parameter name has been described, the second row describes the amount of disruption that could occur during the music production process, the third row holds a short description of how it disrupts the sound, the last row holds a description on the occurrence of this parameter on instrument and effect plugins.

Parameter	Disruption	Description	Occurrence
Level or Volume	High	This could disrupt the balance of the musical piece and does not alter sound in a fun or creative way.	All Instruments, some effects
Tuning	High	Tuning might result in sound that is “out-of-tune” with the rest of the music. This might not be preferable.	Most Instruments, not on most effects
Coarse	Moderate to High	Also a tuning parameter that might result in out of tune music. This parameters is somewhat more subtle then the tuning parameter.	Some instruments, not on most effects
Attack	Moderate	Could be problematic if tweaked too much. Results in non-audible sound if increased too much.	Most instruments, Not on most effects
On/Off OSC	Moderate	Switching on and off oscillators might reduce the sound by a significant amount. if all oscillators are off there is no sound at all.	Most instruments, Not on most effects
Helper parameter	Moderate	There could be certain supportive parameters that keep timing or structure but do not add to the music itself.	Small amount of instruments and effects
Feedback	Moderate to High	This parameter could be disruptive if tweaked by too much.	Some instruments, some effects
Input / Output	High	These parameters could be seen as volume parameters. As stated before, volume control could be problematic.	Most instruments, Most effects
Makeup	High	Parameter that boosts the output sound. Has an influence of overall volume of instrument or effect.	Not on most instruments, on some effects

Table 21 - Parameter analysis

What can be derived from this exercise is that since there are many different instruments and effects there are also many dangerous parameters that can disrupt the sound. These could all be excluded from the spaceship control. An option might be to allow the user to set certain parameters fixed or as unused during the playful game.

- **Fixed control might not improve discovery and exploration**
  If every time the spaceship game is played the same parameters are used for control, this game will not allow for great discovery and exploration of the controlled instrument or effect. Since these are both PLEX categories, this might end up reducing the overall playfulness of the solution.

- **There are both instruments and effect plugins that have very different parameters and purpose.**
  Plugins have a different purpose within the music production process. There are certain plugins that are more formal and leave less room for playful interaction. These plugins might not be a good fit with the current concept and it should be considered if, and how, the current concept can still be applied on these types of music production tools.

During the realisation phase these different points should be addressed in the actual functioning of the game. The connection with the Ableton Live environment will be important to consider here, since Ableton Live has all the data about parameters and types of plugins. This data can be accessed through python scripts as stated in the system analysis.

5.8 - Final Concept

The final concept is a space-invaders like game that can be played while controlling different digital instruments and effects. The parameters of the Modsy controller will have a different meaning inside of the game and throughout playing the game the user will interact with his instruments and effects in a different way.

A physical attachment will also be created which can be added to the Modsy controller and gives extra control and feedback over the game. This attachment will help to increase the playfulness in form and adds to the playfulness in function as well.

5.9 - Final requirements

The preliminary requirements of the Ideation phase will be extended with requirements resulting from the specification phase. The additional requirements have been stated in bold. The requirement: The control of parameters should be non-disruptive for the music production process was split into 3 different aspects of which two have been added to the non-functional should have requirements and one has been added to the functional should have requirements. This was the result of section 5.7 of this specification chapter.

Non-Functional

Must have

- The controller must be more playful in interaction compared to the current Modsy controller.
- The controller must be more playful in form compared to the current Modsy controller.
- The controller must be able to provide an analogue feeling
- The controller must be able to provide the same, or more expression during music production process compared to the current Modsy controller.
- The controller must be able to provide the same, or more creative control during music production process compared to the current Modsy controller.
- The controller must be the same or more fun to use compared to the current Modsy controller.

Should have

- **The control of parameters during game interaction should be suitable for both instrument and effect plugins.**
- **The control of parameter during game interaction should allow for good discovery and exploration of digital instruments and effects.**
- The controller should keep the ability to fully focus on music production process
- The controller should be able to be used by all potential users of the Modsy.

Could have

No could haves have been derived during ideation and specification phase.

Won't have

No could haves have been derived during ideation and specification phase.

Functional

Must have

- The controller must be able to control 32 parameters.
- The controller must be able to provide feedback on the names of currently mapped parameters.
- The controller must be able to provide feedback on the value of currently mapped parameters.
- The controller must allow parameters to be automated.
- The controller must be able to be used in a studio environment.
- **The Modsy controller must be programmable with Arduino environment**

Should have

- The controller should be easy to transport.
- The controller should be able to be used during a Live performance.
- **During game interaction it should not be possible to alter disruptive parameters.**

Could have

- The controller could have dynamic system for selecting disruptive parameters per instrument or effect during playful interaction.

Won’t have

- The controller won’t use a different way of parameter control compared to the current Modsy controller.

Chapter 6 – Realisation

This chapter will describe the development of the technology specified in the previous chapter. It will discuss game mechanics, software and hardware development.

6.1- Initial prototyping

The initial plan, as stated in the specification phase, was to implement the full game on the Modsy controller. This includes all visualizations, game mechanics and user interactions. The capabilities of the chip of the new Modsy prototype were assessed and this seemed feasible. However, after a period of game implantation on this new Modsy prototype it turned out that there was too little RAM memory available for the dynamic values of both the functioning of the Modsy controller and the spaceship game. Next to this, the controller was too slow in updating all its display values. Both of these aspects were shortcoming of the current Modsy hardware and these problems were not known prior to the realisation phase.

Even though a possible upgrade in microprocessor seemed feasible during specification phase this turned out to be more complex than anticipated. A change in the hardware structure was not possible, the dynamic memory of the atmega32u4 (the chip used in the current Modsy controller) could not easily be expended and its speed could also not easily be increased. Thus, a different solution needed to be created, one that could reduce the dynamic memory usage while remaining same playful interaction and form in order to evaluate the enhancement of playfulness. After research, brainstorming and short discussion with thesis supervisor Wouter Eggink a decision was made to move forward with prototyping which will be explained below.

6.2 – Altered prototyping

The following solution to the hardware problem was found to be most feasible:

The game mechanics and visualisations will be outsourced to the computer, this includes visualisation of the spaceship, asteroids and enemy spaceships. The user interaction and some basic visualizations will take place on the Modsy controller. This would outsource most dynamic memory consuming calculations and screen changes to the computer and still allow for a complete user experience and user test. The implementation of this solution can be read throughout this chapter.

6.2.1 - Game mechanics

The game mechanics specified in chapter 5 were implemented and have been listed below with a short description. The computer program that was developed in order to show the visualizations and take care of game mechanics was written in Java in the Processing environment.

Manoeuvre ship

The ship has to be able to move in two axis of motion, X and Y. This gives users a sense of control and exploration. An impression of the spaceship can be seen in figure 44 below.

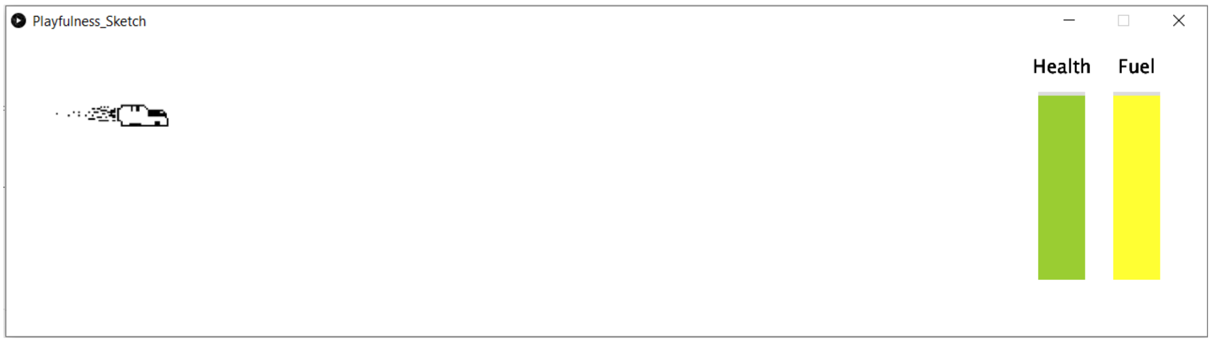


Figure 44 - Modsy spaceship software realisation

A user can take control over the spaceship using the joystick controller on the playful attachment. The joystick of this playful attachment can be seen in figure 45 to the left.

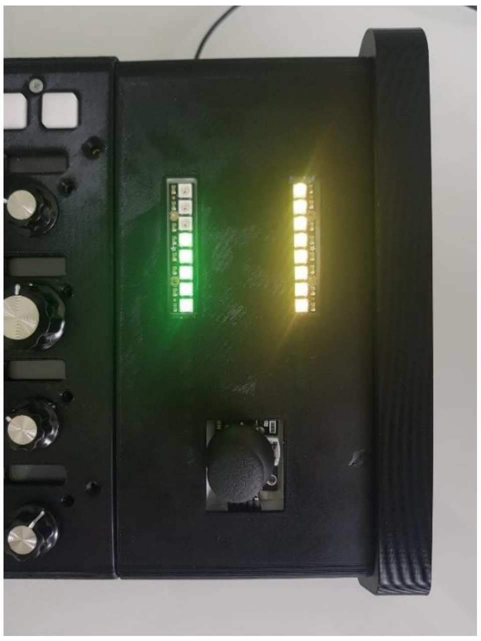


Figure 45 - Modsy playful attachment realisation

Health and fuel bar

As described in the specification phase a majority of the interaction with the parameters of an instrument or effect will be due to the fixing and fuelling of the spaceship. To provide an incentive for using these parameters a health and fuel bar have been created. The health and fuel of the ship will reduce at certain moment throughout the interaction as will be explained at the relevant interactions, for instance when hitting an asteroid. The health and fuel level can be seen in the program window on the computer as displayed in figure 44, or on the playful extension as can be seen in figure 45 on the last page.

Avoid asteroids / enemy ships

During the game the user has to avoid different asteroids and enemy ships. This can be done by manoeuvring around these elements. Both elements are spawned at the right of the game view. They move at random different speeds toward the spaceship on the left. The enemy ships will shoot at random intervals. The spaceship can be hit by an asteroid, enemy ship or bullet and all of these elements reduce the health of the ship. All elements can be seen in figure 46 below.

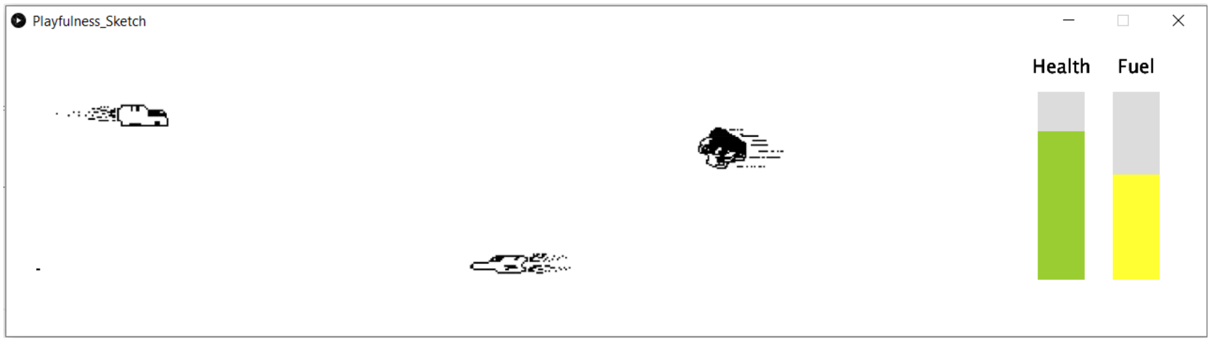


Figure 46 - Asteroid and enemy spaceship realisation

Completion of mission

The game reaches an end-state after certain time period (currently 3 minutes) where the ship reaches a planet, this is supposed to represent the final destination of the ship. This can be seen in figure 47 on the next page.

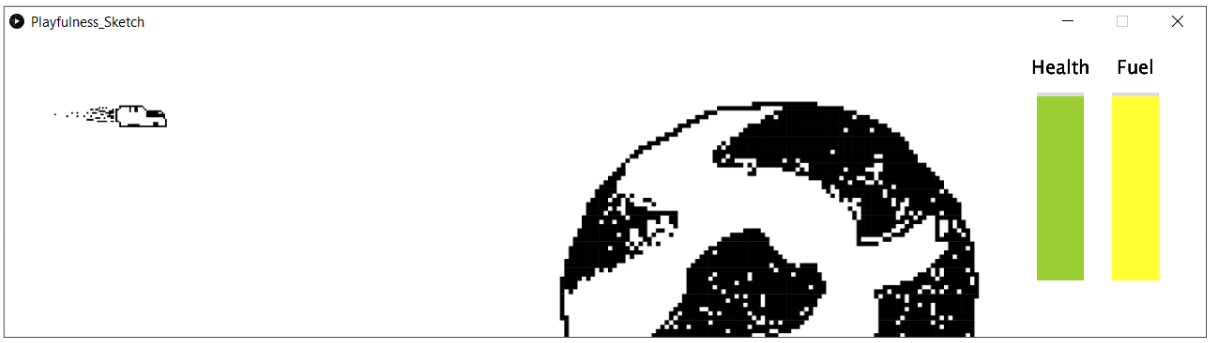


Figure 47 - Mission completion implementation

This element of completion is supported by the PLEX framework and also ensures that the focus of producers or performers will be shifted back to the music production process after a certain time period.

Fix errors / fuel the ship

A user will interact with the parameters of a digital instrument or effect through the twisting of parameters indicated by LED lights. This interaction will be to fix the ship and increase the overall health, or fuel the ship and increase the overall fuel of the spaceship. These parameters can jump between different parameters on the Modsy controller. This type of interaction will initiate in the processing program on the controller with a popup on the screen, this can be seen in figure 48 below.

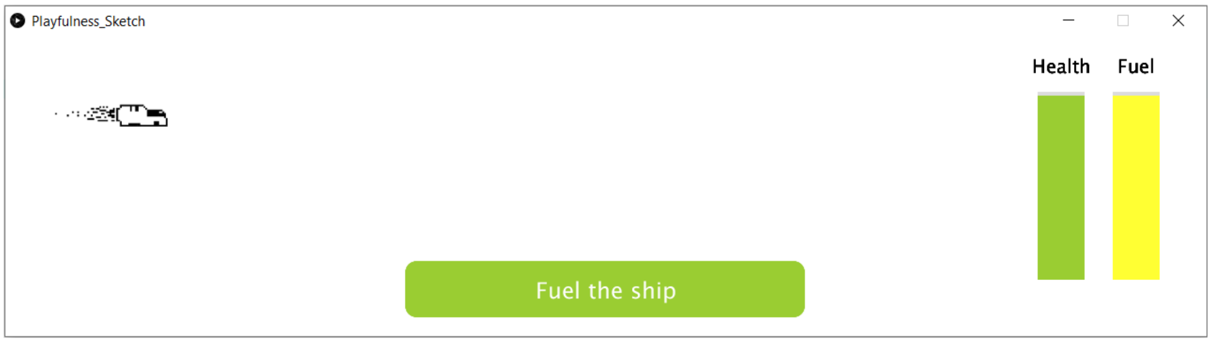


Figure 48 - Parameter pop-up implementation

After the popup can be seen in the processing program one of the LEDs on the controller will light up, this indicates which parameter should be altered in order to fuel or fix the ship. This can be seen on the next page in figure 49.

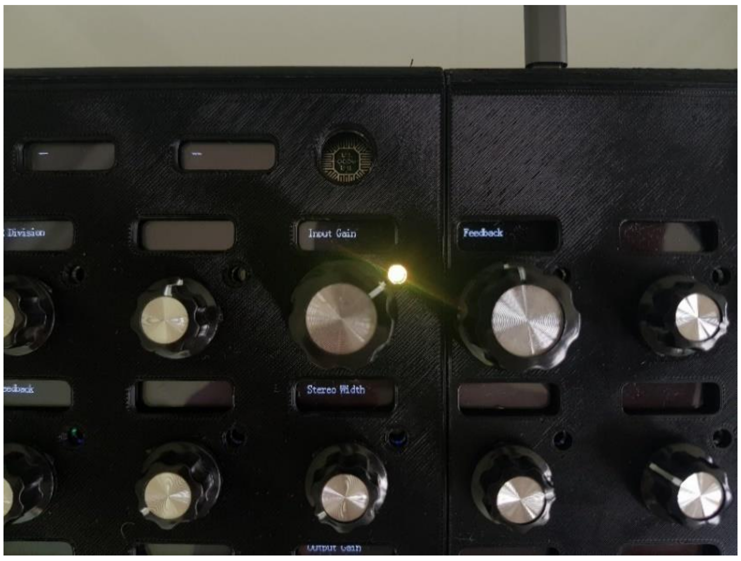


Figure 49- Image of fuel ship indication

If this parameter is twisted within a certain timeframe the processing program will show that the reload or fix was performed in time and health or fuel will be added as can be seen in figure 50. The timeframe for twisting the parameter is currently set at a random time interval between 2 and 6 seconds. This changes every time a new parameter is selected.

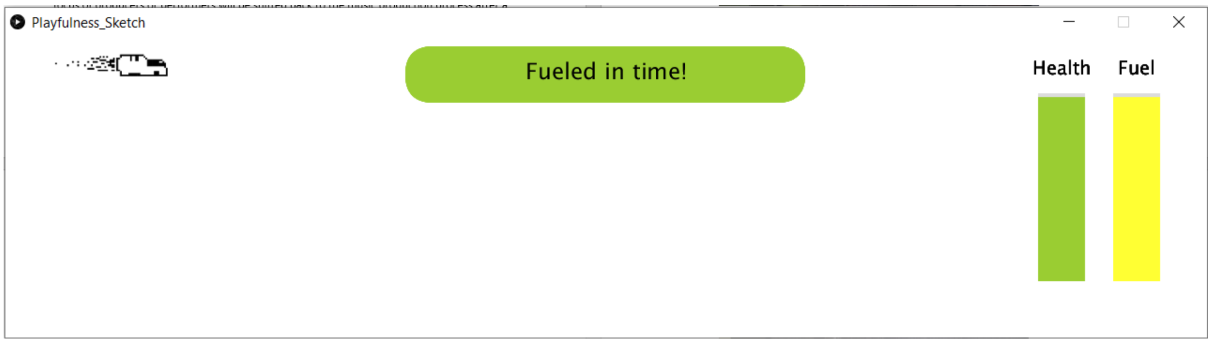


Figure 50 - Fuelled in time feedback

If the parameter is not twisted in time the display will show that you did not complete the task and health or fuel will be deducted as can be seen in figure 51.

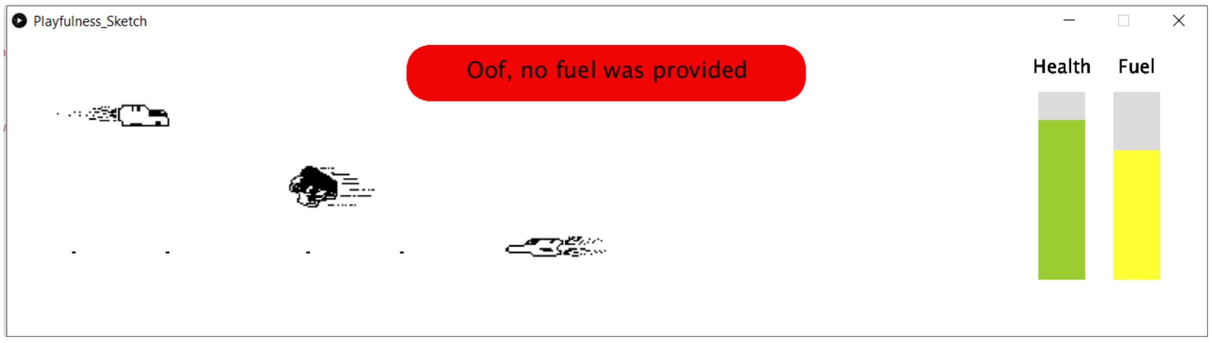


Figure 51 - Fuel parameter not twisted in time

Shoot gun

The spaceship is able to shoot at asteroids or enemy spaceships that approach it. The shooting can be performed by pressing one of the 4 buttons on the Modsy controller. The indication next to this button will be a red LED, this is explained before partaking in the game. This indication can be seen in figure 52.

Activate shield

The shield can protect the ship from hitting asteroids and enemy spaceships, this can be seen in figure 53. This shield can be activated by pressing the button indicated with a blue LED. This indication can be seen in figure 52.

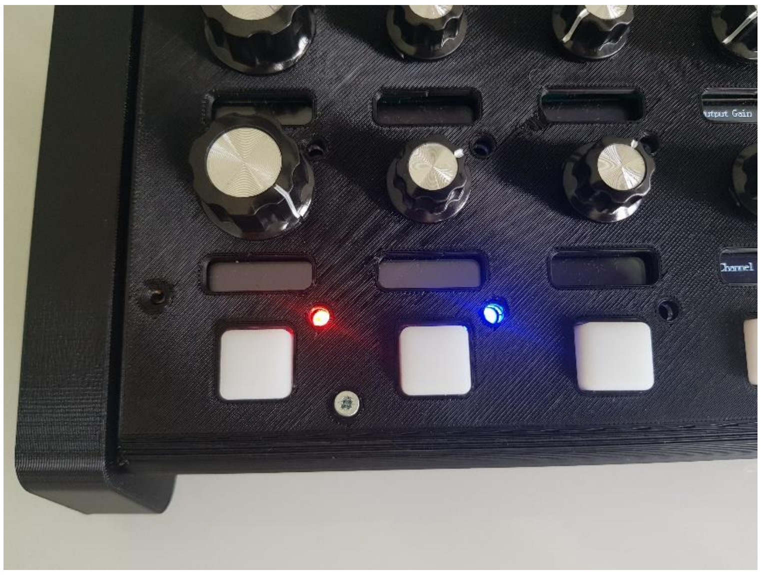


Figure 52 - Image of fire / shield button

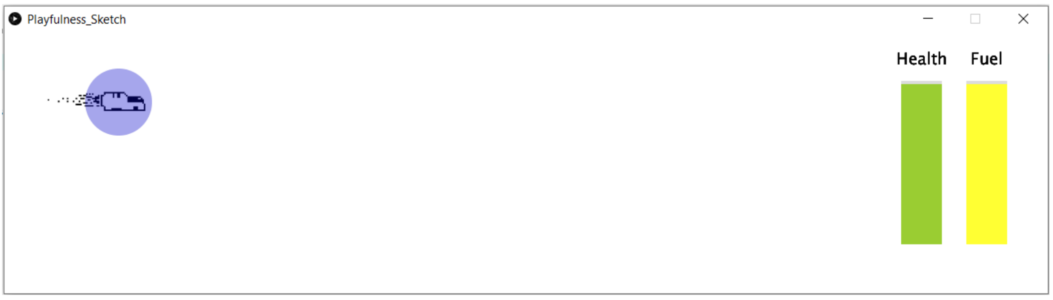


Figure 53 - Shield of Spaceship activated

6.2.2 – Hardware

As stated in the specification chapter, and as already discussed in the game mechanic section, a hardware element was added to the Modsy controller. This hardware element should provide more playfulness in form. Two elements were added on this playful attachment: a playful way of user input with the use of a 2-axis joystick, and a playful way of system output using LED sticks. This playful attachment can still be removed from the main Modsy controller during usage to remain the possibility to fully focus on the music production process. The full "more playful" Modsy controller can be seen in figure 54 below.

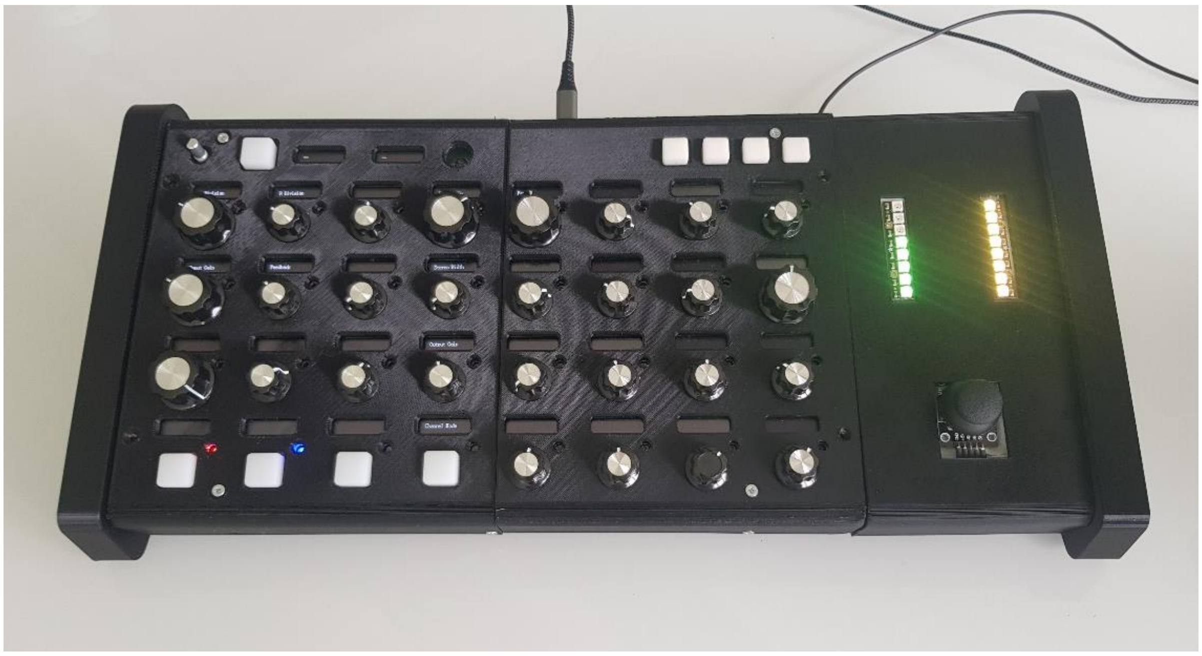


Figure 54 - Realised version of the theoretically more playful Modsy controller

Different hardware components were used to realise this design, these are listed below.

- **2-Axis Joystick**
  For the manoeuvring of the spaceship a 2-axis joystick will be used. This can provide 3 different types of data to the system: X component for left and right movement, an Y component for top and bottom movement, and a push component that can provide data on whether the joystick has been pushed down. Only the X and Y component will be used in the prototype. The following specific joystick was used in the design: Joy-It KY023JM, purchased from Conrad (Conrad, 2021).

- **LED bars**

  For the LED bars addressable RGB LEDs were used. These LEDs can easily be controlled using Arduino libraries and have the ability to change to many colours. They are also easy to implement with regards to the wiring since the data line of all of the LEDs is shared. Prefabricated LED sticks with WS2812 LEDs where used. The specific part name is: NeoPixel Stick - 8 x 5050 RGB LED with Integrated Drivers from Adafruit (Adafruit, 2021).
- **Arduino Leonardo**

  A microcontroller was used for the communication of the above mentioned elements with the computer. Since an Arduino Leonardo was available this was used in the design. However, most Arduino controller could have been implemented in this design since there are low requirements for update speed and communication for this playful attachment.
- **Plastic casing**

  All above mentioned electrical elements were fitted inside a casing designed and 3d printed by Bram van Driel, a student of the bachelor Creative Technology and co-founder of Weirdly Wired.

Wiring playful extension

The different hardware components had to be connected to the Arduino Leonardo. This was fairly straightforward and can be seen in the schematic below named figure 55. Pins A0 and A1 of the Arduino Leonardo were used for Joystick input, Pin D7 was used for the LED data.

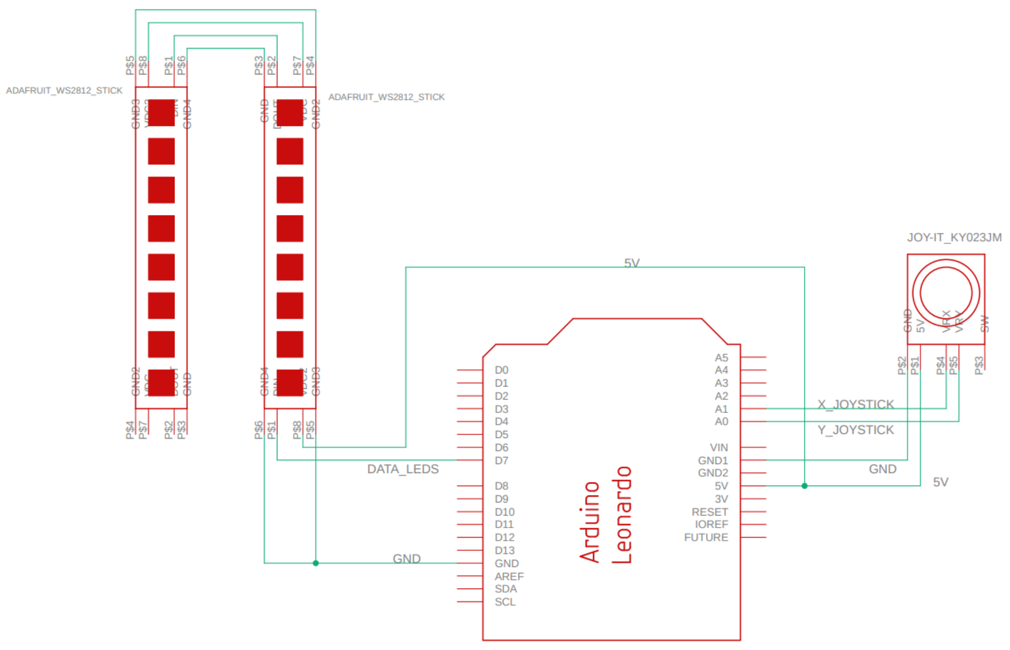


Figure 55 - Wiring playful extension

6.2.3 – Software

For the current implementation of the playful concept a specific software structure had to be implemented that used a processing program for visualisations and game mechanics. This processing program had to communicate with both the playful attachment and normal Modsy controller in order for the concept to work. This section describes how this software implementation was structured and what each element in this software structure should do.

Software structure

The current implementation uses 3 main software elements that can communicate with one another. These can be seen listed below. A general overview of the communication of these 3 software elements can be seen in figure 56 below.

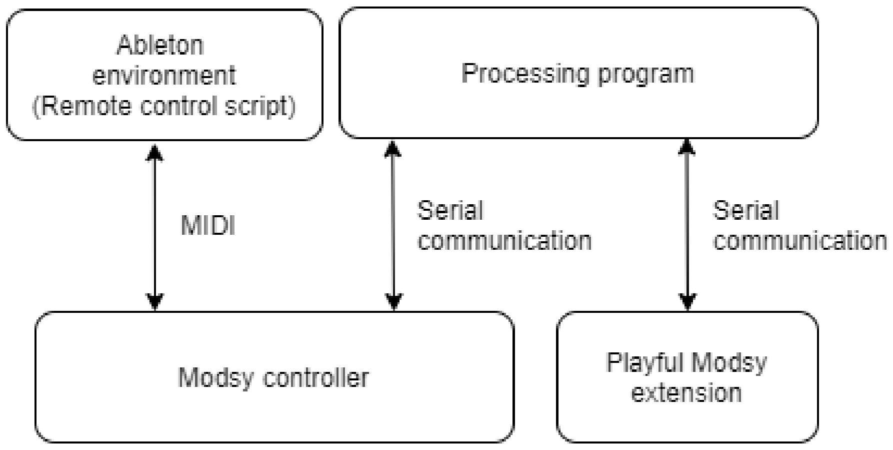


Figure 56 - Software communication structure

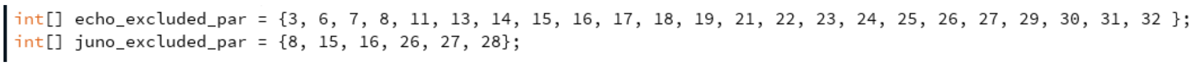
- **Processing program**, in Java, for playful game mechanics / visualizations
  The processing program handles the main game mechanics and visualisations of the game. This program takes care of the position of game elements, the spawning of elements, checking hitboxes, establishing the fuel and fix parameters and taking care of the general game state of the game. This processing program communicates with both the normal Modsy controller and playful attachment over Serial. The code of this processing program has been shared in the appendix as item I.
- **Arduino**, in C++, Modsy controller program
  The Arduino environment is used for the embedded software of both the normal Modsy controller and the playful Modsy extension. The software on the normal Modsy controller

enables the visualisation of fuel, fix, shoot and shield parameters. It also communicates when a certain parameter has been altered. The software on the playful extension enables the use of the joystick for the game and the visualisation of the health and fuel as LED bars.
The code of both Arduino programs have been shared in the appendix, item J.

- **Remote control script**, in Python, Ableton interaction
  The remote control script is a python script that can be enabled inside of Ableton Live that provides control over Ableton Live functionality. The remote control script maps the parameters of the Modsy controller to the parameters of a digital instrument or effect. Within the scope of this thesis this remote control script has not been altered. However, the remote control script could be important in the avoidance of disruptive parameters in further implementation of the concept. The code of the remote control script has not been shared in the appendix since this was not altered within the scope of this thesis.

6.3 - Parameter control

As stated in the specification phase, in order to make the game non-disruptive for the music production process there should be a way to remove the disruptive or harmful parameter from the playful interaction. This could be done by smartly monitoring the parameters of a mapping and filtering these mappings on their parameters for playful control. However, this system is complex to implement for the current prototype and during evaluation only 1 instruments and 1 effect will be used due to time constraints. So, for the scope of this thesis the parameters that could be disruptive will be manually excluded from the playful interaction. This will be done by have certain exclusion lists for certain instruments or effects with disruptive parameters. In a later stage this could be automated by altering the python script in Ableton Live. An example of the lists used with the final prototype can be seen in figure 57 below.



```
int[] echo_excluded_par = {3, 6, 7, 8, 11, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31, 32 };
int[] juno_excluded_par = {8, 15, 16, 26, 27, 28};
```

Figure 57 - Excluded parameters playful interaction

The use of parameter exclusion removes the disruptive parameters from the interaction during evaluation, however there are two more parameter related concerns that should be addressed in implementation as pointed out in the specification phase: fixed parameter control and control over both instruments and effects. The fixed parameter control concern relates to the fact that if the parameters used in the game would not alter throughout the game this would make the musical interaction less explorative and discovery rich. However, due to the implementation of the fix and fuel parameters this seems to have been addressed. The final concern is the fact that the game interaction should make sense for both instruments and effects. During testing only one instrument and effect will be used, both will be chosen to suit the concept and should not create problems. However, in a later stage the usefulness of the concept should be tested for a variety of different instruments and effects.

Chapter 7 – Evaluation

The goal of the evaluation phase is to address all the requirements that have been stated in the specification phase and address the main research question stated at the beginning of the report. During this phase both the normal Modsy controller and playfully enhanced Modsy controller will be evaluated using 4 different methods to assess how both concepts differ in playfulness and user experience.

7.1 - Testing methods

During this evaluation phase 4 different evaluation methods will be used.

- **Appearance identification**
  An appearance identification method will be used to assess the form of the controller. This method will result in qualitative data about the appearance of the prototypes and should help to determine whether the new prototype is more playful in form. In this method different cards with single words are used to describe the appearance of a product. These words have been identified by Blijlevens, Marielle and Schoorman (2009) as the most frequently used categories to describe product appearances. These categories should help participants to assess the form of the controller and provided the researcher with useful information about the perception of the form. Participants are free to add additional terms and explanation of the form of the controller next to the use of the cards. This is not an established method and is only meant to provide indication of the participant's form perception of both Modsy prototypes. The cards, as used in this thesis, have been included as appendix E to this report. The words on the cards are: old-fashioned, classical, oldish, kitsch, retro, functional, simple, boring, plain, playful, funny, unusual minimalistic, sleek, futuristic and modern.
- **Assessment of flow**
  An assessment of the flow state of a participant will be used to assess the playfulness experienced during interaction. As stated in the literature research, flow analysis is a method that can be used to assess the state of playfulness an individual is In. For the evaluation of the prototypes the Flow Short Scale will be used since this is a quick and reliable way to assess the flow experienced during an activity (Rheinberg, Vollmeyer, and Engeser, 2003; Kyriazos et al, 2018). The items 11 – 13 of the questionnaire are designed to assess perceived demand / skill fit. The first 10 questions are the core of the FSS and measure the components of flow

experience (Engeser, 2012). Since flow experience is of most interested during this evaluation only the first 10 questions will be used.

- **Observation**

  To further assess the playfulness of participants during interaction with both prototypes participant behaviour will be analysed. Glynn and Webster (1992) identified 5 characteristics that can be used to assess the playfulness in adults. These 5 categories, spontaneous, expressive, fun, creative, and silliness will be at the centre of the observation. These categories are normally used to assess the playfulness in character of an individual and are not often used in direct observational manner, however, these character traits could provide good insight into the playfulness that results for certain interaction or behaviour with the prototype.
- **User experience evaluation**

  Lastly, the overall user experience should be assessed. This will be done through both a SUS analysis, to assess the overall usability of the system, and by gathering overall user feedback about the system. This SUS method is explained in detail in the method section of this report. The SUS questionnaire as used in this evaluation phase has been added to the report as appendix A.

7.2 - Test procedure

This section shortly describes the procedure that was used during the evaluation of the theoretically more playful Modsy prototype and the normal Modsy prototype.

At the start of the research all participants were provided an information brochure and informed consent form for participation, both have been included as appendix items F and G to this report. The participants were provided a short introduction into the nature of the research but no information was shared about the design or desired result. After the informed consent had been signed the testing would start.

For one half of the participants the normal Modsy prototype was provided first, for the other half of the participants the playfully enhanced prototype was provided first. This was done to control for the variance in outcome that could occur by always providing one of the prototypes first. This method should prevent the outcome from being biased based on order of prototype interaction.

The participant were first asked to analyse the product on appearance using the appearance identification method as described in the previous section. A result of such an identification session can be seen in figure 58 below.

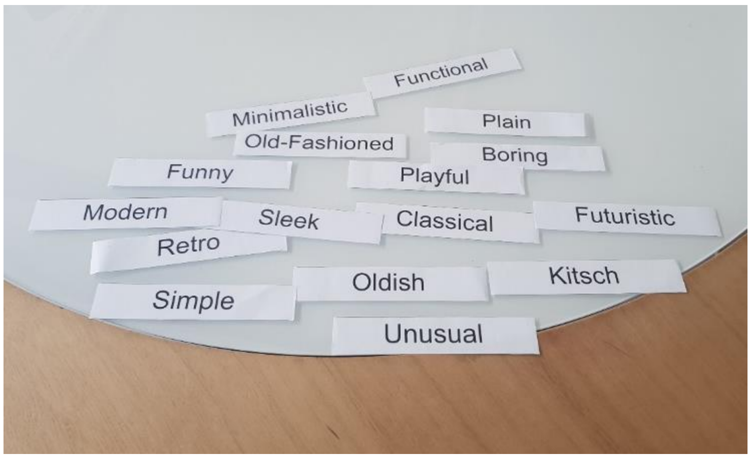


Figure 58 - Form analysis Modsy controller

Thereafter the participant is asked to use his provided prototype with a digital instrument. For the more playful Modsy prototype the game will start and users can use all functionality described in the realisation phase to interact with this instrument. Thereafter an effect would be selected and used in similar fashion. During control of both an instrument and effect the participant will be observed and interesting aspects will be noted.

After both an instrument and effect has been used with the prototype the user is asked to fill in two forms: the Flow Short Scale and SUS questionnaire. Both will be answered using Google Forms.

After the forms had been completed there was room for general feedback and remarks about the currently tested prototype. Thereafter the prototype would be switched, if the theoretically more playful Modsy controller was provided first then the normal Modsy controller would be provided and vice versa. Then all methods would be repeated.

7.3 - Results

The results consist of the data gathered through all 4 methods, this resulted in quantitative data for both the flow analysis and the sus analysis. Qualitative data was gathered through form analysis, observation and user feedback.

There were only 6 participants included in this evaluation. This means that all quantitative data does not produce any significant results, this data will only be used as an indicator and support of qualitative data. All participants were potential customers of the Modsy controller. These participants varied in music production skill level, gender and age. All participants were above 18 years old.

7.3.1 – Results Form analysis

The Form analysis resulted in groupings or lists of words that would describe both prototypes. An example of a grouping created for the normal Modsy controller can be seen in figure 59 below.

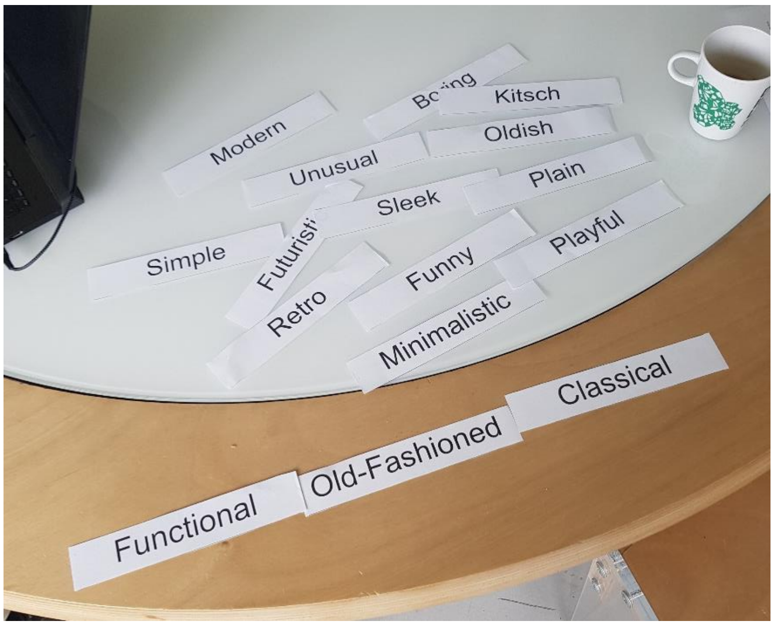


Figure 59 - Example of appearance grouping normal Modsy controller

Most interesting are the groupings that included the word playful. Since this would indicate that the participants find the prototype playful in form. Other words used to describe the prototype are interesting as well, but won't all be analysed within the scope of this thesis. For the normal Modsy prototype **0 out of 6 participants** used the word playful to describe the appearance of the prototype. For the playfully enhanced Modsy controller **2 out of 6 participants** used the word playful to describe the prototype. This would indicate that the newly created prototype is slightly more playful in form, however this is not by a great margin and only assessed by 6 participants.

7.3.2 – Results Flow analysis

The Flow Short Scale resulted in categorical data on a 5-point Likert scale. Since the participant number is low (n = 6) this will not lead to significant result but an indication of the flow experienced by participants can be derived. As stated, only the first 10 questions of the Flow Short Scale will be used. As for the outcome of this analysis, the higher on the 5-point Likert scale the higher the flow that was experienced during product interaction. The average value for each question have been calculated and compared to provide overview of the difference between the two prototypes and can be seen in table 22. The highest value between both prototypes has been indicated in green.

#	Question	Normal Modsy	Playfully enhanced Modsy
1	I feel just the right amount of challenge	3.5	3.83
2	My thoughts/activities run fluidly and smoothly	4.2	3.17
3	I do not notice time passing	4.7	4.3
4	I have no difficulty concentrating	3.7	2.8
5	My mind is completely clear	3.5	3
6	I am totally absorbed in what I am doing	4.5	3.8
7	The right thoughts/movements occur of their own accord	3.7	3.5
8	I know what I have to do each step of the way	3.5	3.83
9	I feel that I have everything under control	3.7	3.5
10	I am completely lost in thought	3.5	2.7

Table 22 - Flow Short Scale results

This data shows that the flow experienced with the normal Modsy prototype seems to be higher than with the extended Modsy prototype for most items on the Flow Short Scale. From these flow measures we can derive an indication of the playfulness that was experienced, which can then be stated to be slightly higher during interaction with the normal Modsy controller than with the playfully extended Modsy controller. However, it is difficult to provide an actual level of playfulness that was experienced, both prototypes indicate high flow measures (score above 3 points), thus likely high playful behaviour.

7.3.3 – Results SUS analysis

The SUS scores of both the playful enhanced and normal Modsy prototype can be seen in table 23 on the next page.

	Normal Modsy controller	Playfully enhanced Modsy controller
SUS score	81.25	75

Table 23 - Results SUS analysis

Overall, the normal Modsy controller can be stated to have better usability than the playfully enhanced Modsy controller since its SUS score is higher. This could be logical since more functionality has been added to the playfully enhanced controller which generally tends to make a system more complex. Both systems have a SUS score above 68, which indicates that there are no major usability issues with the system. The SUS score of the normal Modsy controller is higher than 80.3 which indicates that people are very positive about the usability of the system (Thomas, n.d.), this is not the case for the playfully enhanced Modsy controller.

7.3.4 – Observation / user feedback

User feedback and observation provided qualitative data into the experienced playfulness of the playfully enhanced Modsy controller and its provided user experience. This feedback has been listed below for the normal Modsy controller and playfully enhanced Modsy controller.

Normal Modsy proto

<u>Quotes</u>

“Process of music production is more fun than just the software”

“You create sounds that you otherwise would never create”

“Not creative in terms of functionality, other synthesizers can do this as well”

<u>Observation</u>

The normal Modsy prototype is perceived rather well by most participants. There seemed to be strong captivation during Modsy usage with participants jokingly stating that they could stay there for hours. This shows a certain level of captivation and exploration that is possible with the use of the Modsy controller. However, it differs highly on the participants skill level how the Modsy is used and what sounds are created, participants that are more familiar with sound design and different digital instruments and effects are more aware of the terminology and can get more captivated by the Modsy controller.

Playful Modsy prototype

Quotes

“You follow more what the computer wants”

“Not actively creative, but passively creative”

“Music is an extra, the game seems to be central”

“You are too busy with the game to realize what is happening”

“More accidental cool stuff takes place”

Observation

As can be read from the above mentioned quotes this prototype was received with some concerns and remarks by the participants. One aspect that came up with most participants is the fact that while altering the sound you do not have time to stop the game and further tweak your sound, you have a continuous pressure from the game to keep altering the sound, which was not always perceived as positive for the creative process. However, there was a significant amount of laughter and silliness during interaction which could indicate more playful behaviour. That being said, **none of the participants asked to play the game again.** With one participant even stating that if they want to make weird sounds they would just aggressively turn knobs and get the same effect. Users did state its potential for the learning synthesis and potential implementation for children. Next to this, users would also not describe the experience as negative, but in comparison to the normal Modsy controller most participants would rather spend time with the normal Modsy controller.

7.4 - Discussion

The results indicate that the new prototype is only slightly more playful in form, not directly more playful in interaction and not directly positive for the user experience compared to the current Modsy controller.

After reflection and analysis of the results the most likely explanation is that instead of enhancing the current playful experience of the Modsy controller a new playful experience was created that did not include certain PLEX categories that were involved during interaction with the normal Modsy controller. These categories are **Control, Relaxation and Captivation.** The experience with the normal Modsy controller turned out to be highly playful. During usage of the normal Modsy system users would have full control over an instrument or effect, could take their time to alter any parameter, and get completely captivated in this activity. This combination seemed to result in a highly playful state and was not experienced with the new Modsy prototype.

Now this outcome requires a bit of perspective, the playful attachment of the Modsy controller could be removed and the game does not necessarily have to be played during the music production process. This means that the original playfulness of the Modsy can always be restored. Within this thesis a direct comparison has been performed between the normal Modsy controller and the more playful Modsy controller, but this might not have been completely valid.

If the current concept is provided as an addition to the Modsy controller will this increase the overall playfulness of the Modsy controller? This question is very hard to answer following the current evaluation. The Flow analysis and observation do indicate a state of playfulness (high flow state and observation of playfulness) during interaction with the new Modsy prototype, however is this due to the same playful experience as the normal Modsy? Or is this a new playful experience? More research should be performed to further analyse the potential of this concept on the overall perception of the Modsy controller.

7.5 - Evaluation of requirements

The evaluation of requirements will discuss all requirements as stated in the specification phase and assess whether these have been met. Both the functional and non-function requirements will be evaluated. These have been listed with NFR for non-functional requirement and FR for functional requirement. The requirements have been grouped based on the MoSCoW categories.

Must haves

NFR1 - The controller must be more playful in function compared to the current Modsy controller.

This requirement has not been met.

Current evaluation indicates the controller to be less playful in interaction in comparison with the current Modsy controller.

NFR2 - The controller must be more playful in form compared to the current Modsy controller.

This requirement has been met.

2 out of 6 users used playful to describe the new prototype. No participant described the normal Modsy controller as playful. This indicates that the created prototype is slightly more playful in form.

NFR3 - The controller must be able to provide an analogue feeling

` This requirement has partially been met.

Evaluation indicated that the prototype was mostly able to remain its analog feeling in form and functionality. This has been derived from the form analysis where users used word like retro and oldish to describe the new prototype. However, some users used words like modern and futuristic to describe the new prototype which could undermine the analog feeling. Control over instruments and effect remained the same, providing the same knob-feel as before, which can be understood as an analog feeling within the scope of this thesis.

NFR4 - The controller must be able to provide the same, or more expression during music production process compared to the current Modsy controller.

This requirement has partially been met.

User interviews indicated that the supposedly more playful prototype was not able to maintain the same expression. However, since the playful elements could be removed from the interaction in current implementation the controller could still result in the same expression as the current Modsy controller.

NFR5 - The controller must be able to provide the same, or more creative control during music production process compared to the current Modsy controller.

This requirement was met.

The playful interaction was able to provide a new form of creative control while maintaining the creative control possible with the normal Modsy controller.

NFR6 - The controller must be the same or more fun to use compared to the current Modsy controller.

This requirements was met.

The playful interaction was perceived as an additional item that could bring joy to participants, thus this would result in an overall more fun experience.

FR11 - The controller must be able to control 32 parameters.

This requirement was met.

32 parameters could be used for control.

FR12 - The controller must be able to provide feedback on the names of currently mapped parameters.

This requirement was met.

The Modsy prototype was able to show instrument or effect parameter names.

FR13 - The controller must be able to provide feedback on the value of currently mapped parameters.

This requirement could not be met.

This requirement was not met due to the current Modsy prototype. This prototype was unable, due to speed and memory problems, to update the displays with value feedback of parameters. However, with an improved prototype this could easily be implemented.

FR14 - The controller must allow parameters to be automated.

This requirement was met.

The controllers parameters could still be automated.

FR 15 - The controller must be able to be used in a studio environment.

This requirement was met.

The controller was effectively tested in a studio environment.

FR 16 - The Modsy controller must be programmable with Arduino environment

This requirement was met.

The controller could be programmed with Arduino IDE.

Should haves

NFR7 - The control of parameters during game interaction should be suitable for both instrument and effect plugins.

This requirement has partially been met

Only 1 digital instrument and effect were evaluated within the scope of this thesis. Both these elements were found usable within the current game. The main difference between both elements related to the amount of control that users would have. An effect naturally has less parameters which leads to less parameters to play with using the Modsy controller. However this also provided good oversight of all usable parameters and was found more relaxed by users. Since not a lot of instruments and effects were tested no complete compliance with this requirements can be stated.

NFR8 - The control of parameter during game interaction should allow for good discovery and exploration of digital instruments and effects.

This requirement has been met.

The playful interaction supports the exploration and discovery of new parameters very well. This was also stated as one of the positive aspects by the participants.

NFR9 - The controller should keep the ability to fully focus on music production process

This requirement has partially been met.

The controller was stated to remove focus on music production during interaction, however, through the nature of the design the playful attachment can be taken of and the playful game interaction could be stopped. This means that the focus could always be restored.

NFR10 - The controller should be able to be used by all potential users of the Modsy.

This requirement has been met.

Participants with a variety of skill level used the prototype during evaluation and all could use this prototype with no major problems. The SUS score of 75 supports this claim.

FR17 – The controller should be easy to transport

This requirement has been met.

The controller maintained compact and could still be transported in same fashion as the normal Modsy controller.

FR18 – The controller should be able to be used during a Live performance

This requirement has not been met.

This was not directly tested but the testing method provided good insight for assessment of live performance possibility. From this we can conclude that the playful enhancement would not be suited during a live performance in its current state, so this requirement has not been met.

FR19 - During game interaction it should not be possible to alter disruptive parameters.

This requirement has been met.

Through the use of exclusion lists it was currently not possible to control any disruptive parameters during the game interaction.

Could haves

FR20 - The controller could have dynamic system for selecting disruptive parameters per instrument or effect during playful interaction

This requirement has not been met.

Within the scope of this research it was not feasible to realise a dynamic system for parameter selection for playful interaction. It was too complex to develop this feature with the current system using processing, Arduino and the remote control script. This could still be developed in a later stage.

Chapter 8 – Conclusions and future work

In this final chapter a conclusion will be formed from the results of the previous chapter, furthermore a recommendation will be given for future work. The goals as stated in the beginning of this thesis will be addressed.

8.1 - Conclusion

The main goal of this thesis was to create a more playful Modsy controller that could improve user experience. This goal has not completely been achieved, however, much can be learned from the process and the created concept. The created concept consists of a playful addition that can be used in combination with the standard Modsy controller.

Within this process different subgoals were addressed and answered. Starting with the following subquestion: What is play and playfulness within the context of a music production controller? This question was answered through literature research. Within the scope of this thesis play can be described as an activity that elicits involvement and provides pleasure. Playfulness can be used to describe a state or trait of an individual, or as a product or interaction that has the ability to alter the playfulness state or trait of an individual.

Thereafter the following subquestion was answered: How can playfulness be used in the design of a music production controller? Through literature research a practical implementation of playfulness in product design was derived. Two major tools were identified to design for playful experiences and thus create more playful products. These tools are the Playful Experiences Framework (PLEX) and use of playfulness in both form and function.

Next, a theoretically more playful Modsy controller was created using the creative technology design process. This process included ideation, specification, realisation and evaluation in which a variety of methods were used. This resulted in a prototype that relied heavily on the following PLEX categories: Exploration, Discovery, Fantasy and Captivation. The created prototype has the ability to enter a game-like state where the user could interact with a spaceship game during the music production process. The user could interact with this spaceship game by using Modsy parameters that are also used to control instruments and effects. This means that while interacting with the spaceship game the user is simultaneously altering the sound. This concepts was developed to be very playful in function, the playful formfactor was enhanced by adding a specific playful attachment that could be used in the spaceship game interaction.

For the evaluation of this playfully enhanced Modsy controller the following sub-questions had to be answered: How can playfulness be measured and how does the playfulness of a product affect the user experience? After literary research it was determine that playfulness could be identified through the use of flow state analysis and observation. The effect of this playfulness on user experience could be assessed with the use of general user feedback and a SUS analysis. All mentioned methods were used in evaluation of the created prototype.

The results were analysed and discussed. From these results it could be concluded that currently the playfully enhanced Modsy prototype was slightly more playful in form, not more playful in interaction and did not have a positive influence on the user experience compared to the normal Modsy controller. However the evaluation method could be slightly invalid due to the direct comparison with the normal Modsy controller. In hindsight an alternative evaluation method could have been better at assessing the enhancement of the playfulness of the created prototype and actual implementation of the concept.

Finally, the main research question can be addressed: How can the playfulness of the Modsy controller be optimized to improve the overall user experience? From this thesis it seems high feasible that playfulness could be optimized in both form and function with the use of PLEX categories. However within the scope of this thesis no direct enhancement of playfulness could be concluded and thus no conclusion can be drawn about the impact on user experience. Results indicate that a playful experience was present during interaction with the new prototype but this experience was perceived as less playful in comparison with the normal Modsy controller. Due to the modularity and possible ways to implement the concept it might still be a valuable asset for Weirdly Wired, but more research should be performed to assess possible use and its effect on users.

8.2 - Future work

Since this thesis was somewhat inconclusive about the effect that playful can have on user experience, future work will be very important to further identify its potential. Playfulness is still a largely understudied field with regards to product design. There has been no major research into the effect that playfulness in products can have on user experience, and this is a shame since playfulness can have a real impact on human life. With regards to Modsy, there is a variety of next research steps that could be taken to better understand playfulness within the context of the Modsy controller and how a company like Weirdly Wired could benefit from this discipline. Listed below are some of the interesting aspects that could be addressed in future research.

- **Further research of playfulness in form within the context of the Modsy controller.**
  Playfulness in form has briefly been discussed and a simple implementation has been evaluated within this thesis. This showed promising results since there was an alteration in the perceived playfulness of the controller. However, a lot more research could be performed into the implementation of visually playfully engaging elements in the design of the Modsy controller. Within the scope of this thesis there was not enough time to sufficiently analyse this aspect. Future research could determine which aspects people find especially visually engaging within the context of the Modsy controller.
- **Re-evaluate the current concept**
  Future research could re-evaluate the current concept. As discussed in chapter 7 of this report there could be some flaws in the evaluation method used to assess the currently created prototype. These flaws are mainly due to the direct comparison of the playful interaction with normal Modsy functionality. Future research could provide a better way to assess the enhancement of playfulness of the Modsy controller and show the potential of adding additional playful experiences.
- **Different implementation of current concept**
  The current concept is still very flexible and different implementation might be possible. Future work could assess the use of a different sub-concept as discussed during the specification phase in order to assess their effectiveness. A different sub-concept, other than the space ship game, could interact very differently with the Modsy controller and it could be interesting to evaluate these approaches.
- **Brand perception and playfulness**
  Not a lot of research could be found that discussed the perception of brands and their playfulness. If a company creates more playful products or services does this alter the perception that customers have of this company? For the company like Weirdly Wired that is interested in playfulness it might be useful to analyse how their customers will respond to more playful products. Future work could identify the potential in using playfulness as a company asset.

9 - References

Ableton. (2017, November 2). Ableton Live 10: New Device Visualisations on Push [Video]. YouTube. Accessed on: Mar. 16, 2021. Available: Ableton Live 10: New Device Visualisations on Push - YouTube

Ableton. (n.d.) Push, Music at your fingertips. Accessed on: Jul. 9, 2021. Available: https://www.ableton.com/en/push/

Adafruit. (2021). NeoPixel Stick - 8 x 5050 RGB LED with Integrated Drivers. [Webstore]. Accessed on: Jul. 9, 2021. Available: https://www.adafruit.com/product/1426

Apter, M. J. (1989). Reversal theory: a new approach to motivation, emotion and personality. Anuario de Psicologia. Vol. 42

Arrasvuori J., Korhonen, H., Boberg, M. (2010) Understanding playfulness: An overview of the revised playful experience (PLEX) framework. Proceedings of the 7th International Conference on Design and Emotion

Arrasvuori, J., Boberg, M., Holopainen, J., Korhonen, H., Lucero, A., Montola, M. (2011). Applying the PLEX Framework in Designing for Playfulness. Conference: DPPI'11 - Designing Pleasurable Products and Interfaces, Proceedings. doi: 10.1145/2347504.2347531.

Barnett. L. (1990). Playfulness: Definition, design, and measurement. Play & Culture, 1990, 3, 319-336

Bateson, P. (2014) Play, Playfulness, Creativity and Innovation. Animal Behavior and Cognition, 1(2), 99-112. doi: 10.12966/abc.05.02.2014

Bateson, P. (2015). Playfulness and creativity. Current Biology, Vol. 25, Issue 1 12-16. Doi: https://doi.org/10.1016/j.cub.2014.09.009

Bateson, P., Martin, P. (2013). Play, Playfulness, Creativity and Innovation

Bawiec, D. (2018). What Is Mix Automation? Everything You've Been Too Afraid to Ask. . Accessed on: Jul. 9, 2021. Available: https://www.izotope.com/en/learn/what-is-mix-automation.html

Baxmusic. (2021). Softube Console 1 mkII DAW controller. [Digital Image]. Softube Console 1 mkII kopen? | Bax Music (bax-shop.nl)

Bekoff, M. (1972), The Development of Social Interaction, Play, and Metacommunication in Mammals: An Ethological Perspective, The Quarterly Review of Biology, Vol. 47, No. 4, 412-434, Published by: The University of Chicago Press

Belyh, A. (2019). Brainstorming – Techniques for Idea Generation. Accessed on: Jul. 9, 2021. Available: https://www.cleverism.com/brainstorming-techniques-for-idea-generation/

Berkenbos, R.-J. (2021). Modsy tutorial: An engaging and easy to understand introductory tutorial. Unpublished Thesis

Blijlevens, J., Creusen, M. E. H., & Schoormans, J. P. L. (2009). How consumers perceive product appearance: The identification of three product appearance attributes. International Journal of Design, 3(3), 27-35.

Canva. (2021). How to create a beautiful moodboard. Accessed on: Jul. 9, 2021. Available: https://www.canva.com/learn/make-a-mood-board/

Chan, W. W. L., Ma, W. W. K. (2014). The Influence of Playfulness and Subject Involvement on Focused Attention when Using Social Media. Journal of Communication and Education, 16-27

Chang, C. (2013). Relationships between Playfulness and Creativity among Students Gifted in Mathematics and Science. Creative Education 04(02):101-109. DOI:10.4236/ce.2013.42015

Conna, C. (2016). A Comb Holds Nails in Place. [Digital image]. http://hackable.com/a-comb-holds-nails-in-place/

Conrad.nl (2021). Joy-it KY023JM Sensorkit Geschikt voor: Raspberry Pi, pcDuino, Banana Pi, Arduino 1 stuk(s). [Webstore]. Accessed on: Jul. 9, 2021. Available: Joy-it KY023JM Sensorkit Geschikt voor: Raspberry Pi, pcDuino, Banana Pi, Arduino 1 stuk(s) | Conrad.nl

Costello, B. M., Edmonds, E. A. (2007). A study in play, pleasure and interaction design. Conference: Proceedings of the 2007 International Conference on Designing Pleasurable Products and Interfaces. doi:10.1145/1314161.1314168.

Csepregi, G. (2013). On Musical Performance as Play. Nordic Journal of Aesthetics 23(46):96-114. Doi: 10.7146/nja.v23i46.16384

Csikszentmihalyi, M. (1975). Beyond Boredom and Anxiety. San Francisco : Jossey-Bass Publishers

Csikszentmihalyi, M. (2008). Flow, The Psychology of Optimal Experience. Publisher: Harper & Row

Diaz-Varela, A., Wright, L. H. V. (2019). Play for Adults: Play-based Approaches in Teacher Training. Scottish Educational Review 51(2). 132-136.

Endemol Shine Group (Producer). (2021). Lego Masters. [Television broadcast]. RTL Network

Engeser, S. (2012). Advances in Flow Research. Springer, New York. Doi: 10.1007/978-1-4614-2359-1,

Funkydesignspaces. (n.d.) PLEX Cards, Playful Experiences Cards. [PDF File]. http://www.funkydesignspaces.com/plex/

Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model. Simulation & gaming, 33(4), 441-467.

George, J. M. (2007). Creativity in Organisations, The Academy of Management Annals. 1:1, 439-277, Doi: 10.1080/078559814

Gray, Peter. (2017). What Exactly Is Play, and Why Is It Such a Powerful Vehicle for Learning?. Topics in Language Disorders. 37. 217-228. doi: 10.1097/TLD.0000000000000130.

Groos, K. (1898). The Play of Animals. New York : Appleton. Retrieved from: https://archive.org/details/playofanimals00groouoft/mode/2up

Guitard, P., Ferland, F., & Dutil, É. (2005). Toward a Better Understanding of Playfulness in Adults. OTJR: Occupation, Participation and Health, 25(1), 9–22. doi: https://doi.org/10.1177/153944920502500103

Hong, Z. (2012). A study for Playful product design. University of Auburn, Alabama

Huizinga, J. (1938). Homo Ludens: A study of the play-element in culture. Routledge & Kegan Paul London, Boston and Henley.

Human Motivation, 3rd ed., by Robert E. Franken, 1994

IKEA. (2021). BYGGLEK. [Digital Image]. https://www.ikea.com/nl/nl/p/bygglek-lego-r-stenen-201-delig-gemengde-kleuren-20436888/

Interface. (2020). MP Midi Controller. [Digital Image]. https://www.interface.nl/nieuws/artikel/3-24676/mp-midi-controller

Jackson, S. A., & Eklund, R. C. (2002). Assessing flow in physical activity: The Flow State Scale-2 and Dispositional Flow Scale-2. Journal of Sport & Exercise Psychology, 24(2), 133–150

Karras et al. (2019). Imagined by a GAN (generative adversarial network), StyleGAN2. Accessed on: Jul. 9, 2021. Available: https://thispersondoesnotexist.com/

Kim, B., (2015). Understanding Gamification. Vol. 51. No. 2. Library Technology Reports. ISSN 0024-2586

King, S., Chang, K. (2015). Understanding industrial design. O'Reilly Media, Inc. ISBN: 9781491920398

Klarkowski, M., Johnson, D., Wyeth, P., Smith, S., Phillips, C. (2015). Operationalising and measuring flow in video games. In Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction onOzCHI '15, ACM, Melbourne, Vic. 114-118

Korhonen, H., Montola, M., Arrasvuori, J. (2009). Understanding playful user experiences through digital games. Conference: International conference on designing pleasurable products and interfaces.

Kyriazos, T. A., Stalikas, A., Prassa, K., Galanakis, M., Flora, K., & Chatzilia, V. (2018). The Flow Short Scale (FSS) Dimensionality and What MIMIC Shows on Heterogeneity and Invariance. Psychology, 9, 1357-1382. https://doi.org/10.4236/psych.2018.96083

KyTary.nl. (2021). ABLETON PUSH 2. [Digital Image]. https://kytary.nl/ableton-push-2/HN160497/

LEGO education (n.d.). Retrieved March 16, 2021, from: Build Confidence with STEM Competitions | LEGO® Education

LEGO world (n.d.). Retrieved March 16, 2021, from: LEGO World 2019 – leukste uitje in de herfstvakantie

LEGO. (2021). LEGO Homepage. [Website]. Accessed on: Jul. 9, 2021. Available: Start | Officiële LEGO® winkel NL

Lieberman, J. N. (1977). Playfulness. New York: Academic Press.

Lonczak, H. S. (2020, December 10th). How to Measure Flow with Scales and Questionnaires. Retrieved from How to Measure Flow with Scales and Questionnaires (positivepsychology.com)

Lucero, A. Holopainen, J., Ollila, E., Suomela, R., Karapanos, E. (2013) The Playful Experiences (PLEX) Framework as a Guide for Expert Evaluation. ISBN: 978-1-4503-2192-1/13/09

Lucero, A., Arrasvuori, J. (2010). PLEX Cards: A source of inspiration when designing for playfulness. ACM International Conference Proceeding Series. doi: 10.1145/1823818.1823821

Mader, A. H., & Eggink, W. (2014). A Design Process for Creative Technology. In E. Bohemia, A. Eger, W. Eggink, A. Kovacevic, B. Parkinson, & W. Wits (Eds.), Proceedings of the 16th International conference on Engineering and Product Design, E&PDE 2014 (pp. 568-573). (E&PDE). The Design Society.

Mathijssen, O. Driel B. van, Berkenbos, R.-J. & L. Wintermans. (2020). Modsy Business Plan. Unpublished report

Merriam-Webster. (n.d.). State of mind. In Merriam-Webster.com dictionary. Retrieved April 7, 2021, from https://www.merriam-webster.com/dictionary/state%20of%20mind

Moog. (n.d.). Minimoog Model D. Accessed on: Jul. 9, 2021. Available: https://www.moogmusic.com/products/minimoog-model-d

Music store. (2021). Moog Minimoog Model D. [Digital Image]. https://www.musicstore.com/nl_OT/EUR/Moog-Minimoog-Model-D/art-SYN0005422-000

Native Instruments. (2021). OUR BEST MIDI KEYBOARDS. Accessed on: Jul. 9, 2021. Available: https://www.native-instruments.com/en/catalog/komplete/keyboards/

Native Instruments. (2021). This is NKS. [Digital Image]. This Is Nks (native-instruments.com)

Nayanathara, R. (2020). PACT Analysis. Accessed on: Jul. 9, 2021. Available: https://bootcamp.uxdesign.cc/pact-analysis-3ac5fbe8817

O'Brien T. (2020). Blipblox After Dark is a kid's synth for adults. It's still basically a toy, but that's not necessarily a bad thing. Retrieved March 16, 2021, from: Blipblox After Dark is a kid's synth for adults | Engadget

Pang, A.S.K. (2012, November 2). The downside of flow: Machine gambling. Strategy + Rest. The downside of flow: Machine gambling – Strategy and Rest

Patel, K. (2020). 5 tips to make ideation sketching approachable to all. Accessed on: Jul. 9, 2021. Available: https://uxdesign.cc/5-tips-to-make-ideation-sketching-approachable-to-all-9a9a23d2cdf2

Pierce, D. (2017). The Hot New Hip-Hop Producer Who Does Everything on His iPhone. Accessed on: 15/04/2021. Steve Lacy Produced That Hot Kendrick Lamar Track Using Only His iPhone | WIRED

Playtime Engineering. (2021). Meet the Blipblox. Accessed on: Jul. 9, 2021. Available: https://blipblox.com/

ProductPlan. (2021). MoSCoW Prioritization. Accessed on: Jul. 9, 2021. Available: https://www.productplan.com/glossary/moscow-prioritization/

Proyer, R. T. (2012). Examining playfulness in adults: Testing its correlates with personality, positive psychological functioning, goal aspirations, and multi-methodically assessed ingenuity. Psychological Test and Assessment Modeling. Vol. 54, 103-127

Proyer, R. T. (2013). The well-being of playful adults: Adult playfulness, subjective well-being, physical well-being, and the pursuit of enjoyable activities. European Journal of Humour Research 1(1), 84-98 doi: 10.5167/uzh-78008

Proyer, R. T., Gander, F., Braeuer, K., Chick, G. (2021). Can Playfulness be Stimulated? A Randomised Placebo-Controlled Online Playfulness Intervention Study on Effects on Trait Playfulness, Well-Being, and Depression. Applied Psychology Health and Well-Being 13, 129-151. doi: 10.1111/aphw.12220

Reid, D. (2014) A Model of Playfulness and Flow in Virtual Reality Interactions. Presence Teleoperators & Virtual Environments 13(4). 451-462. doi:10.1162/1054746041944777

Reverb. (2019, February 25). Ep20: Synth Sounds of Minimoog: Parliament, Pink Floyd, Dr. Dre & More | Reverb [Video]. YouTube. https://www.youtube.com/watch?v=33gZwkSUZRs

Rheinberg, F., Vollmeyer, R., & Engeser, S. (2003). Die Erfassung des Flow-Erlebens [The assessment of flow experience]. In J. Stiensmeier-Pelster & F. Rheinberg (Eds.), Diagnostik von Motivation und Selbstkonzept, 261–279

Rheinberg, F., Vollmeyer, R., & Engeser, S. (2003). Die Erfassung des Flow-Erlebens [The assessment of flow experience]. In J. Stiensmeier-Pelster & F. Rheinberg (Eds.), Diagnostik von Motivation und Selbstkonzept (pp. 261–279). Göttingen: Hogrefe.

Romanwave. (2020, November 16). Is it worth to buy PUSH 2 in the end of 2020? Pros & Cons. [Online forum post]. Reddit. https://www.reddit.com/r/ableton/comments/jvaf4i/is_it_worth_to_buy_push_2_in_the_end_of_2020_pros/

Samuelsson, I. P., Johansson, E. (2006). Play and learning—inseparable dimensions in preschool practice, Early Child Development and Care, 176:1. 47-65. doi: 10.1080/0300443042000302654

Sandelands, L., Ashford S. A., Dutton, J. E., (1983). Reconceptualizing of the overjustification effect: A template-matching approach. Motivation and Emotion. 7. 229-255. doi: 10.1007/BF00991675

Sandelands, Loyd E. (1988) The Effect of work and Play signals on task evaluation. Journal of Applied Social Psychology, Vol.18, Issue 12, 1032-1048. doi: https://doi.org/10.1111/j.1559-1816.1988.tb01191.x

Schiller, F. (1795). On the Aesthetic Education of Man. Dover Publications, Inc. Mineola, New York

Shahri, J. M. (2016). Playful engagements in product design: Developing a theoretical framework for ludo-aesthetic interactions in kitchen appliances. University of Edinburgh.

Sicart, M. (2014). Play Matters.The Mit Press. ISBN: 9780262027922

Singer, D. G., Golinkoff, R. M., & Hirsh-Pasek, K. (2006). Play = learning: How play motivates and enhances children's cognitive and social-emotional growth. Oxford University Press. doi: 10.1093/acprof:oso/9780195304381.001.0001

Smith, L. W. (2000). Stakeholder analysis: a pivotal practice of successful projects. Paper presented at Project Management Institute Annual Seminars & Symposium, Houston, TX. Newtown Square, PA: Project Management Institute.

Softube. (2021). CONSOLE 1 Sound. Workflow. Control. Accessed on: Jul. 9, 2021. Available: https://www.softube.com/console1

Softube. (2021). Console 1. Sound. Workflow. Control. [Digital Image]. Console 1 | Softube

Starbuck, W., Webster, J. (1991). When is play productive?. Accounting, Management and Information Technologies. 1. 71-90. 10.1016/0959-8022(91)90013-5

Tannis, D. J. (2012). Exploring play/playfulness and learning in adult and higher education classroom. The Pennsylvania State University, ProQuest Dissertations Publishing.

Teenage Engineering. (2021). the portable wonder synthesizer. Accessed on: Jul. 9, 2021. Available: https://teenage.engineering/products/op-1

Telekom Electronic Beats. (n.d.). Playlists [YouTube channel]. YouTube. Accessed on: March 16, 2021, from Ask The Producer: Steffi (Electronic Beats TV) - YouTube

Thomann. (2021-a). Native Instruments Komplete Kontrol S49 MK2. [Digital Image]. https://www.thomann.de/nl/native_instruments_komplete_kontrol_s49_mk2.htm

Thomann. (2021-b). Teenage Engineering OP-1. [Digital Image]. Teenage Engineering OP-1 – Thomann Nederland

Thomann. (2021-c). Playtime Engineering Blipblox. [Digital Image]. https://www.thomann.de/gb/playtime_engineering_blipblox.htm

Thomas, N. (n.d.) How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website. Accessed on: Jul. 9, 2021. Available: https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/

Tombolare. (2012). Teenage Engineering’s OP-1 Portable Synthesizer. [Digital Image]. https://c86.tumblr.com/post/25845019985/teenage-engineerings-op-1-portable-synthesizer

Townsend, J. D. Montoya, M. M. & Calantone, R. J. (2011). Form and Function: A Matter of Perspective. Journal of Product Innovation Management. Doi: 10.1111/j.1540-5885.2011.00804.x

UNICEF (2018). Learning through play, strengthening learning through play in early childhood educational programs.

University of Adelaine. (2014). Mind Mapping Writing Centre Learning Guide. Accessed on: Jul. 9, 2021. Available: Mind Mapping (adelaide.edu.au)

Usability.gov. (2020). System Usability Scale (SUS). . Accessed on: Jul. 9, 2021. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html#:~:text=The%20System%20Usability%20Scale%20(SUS,Strongly%20agree%20to%20Strongly%20disagree.

Webster, J., Trevino, L.K., Ryan, L. (1993). The Dimensionality and Correlates of Flow in Human-Computer Interaction. Computers in Human Behavior, Vol. 9, 411-426. doi: https://doi.org/10.1016/0747-5632(93)90032-N

White Sea Studio. (2019, February 21). REVIEW: Softtube Console 1 [Video]. YouTube. REVIEW: Softube Console 1 - YouTube

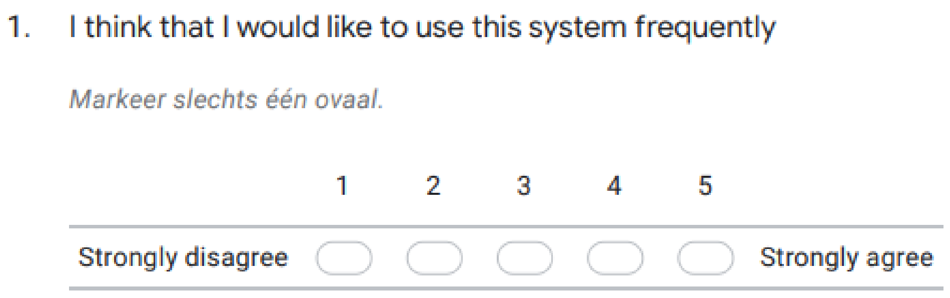
Wignall, N. (2019). Psychological Benefits of Playfulness for Adults. Accessed on: Mar. 16, 2021. Available: https://nickwignall.com/benefits-of-playfulness/

Yager, S. E., Kappelman, L. A., Maples, G. A., Prybutok, V. R. (1997). Microcomputer Playfulness: Stable or Dynamic Trait? The database for Advances in Information Systems, Vol. 28, No. 2

Zosh J.M., Hopkins, E.J., Jensen, H., Liu, C., Neale, D., Hirsh-Pasek, K., Solis, S.L., Whitebread, D. (2017). Learning through play: a review of the evidence. White Paper, LEGO foundation. ISBN: 978-87-999589-1-7

10 – Appendix

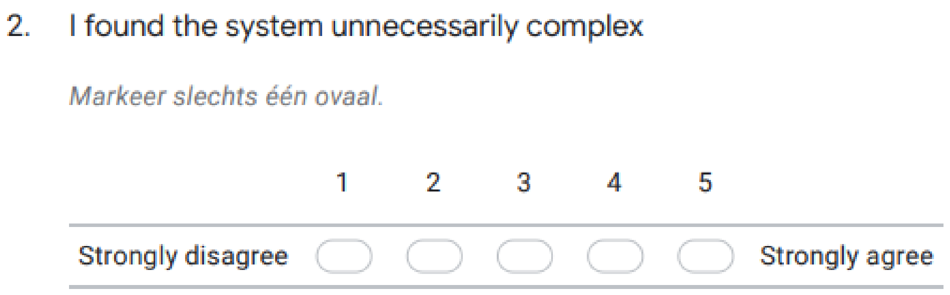
A – SUS questionnaire



1. I think that I would like to use this system frequently

Markeer slechts één ovaal.

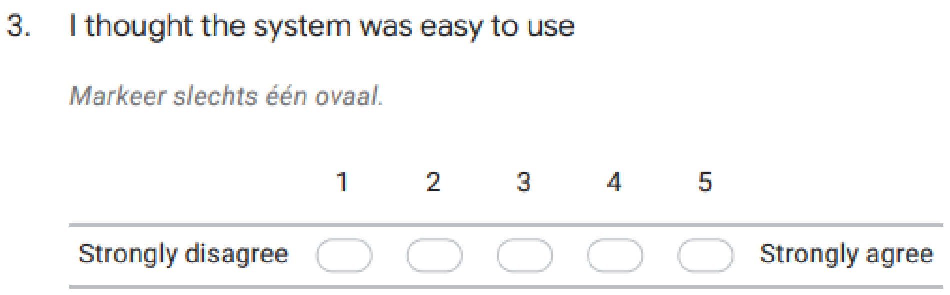
	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree



2. I found the system unnecessarily complex

Markeer slechts één ovaal.

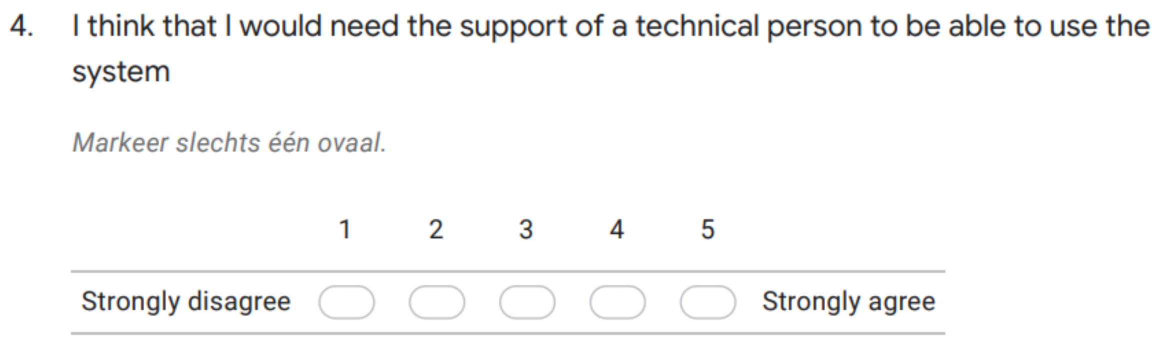
	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree



3. I thought the system was easy to use

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree



4. I think that I would need the support of a technical person to be able to use the system

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

5. I found the various functions in this system were well integrated

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

6. I thought there was too much inconsistency in this system

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

7. I would imagine that most people would learn to use this system very quickly

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

8. I found the system very cumbersome to use

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

9. I felt very confident using the system

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

10. I needed to learn a lot of things before I could get going with this system

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

B – Flow Short Scale questionnaire

1. I feel just the right amount of challenge

Markeer slechts één ovaal.

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much

2. My thoughts/activities run fluidly and smoothly

Markeer slechts één ovaal.

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much

3. I do not notice time passing

Markeer slechts één ovaal.

	1	2	3	4	5	
Not at all	◯	◯	◯	◯	◯	Very much

4. I have no difficulty concentrating

Markeer slechts één ovaal.

	1	2	3	4	5	
Not at all	◯	◯	◯	◯	◯	Very much

5. My mind is completely clear

Markeer slechts één ovaal.

	1	2	3	4	5	
Not at all	◯	◯	◯	◯	◯	Very much

6. I am totally absorbed in what I am doing

Markeer slechts één ovaal.

	1	2	3	4	5	
Not at all	◯	◯	◯	◯	◯	Very much

7. The right thoughts/movements occur of their own accord

Markeer slechts één ovaal.

	1	2	3	4	5	
Not at all	◯	◯	◯	◯	◯	Very much

8. I know what I have to do each step of the way

Markeer slechts één ovaal.

	1	2	3	4	5	
Not at all	◯	◯	◯	◯	◯	Very much

9. I feel that I have everything under control

Markeer slechts één ovaal.

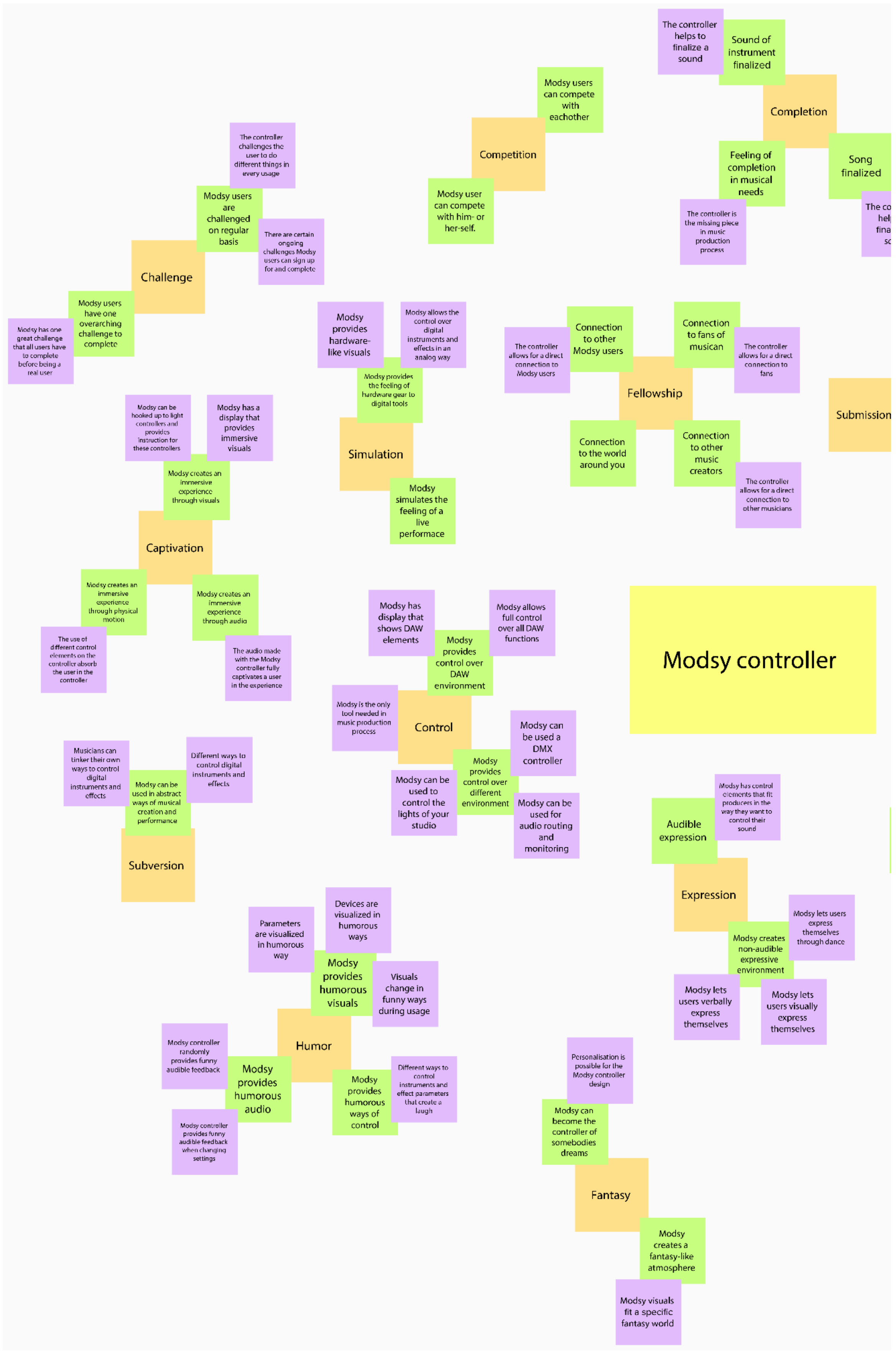
	1	2	3	4	5	
Not at all	◯	◯	◯	◯	◯	Very much

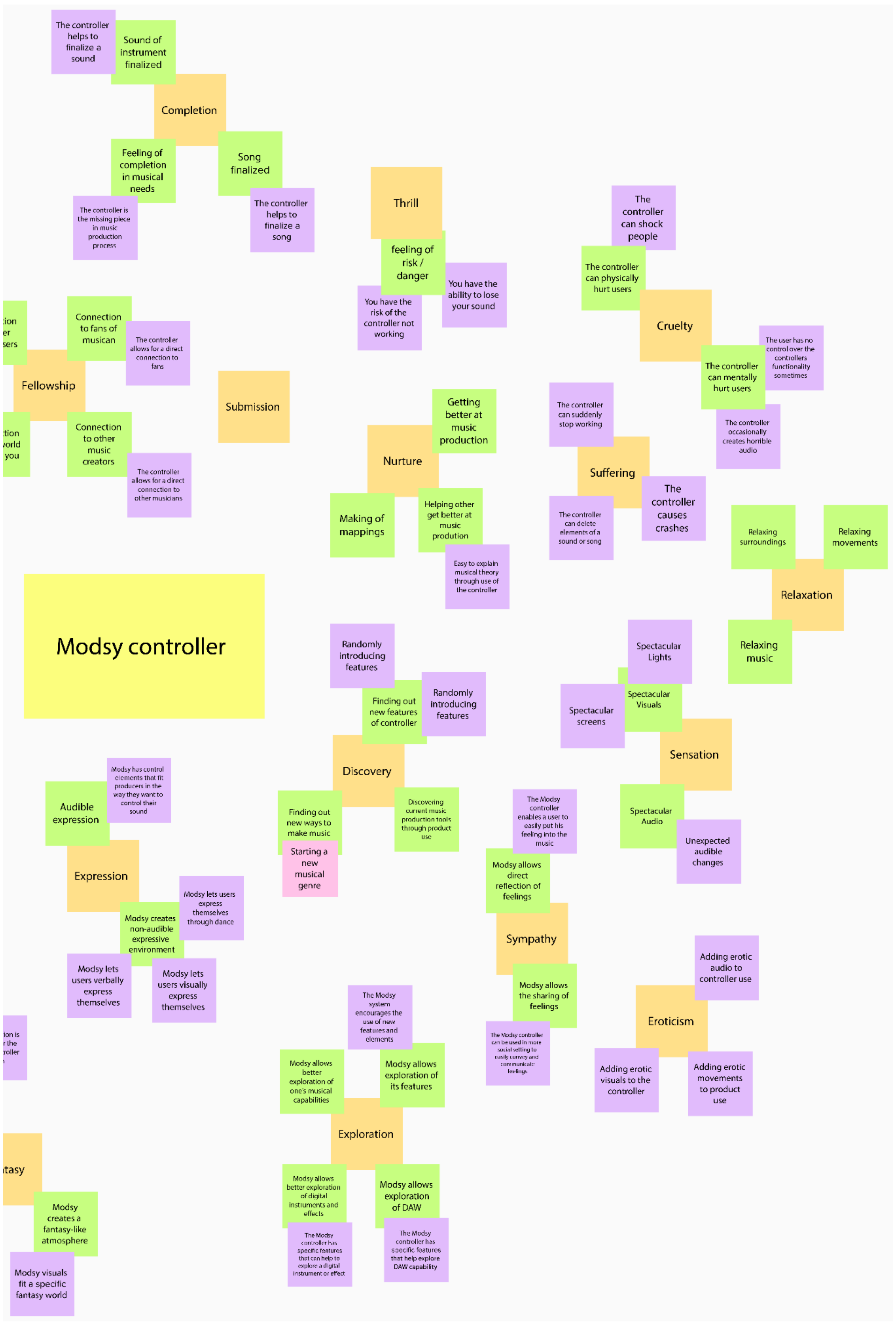
10. I am completely lost in thought

Markeer slechts één ovaal.

	1	2	3	4	5	
Not at all	◯	◯	◯	◯	◯	Very much

C – Mind map ideation


The controller helps to finalize a sound
Sound of instrument finalized
Completion
Feeling of completion in musical needs
Song finalized
The controller is the missing piece in music production process
Modsy users can compete with eachother
Competition
Modsy user can compete with him- or her-self.
The controller challenges the user to do different things in every usage
Modsy users are challenged on regular basis
There are certain ongoing challenges Modsy users can sign up for and complete
Challenge
Modsy users have one overarching challenge to complete
Modsy has one great challenge that all users have to complete before being a real user
Modsy provides hardware-like visuals
Modsy allows the control over digital instruments and effects in an analog way
Modsy provides the feeling of hardware gear to digital tools
Simulation
Modsy simulates the feeling of a live performace
The controller allows for a direct connection to Modsy users
Connection to other Modsy users
Connection to fans of musican
The controller allows for a direct connection to fans
Fellowship
Connection to the world around you
Connection to other music creators
The controller allows for a direct connection to other musicians
Submission
Modsy can be hooked up to light controllers and provides instruction for these controllers
Modsy has a display that provides immersive visuals
Modsy creates an immersive experience through visuals
Captivation
Modsy creates an immersive experience through physical motion
Modsy creates an immersive experience through audio
The use of different control elements on the controller absorb the user in the controller
The audio made with the Modsy controller fully captivates a user in the experience
Modsy has display that shows DAW elements
Modsy allows full control over all DAW functions
Modsy provides control over DAW environment
Modsy is the only tool needed in music production process
Control
Modsy can be used a DMX controller
Modsy provides control over different environment
Modsy can be used to control the lights of your studio
Modsy can be used for audio routing and monitoring
Modsy controller
Musicians can tinker their own ways to control digital instruments and effects
Different ways to control digital instruments and effects
Modsy can be used in abstract ways of musical creation and performance
Subversion
Modsy has control elements that fit producers in the way they want to control their sound
Audible expression
Expression
Modsy lets users express themselves through dance
Modsy creates non-audible expressive environment
Modsy lets users verbally express themselves
Modsy lets users visually express themselves
Parameters are visualized in humorous way
Devices are visualized in humorous ways
Modsy provides humorous visuals
Visuals change in funny ways during usage
Humor
Modsy controller randomly provides funny audible feedback
Modsy provides humorous audio
Modsy controller provides funny audible feedback when changing settings
Modsy provides humorous ways of control
Different ways to control instruments and effect parameters that create a laugh
Personalisation is possible for the Modsy controller design
Modsy can become the controller of somebodies dreams
Fantasy
Modsy creates a fantasy-like atmosphere
Modsy visuals fit a specific fantasy world


Modsy controller
The controller helps to finalize a sound
Sound of instrument finalized
Completion
Feeling of completion in musical needs
The controller is the missing piece in music production process
Song finalized
The controller helps to finalize a song
tion er sers
Connection to fans of musican
The controller allows for a direct connection to fans
Fellowship
tion vorld you
Connection to other music creators
The controller allows for a direct connection to other musicians
Submission
Thrill
feeling of risk / danger
You have the risk of the controller not working
You have the ability to lose your sound
The controller can shock people
The controller can physically hurt users
Cruelty
The controller can mentally hurt users
The user has no control over the controllers functionality sometimes
The controller occasionally creates horrible audio
The controller can suddenly stop working
Suffering
The controller can delete elements of a sound or song
The controller causes crashes
Nurture
Getting better at music production
Making of mappings
Helping other get better at music prodution
Easy to explain musical theory through use of the controller
Relaxing surroundings
Relaxing movements
Relaxation
Relaxing music
Randomly introducing features
Finding out new features of controller
Randomly introducing features
Discovery
Finding out new ways to make music
Starting a new musical genre
Discovering current music production tools through product use
Spectacular Lights
Spectacular screens
Spectacular Visuals
Sensation
Spectacular Audio
Unexpected audible changes
Audible expression
Modsy has control elements that fit producers in the way they want to control their sound
Expression
Modsy creates non-audible expressive environment
Modsy lets users express themselves through dance
Modsy lets users verbally express themselves
Modsy lets users visually express themselves
The Modsy controller enables a user to easily put his feeling into the music
Modsy allows direct reflection of feelings
Sympathy
Modsy allows the sharing of feelings
The Modsy controller can be used in more social setting to easily convey and communicate feelings
Adding erotic audio to controller use
Eroticism
Adding erotic visuals to the controller
Adding erotic movements to product use
ion is r the roller
The Modsy system encourages the use of new features and elements
Modsy allows better exploration of one's musical capabilities
Modsy allows exploration of its features
Exploration
Modsy allows better exploration of digital instruments and effects
Modsy allows exploration of DAW
The Modsy controller has specific features that can help to explore a digital instrument or effect
The Modsy controller has specific features that help explore DAW capability
tasy
Modsy creates a fantasy-like atmosphere
Modsy visuals fit a specific fantasy world

D – PLEX concept assessment

Concepts	Captivation	Challenge	Control	Discovery	Exploration	Fantasy	Humour	Relaxation	Sensational	Simulation
Entity inside the Modsy	0	0	0	1	0	2	2	0	1	0
Virus in Modsy	0	0	0	1	0	2	1	0	2	1
Bomb in Modsy	0	0	1	1	0	1	1	0	2	1
Pong	1	1	1	1	1	1	1	0	1	0
Tetris	1	1	1	1	1	1	1	0	1	0
Pacman	1	1	1	1	1	1	1	0	1	0
Snake	1	1	1	1	1	1	1	0	1	0
DJ controller emulation	1	0	1	0	1	1	1	0	2	1
Radio Receiving	0	1	1	1	1	2	1	0	1	2
Gold digging	1	2	1	0	1	1	1	0	1	0
Platform game	1	1	1	0	1	1	1	0	1	0
Safe cracking	1	2	1	1	1	2	1	0		
Space ship controller	2	1	2	2	2	2	1	0	2	2
Battleship	1	1	1	0	1	1	1	0	1	1

Concepts	Submission	Sympathy	Thrill	Competitio	Completion	Cruelty	Eroticism	Fellowship	Nurture	Suffering	Subversion
Entity inside the Modsy	1	0	1	0	0	0	0	1	1	0	
Virus in Modsy	1	0	2	0	0	0	0	0	0	1	1
Bomb in Modsy	1	0	2	0	0	0	0	0	0	1	1
Pong	0	0	0	2	1	0	0	0	0	0	0
Tetris	0	0	0	2	1	0	0	0	0	0	0
Pacman	0	0	0	2	1	0	0	0	0	0	0
Snake	0	0	0	2	1	0	0	0	0	0	0
DJ controller emulation	0	1	0	0	0	0	0	0	0	0	0
Radio Receiving	0	0	0	1	1	0	0	0	0	0	0
Gold digging	0	0	0	1	0	0	0	0	0	0	0
Platform game	0	0	0	1	1	0	0	0	0	0	0
Safe cracking	0	0	1	1	1	0	0	0	0	0	0
Space ship controller	0	0	1	0	0	0	0	0	1	0	1
Battleship	0	0	1	1	1	0	0	0	0	0	0

E – Form assessment cards

Old-Fashioned

Classical

Oldish

Kitsch
Retro
Functional
Boring
Plain
Playful
Funny
Unusual
Minimalistic

Sleek

Futuristic

Modern

F – PLEX analysis state of the art

BlipBlox

PLEX category	Explanation	Points (0 - 3)
Captivation	Users can get sortof lost	2
Challenge	Yes challenge of knowing what is what	3
Competition	No	0
Completion	No	0
Control	Some sense of control	2
Cruelty	no	0
Discovery	High	3
Eroticism	no	0
Exploration	High	3
Fantasy	Little fantasy	1
Fellowship	No	0
Humor	Some humourous creations	2
Nurture	no	0
Relaxation	not really	1
Sensational	Yes very much	3
Simulation	no	0
Submission	no	0
Subversion	no	0
Suffering	no	0
Sympathy	no	0
Thrill	no	0

Model D

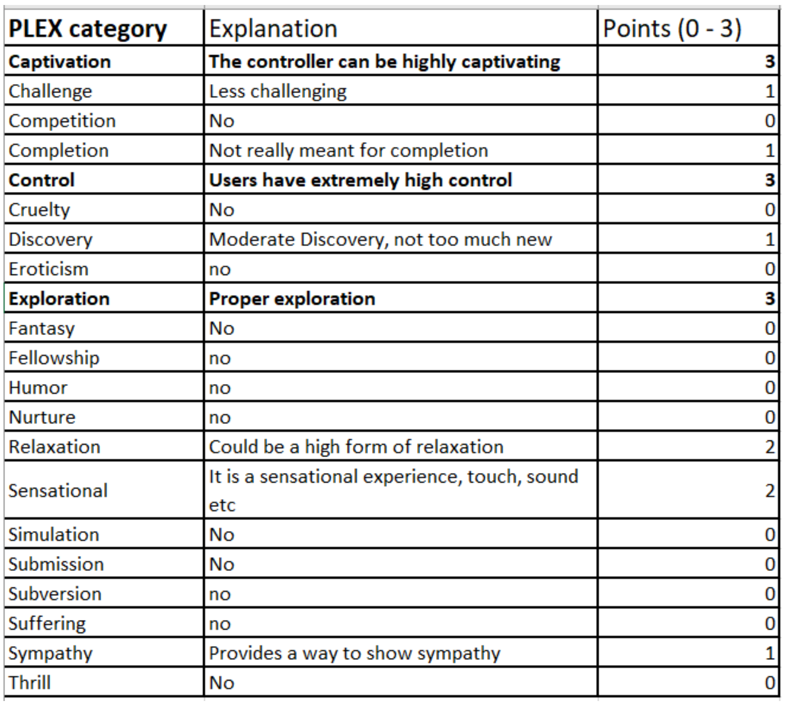
PLEX category	Explanation	Points (0 - 3)
Captivation	Very Captivating	3
Challenge	Fair amount of challenge	2
Competition	No	0
Completion	No	0
Control	Yes much so	2
Cruelty	No	0
Discovery	Yes much so	3
Eroticism	No	0
Exploration	Yes much so	3
Fantasy	Little bit yes	1
Fellowship	No	0
Humor	No	0
Nurture	no	0
Relaxation	Could be	2
Sensational	To some degree	1
Simulation	No	0
Submission	No	0
Subversion	No	0
Suffering	No	0
Sympathy	No	0
Thrill	Some thrill, old synth	1

TE OP-1

PLEX category	**Explanation**	**Points (0 - 3)**
Captivation	**Highly captivating**	**3**
Challenge	Not really	0
Competition	Not really	0
Completion	Not really	0
Control	There is a proper sense of control but not completely	1
Cruelty	Not really	0
Discovery	**Very much so**	**3**
Eroticism	Not really	0
Exploration	**Very much so**	**3**
Fantasy	**Yes**	**2**
Fellowship	Not really	0
Humor	**Very much so**	**2**
Nurture	Not really	0
Relaxation	Yes could be	2
Sensational	**Yes could be**	**3**
Simulation	Not really	0
Submission	Not really	0
Subversion	Not really	0
Suffering	Not really	0
Sympathy	Could be but not really	1
Thrill	Not really	0

LEGO

PLEX category	**Explanation**	**Points (0 - 3)**
Captivation	**Can be completely captivating**	**3**
Challenge	Could test ones ability	2
Competition	Could be competitive but not main aim	1
Completion	No, not finished quickly	1
Control	Some sense of control depends	2
Cruelty	No	0
Discovery	**Very much so**	**3**
Eroticism	no	0
Exploration	**Very much so**	**3**
Fantasy	**Very much so**	**3**
Fellowship	Not really	1
Humor	Could be very much	1
Nurture	No	0
Relaxation	No	0
Sensational	Midly sensational	1
Simulation	Could be	2
Submission	not really	0
Subversion	no	0
Suffering	no	0
Sympathy	no	0
Thrill	no	0



PLEX category	Explanation	Points (0 - 3)
Captivation	**The controller can be highly captivating**	**3**
Challenge	Less challenging	1
Competition	No	0
Completion	Not really meant for completion	1
Control	**Users have extremely high control**	**3**
Cruelty	No	0
Discovery	Moderate Discovery, not too much new	1
Eroticism	no	0
Exploration	**Proper exploration**	**3**
Fantasy	No	0
Fellowship	no	0
Humor	no	0
Nurture	no	0
Relaxation	Could be a high form of relaxation	2
Sensational	It is a sensational experience, touch, sound etc	2
Simulation	No	0
Submission	No	0
Subversion	no	0
Suffering	no	0
Sympathy	Provides a way to show sympathy	1
Thrill	No	0

G – Consent form

Informed consent – Playfulness research

About

This informed consent is related to the evaluation method of a thesis about the playfulness of the Modsy controller. This research assesses whether playfulness can be used as a good tool in design and whether this can improve the overall experience a user has with the product.

You give permission to participate in a user test in which statements and information can be used for this research. More information about this research can be found in the information brochure.

Lead researcher: Olivier Mathijssen

Supervisor: Wouter Eggink

Contact information

For any questions, you can contact the researcher that is present at the user test, Olivier Mathijssen (o.p.mathijssen@student.utwente.nl) or the supervisor Wouter Eggink (w.eggink@utwente.nl). You can also contact the Ethics Committee (ethicscommittee-cis@utwente.nl). This committee consists of independent experts from the university and is available for questions and complaints regarding this research.

Research: Playfulness of Modsy controller

- ☐ I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions or perform the tasks and I can withdraw from the study at any time, without having to give a reason.
- ☐ I understand the reason for this research, methods are explained and I have had the option to ask questions.
- ☐ I hereby give upfront consent to participate in this research
- ☐ I give permission for the use of my statements during this user test for research purposes
- ☐ I give permission for the making of a video recording during the user test for research purposes.

The video recording and transcriptions of the interview will be only accessible for the main researcher, Olivier Mathijssen. The video will be stored and anonymously processed according to the GDPR guidelines. All data will be stored for a minimum of 10 years, but in suiting cases for an undetermined time, appropriate to the current guidelines from the Vereniging van Universiteiten (VSNU).

Date: Place:

Name: Paragraph participant:

H – Information brochure

Information brochure – Playfulness research

Background

The Modsy controller is a new type of music production controller. The controller can provide more analogue feeling over different digital instruments and effects. Within music production, play and playfulness are very important to consider. More playful interaction can result in more creativity, fun, and a better learning experience. For a new controller entering the market, it is important to consider the aspect of playfulness and see how this can influence the overall user experience.

The goal of this research is to assess the effect that playfulness can have on the Modsy controller. In this research, a more playful Modsy controller will be created and evaluated through user tests. This brochure relates to these user tests.

Research Procedure

Through the means of a user test feedback will be gathered to analyse whether a more playful design can help to improve the overall user experience with a Modsy controller.

The research will include the use of a theoretically more playful Modsy controller in normal circumstances in order to test all features and see how design changes influence user interaction with this controller. During these user tests, the researcher will take notes and make a video recording.

This information brochure will explain what it means for you to partake in this research. If you have any further questions, don't hesitate to contact Olivier Mathijssen (o.p.mathijssen@student.utwente.nl).

Participation

Participation is entirely voluntary and you can quit this research at any moment without stating the reason why. Permission for participation only has to be granted once.

What happens during the test?

During the test, a participant is asked to perform specific tasks using the Modsy controller. This way the interaction with the controller can be observed. This is meant to resemble a normal music production process and follows some standard music production interactions, like controlling an instrument or effect. This will also include the use of a special mode in which playfulness is the main focus.

After the Modsy controller has been used, the researcher has both a short semi-structured interview and questionnaire prepared for the participant. These question will relate to the experienced playfulness and overall experience of the Modsy controller.

Which data will be collected?

The user test will be visually and audibly recorded to optimally utilize and analyze the user interaction with the product. Next to this, the researcher will make digital notes during the interview and use the data of the questionnaire. Statements made during the interview could be used to support the research outcome. These statements will be used anonymously.

How will the data be stored?

The video data will be stored on an encrypted hard drive of the researcher. Next to this, the statements made during the interview and filled-in questionnaire are scanned and later destroyed, the digital files are saved on an encrypted hard drive as well.

Who has access to the data?

The video recordings and transcriptions of the test will only be accessible to the lead researcher, Olivier Mathijssen.

Will any personal data be made public?

The video material will not be publicly displayed. Statements made during the interview could be anonymously used as a source in the research. Some statements could be included during the publication of the thesis.

More information and independent advise

Would you like independent advice about participating in this research or do you have a complaint? Then you can contact the Ethics Committee (ethicscommittee-cis@utwente.nl). This committee consists of independent experts from the university and is available for questions and complaints regarding this research.

For any questions, you can contact the researcher, Olivier Mathijssen (o.p.mathijssen@student.utwente.nl) or the supervisor Wouter Eggink (w.eggink@utwente.nl).

I – Processing program code

```
//Playful game interaction for the Modsy controller
//
//By Olivier Mathijssen
//05-07-2021
//
//Developed as part of the Bachelor Thesis the Playfulness of Modsy

import processing.serial.*;
Serial playful_extention_Port;  // Create object from Serial class
Serial modsyPort;

HealthBar health_bar;
HealthBar power_bar;

int spaceship_x;
int spaceship_y;
int[] spaceship_hitbox_size = {128, 32}; //The size of the box that makes up the spaceship hitbox, X,Y
int[] hitbox_spaceship = {0, 0, 0, 0}; //

char[] spawned = {'n', 'n', 'n', 'n'};  //'n' = nothing, 's' = spaceship, 'a' = astroid //
int[] spawned_xpos = {1024, 1024, 1024, 1024}; //
int[] spawned_ypos = {32, 32, 32, 32}; //
int[] spawned_speed = {0, 0, 0, 0}; //

long time_now_enemyship;
long time_now_astroid;
int timing_enemyship = 10000; //This timing could be made more dynamic
int timing_astroid = 7200; //This timing could be made more dynamic

int[] speed_ranges = {9, 20}; //The speed will be determined by a random function with these two
boundries
```

```
int[] spawned_y_ranges = {0, 256}; //The boundries in which elements can be spawned along the y
axis


int[] enemyship_hitbox_size = {128, 32}; //The size of the box that makes up the enemyship hitbox,
X,Y

int[] astroid_hitbox_size = {100, 50}; //The size of the box that makes up the astroid hitbox, X,Y


int[] hitbox_spawned = {1000, 1000, 1000, 1000}; //One hitbox array that can be used during the
Hitbox calculations, L,R,T,B


//Bullet code ---------------------- 10 bullets max

char[] present_bullets = {'n', 'n', 'n', 'n', 'n', 'n', 'n', 'n', 'n', 'n'};  //'n' = nothing, 'b' = bullet

int[] xpos_enemybullets = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};  //Max ten bullets

int[] ypos_enemybullets = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};   //

int[] bullet_speed = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

int[] hitbox_bullet = {0, 0, 0, 0}; //One hitbox array that can be used during the Hitbox calculations

int[] bullet_hitbox_size = {5, 5}; //The size of the box that makes up the bullet hitbox, also for
friendly bullets


//Spaceship bullets (friendly) ----------------------- 10 bullets max

char[] friendly_bullets = {'n', 'n', 'n', 'n', 'n', 'n', 'n', 'n', 'n', 'n'};  //'n' = nothing, 'b' = bullet

int[] xpos_friendlybullets = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};  //Max ten bullets

int[] ypos_friendlybullets = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};   //

int[] friendly_hitbox_bullet = {0, 0, 0, 0}; //One hitbox array that can be used during the Hitbox
calculations

int friendly_bullet_speed = 50;


//Overall timing variables

long time_now_refresh;

int timing_refresh = 80;//0.08Hz

boolean run_game = false;

boolean first_fix = true;

boolean first_reload = true;
```

```
long start_time; //Time will be set at start of the game

int total_game_duration = 120000; //Game should be played for 3 minutes. > 3* 60 * 1000 =
120000ms

boolean end_animation = false;

int end_earth_x = 1280;

int end_earth_y = 0; //This should be changed


//Elements to be used during the game.

int fireButton;

int shieldButton;

int reload_parameter;

int fix_parameter;


boolean set_shield = false;

int timing_shield = 2000;

long timing_now_shield;

boolean shield_expand = true;

int shield_diameter = 90;


//For the timing of reload and fix actions

long time_now_reload = 0;

long time_now_fix = 0;

int reload_timing;

int fix_timing;
//If one of the parameter is fixed or reloaded in time this positively affects the bars, else negatively

boolean fix_intime = false;

boolean reload_intime = false;


//Timing of popup in screen

long timing_now_popup = 0;

int timing_popup = 1000;

String popup_message = "Spacecruise started";
```

```
long timing_now_popup_above = 0;

int timing_popup_above = 2000;

String popup_message_2 = "Spacecruise started";


int[] reload_timing_boundries = {3000, 10000};

int[] fix_timing_boundries = {4000, 10000};


PImage spaceship;

PImage background;

PImage endearth;


//Array of images

PImage enemyship_images[];

PImage astroid_images[];

PImage bullet_images[];

PImage friendly_bullets_img[];


//For health and power bar

int health;

int power;


boolean enable_port_extention;

boolean enable_port_modsy;


//Excluded parameters

boolean echoctrl = false;

int[] echo_excluded_par = {3, 6, 7, 8, 11, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 29, 30,
31, 32 };

int[] juno_excluded_par = {8, 15, 16, 26, 27, 28};

int[] echo_non_excluded = {1, 2, 4, 5, 9, 10, 12, 20, 28};

int[] juno_non_excluded = {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21, 22, 23, 24, 25,
29, 30, 31, 32};
```

```
void setup() {

  size(1280, 320);
  background(0);

  printArray(Serial.list());

  enable_port_extention = false;
  enable_port_modsy = false;

  if (enable_port_extention == true) {
    String playful_extension = Serial.list()[1]; //change the 0 to a 1 or 2 etc. to match your port
    playful_extention_Port = new Serial(this, playful_extension, 9600);
  }

  if (enable_port_modsy == true) {
    String modsy_controller = Serial.list()[0]; //change the 0 to a 1  or 2 etc. to match your port
    modsyPort = new Serial(this, modsy_controller, 9600);
  }

  //To set the arduino leds
  health_bar = new HealthBar(width-180, color(154, 205, 50));
  power_bar = new HealthBar(width-100, color(255, 255, 51));
  health = 100;
  power = 80;

  if (enable_port_extention == true) {
    set_game_bars();
  }

  assignSpaceShipFunctions();
```

```
enemyship_images = new PImage[4];
astroid_images = new PImage[4];
bullet_images = new PImage[10];
friendly_bullets_img = new PImage[10];

spaceship = loadImage("spaceship.png");
endearth = loadImage("earth.png");

for (int i = 0; i < spawned.length; i++) {
  enemyship_images[i] = loadImage("enemyship.png");
  astroid_images[i] = loadImage("asteroid.png");
}

for (int j = 0; j < present_bullets.length; j++) {
  bullet_images[j] = loadImage("Bullet-07.png");
}

for (int s = 0; s < friendly_bullets.length; s++) {
  friendly_bullets_img[s] = loadImage("bullet_friendly.png");
}

//Set the values of the spaceship
spaceship_x = 50; //Approximatly half of the x of the first display (128)
spaceship_y = 70; //Approximalty half of the y of the first display (32)

//Set the supposed spawned elements initially
time_now_enemyship = millis();
time_now_astroid = millis();
time_now_refresh = millis();

//Total time of the game
start_time = millis();
```

```
}

void draw() {
 //spawn_astroid_ship_timer(); //This method spawns the ships and astroids when time is due
 background(255);

 if (run_game == true) {

   read_serial();

   draw_elements();

   if (millis() >= time_now_refresh + timing_refresh) {
     time_now_refresh = millis();

     spawn_astroid_ship_timer(); //This method spawns the ships and astroids when time is due

     move_spawned(); //This method moves all the spawned elements along the Modsy

     timedShipInteraction(); //This method is used for the reload and fix triggers.
   }
 } else {
   //Give start screen
   fill(220, 220, 220);
   noStroke();
   rect(0, 0, width, height);
   fill(0);
   textSize(24);
   textAlign(CENTER);
   text("Press s to start the game", width/2, height/2);

   time_now_enemyship = millis();
```

```
    time_now_astroid = millis();
    time_now_refresh = millis();

    //Total time of the game
    start_time = millis();
  }
}

void read_serial() {
  //For serial port
  String incoming_msg_p1 = "";
  String incoming_msg_p2 = "";

  if (enable_port_extention == true) {
    if ( playful_extention_Port.available() > 0) {  // If data is available,
      incoming_msg_p1 = playful_extention_Port.readStringUntil('\n');         // read it and store it in val
      println(incoming_msg_p1); //print it out in the console

      if (incoming_msg_p1 != null) {
        if (incoming_msg_p1.charAt(0) == 'j') {
          switch (incoming_msg_p1.charAt(1)) {
          case 'u':
            if (spaceship_y > 0) {
              spaceship_y -= 8;
            }
            break;
          case 'd':
            if (spaceship_y < 300) {
              spaceship_y += 8;
            }
            break;
          case 'l':
```

```
          if (spaceship_x > 0) {
            spaceship_x -= 8;
          }
          break;
        case 'r':
          if (spaceship_x < 500) {
            spaceship_x += 8;
          }
          break;
        default:
          print("Default case");
          break;
        }
      }
    }
  }
}

if (enable_port_modsy == true) {
  if ( modsyPort.available() > 0) {  // If data is available
    incoming_msg_p2 = modsyPort.readStringUntil('\n');         // read it and store it in val
    println(incoming_msg_p2); //print it out in the console


    if (incoming_msg_p2 != null) {


      if (incoming_msg_p2.charAt(0) == 'p') {
        switch (incoming_msg_p2.charAt(1)) {
        case 'r':
          reload_intime = true;
          visualize_parameter("Fueled in time!", 't'); //Does not work right now, the other element
draws over this.
          power += int(random(2, 5));
          if (power >= 100) {
            power = 100;
          }
          power_bar.update_bar(power);
          break;
        case 'f':
          fix_intime = true;
          visualize_parameter("Fixed in time!", 't'); //Does not work right now, the other element
draws over this.
          health += int(random(2, 5));
          if (health >= 100) {
            health = 100;
          }
          health_bar.update_bar(health);


          break;
        }
      } else if (incoming_msg_p2.charAt(0) == 'g') {
        //g for gun
        spaceship_shoots();
      } else if (incoming_msg_p2.charAt(0) == 's') {
        //s for shield
        spaceship_shield();
      }
    }
  }
 }
}


void set_game_bars() {
  println("Set game Bars");


  char health_level = '0';
  char power_level = '0';
```

```
if (health >= 0 && health <= 13) {
  health_level = '1';
} else if (health > 13 && health <= 25) {
  health_level = '2';
} else if (health > 25 && health <= 38) {
  health_level = '3';
} else if (health > 38 && health <= 50) {
  health_level = '4';
} else if (health > 50 && health <= 63) {
  health_level = '5';
} else if (health > 63 && health <= 75) {
  health_level = '6';
} else if (health > 75 && health <= 88) {
  health_level = '7';
} else if (health > 88 && health <= 100) {
  health_level = '8';
}

if (power >= 0 && power <= 13) {
  power_level = '1';
} else if (power > 13 && power <= 25) {
  power_level = '2';
} else if (power > 25 && power <= 38) {
  power_level = '3';
} else if (power > 38 && power <= 50) {
  power_level = '4';
} else if (power > 50 && power <= 63) {
  power_level = '5';
} else if (power > 63 && power <= 75) {
  power_level = '6';
} else if (power > 75 && power <= 88) {
    power_level = '7';
  } else if (power > 88 && power <= 100) {
    power_level = '8';
  }

  //For the health message
  playful_extention_Port.write('<'); //to start the message
  playful_extention_Port.write('h'); //to start the message
  playful_extention_Port.write(health_level);
  playful_extention_Port.write('>'); //close the message

  playful_extention_Port.write('<'); //to start the message
  playful_extention_Port.write('p'); //to start the message
  playful_extention_Port.write(power_level);
  playful_extention_Port.write('>'); //close the message
}

void draw_elements() {

  if (millis() >= start_time + total_game_duration) {
    end_animation = true;
  }

  spaceship.resize(128, 32);
  image(spaceship, spaceship_x, spaceship_y);

  if (set_shield == true && millis() < timing_now_shield + timing_shield) {
    //For the shield
    if (shield_diameter > 100) {
      shield_expand = false;
    } else if (shield_diameter < 80) {
      shield_expand = true;
    }
    if (shield_expand == true) {
      shield_diameter += 1;
    }
    if (shield_expand == false) {
      shield_diameter -= 1;
    }

    fill(2, 2, 200, 90);
    circle(spaceship_x + 90, spaceship_y + 16, shield_diameter);
  }

  //Debuging hitboxes
  //rect(hitbox_spaceship[0], hitbox_spaceship[2], spaceship_hitbox_size[0],
spaceship_hitbox_size[1]);

  //Spawn ships and stuff
  for (int i = 0; i < spawned.length; i++) {

    if (spawned[i] == 'e') {
      //enemyship movement
      //Debuging hitboxes

      enemyship_images[i].resize(128, 32);
      image(enemyship_images[i], spawned_xpos[i], spawned_ypos[i]);
    } else if (spawned[i] == 'a') {

      //Debuging hitboxes
      //rect(hitbox_spawned[0], hitbox_spawned[2], astroid_hitbox_size[0], astroid_hitbox_size[1]);

      astroid_images[i].resize(100, 50);
      image(astroid_images[i], spawned_xpos[i], spawned_ypos[i]);
    }
```

```
//Draw timely stuff
if (millis() <= timing_now_popup + timing_popup) {
  fill(154, 205, 50);
  noStroke();
  rect(width/3, height-80, width/3, 60, 15);
  fill(255);
  textSize(24);
  textAlign(CENTER);
  text(popup_message, width/2, height-40);
}

if (millis() <= timing_now_popup_above + timing_popup_above) {
  fill(220, 20, 60);
  popup_message_2 = "Fueled in time!";

  fill(154, 205, 50);
  noStroke();
  rect(width/3, 10, width/3, 60, 30);
  fill(0);
  textSize(24);
  textAlign(CENTER);
  text(popup_message_2, width/2, 45);
}

//Draw bars
fill(0);
textSize(20);
textAlign(CENTER);
text("Health", width-155, 40);
health_bar.draw();
```

```
    fill(0);
    textSize(20);
    text("Fuel", width-75, 40);
    power_bar.draw();
  }

  //Spawn bullets
  for (int j = 0; j < present_bullets.length; j++) {
    if (present_bullets[j] == 'b') {
      bullet_images[j].resize(4, 2);
      image(bullet_images[j], xpos_enemybullets[j], ypos_enemybullets[j]);
    }
  }

  //For friendly bullets
  for (int j = 0; j < friendly_bullets.length; j++) {
    if (friendly_bullets[j] == 'b') {
      friendly_bullets_img[j].resize(4, 2);
      image(friendly_bullets_img[j], xpos_friendlybullets[j], ypos_friendlybullets[j]);
    }
  }

  if (end_animation == true) {
    end_earth_x -= 4;
    endearth.resize(600, 600);
    image(endearth, end_earth_x, end_earth_y);

    if (end_earth_x < 0) {
      stopgame();
    }
  }
}
```

```
void spawn_astroid_ship_timer() {
  if (millis() >= time_now_enemyship + timing_enemyship) {
    time_now_enemyship = millis();
    // Spawn a enemy ship
    spawnElement(0);
    println("Enemy ship spawned");
  }
  if (millis() >= time_now_astroid + timing_astroid) {
    time_now_astroid = millis();
    // Spawn an astroid
    spawnElement(1);
    println("Astroid spawned");
  }
}

void spawnElement(int type) {
  //Type 0 for enemyship, 1 for astroid
  boolean spawned_element = false;

  for (int i = 0; i <spawned.length; i++) {
    if (spawned[i] == 'n' && spawned_element == false) {

      if (type == 0) {
        //Serial.println("Spawn enemy");
        //Spawn enemyship at location i in array
        spawned[i] = 'e';
        spawned_speed[i] = int(random(speed_ranges[0], speed_ranges[1]));
        spawned_xpos[i] = 1024; //All the way to the right of the displays
        spawned_ypos[i] = int(random(spawned_y_ranges[0], spawned_y_ranges[1]));
        spawned_element = true;
      } else if (type == 1) {
        //Serial.println("Spawn astroid");
        //Spawn astroid at location i in array
        spawned[i] = 'a';
        spawned_speed[i] = int(random(speed_ranges[0], speed_ranges[1]));
        spawned_xpos[i] = 1024; //All the way to the right of the displays
        spawned_ypos[i] = int(random(spawned_y_ranges[0], spawned_y_ranges[1]));
        spawned_element = true;
      }
    }
  }
}

void timedShipInteraction() {

  if (millis() >= time_now_reload + reload_timing) {
    time_now_reload = millis();

    if (first_reload) {
      reload_intime = true;
      first_reload = false;
    }

    if (reload_intime == false) {
      //Not reloaded in time
      visualize_parameter("Oof, no fuel was provided", 't'); //Does not work right now, the other
element draws over this.
      power -= int(random(6, 20));
      power_bar.update_bar(power);

      if (enable_port_extention) {
        set_game_bars();
      }
      if (power <= 0) {
        stopgame();
      }
    }

    reload_timing = int(random(reload_timing_boundries[0], reload_timing_boundries[1]));
    reload_parameter = pick_parameter(2); // 0 to indicate that this is a reload request.
    //Perform the reload visualization > Pick a random parameter from the ones that are availible and
then go wild

    send_Serial(reload_parameter, 'r');

    visualize_parameter("Fuel the ship", 'b');

    //print("Fuel parameter: ");
    //println(reload_parameter);

    reload_intime = false;
  }

  if (millis() >= time_now_fix + fix_timing) {
    time_now_fix = millis();

    if (first_fix) {
      fix_intime = true;
      first_fix = false;
    }

    if (fix_intime == false) {
      //Not fixed in time
      visualize_parameter("Ai, ship not fixed in time", 't'); //Does not work right now, the other
element draws over this.
      health -= int(random(6, 20));
      health_bar.update_bar(health);


      if (enable_port_extention) {
        set_game_bars();
      }


      if (health <= 0) {
        stopgame();
      }
    }


    fix_timing = int(random(fix_timing_boundries[0], fix_timing_boundries[1]));
    fix_parameter = pick_parameter(1);
    //Perform the fix visualization > Pick a random parameter from the ones that are availible and
then go wild


    send_Serial(fix_parameter, 'f');


    visualize_parameter("Fix the ship", 'b');


    //print("Fix parameter: ");
    //println(fix_parameter);


    fix_intime = false;
  }
}


void send_Serial(int number, char type) {


  if (enable_port_modsy == true) {


    modsyPort.write('<');
    if (type == 'f') {
      modsyPort.write('f'); //Sending fix par
    } else if (type == 'r') {
      modsyPort.write('r'); //Sending reload par
    }

    if (0 < number && number < 9) {
      modsyPort.write('1');
      modsyPort.write(str(number));
    } else if (8 < number && number < 17) {
      modsyPort.write('2');
      number -= 8;
      modsyPort.write(str(number));
    } else if (16 < number && number < 25) {
      modsyPort.write('3');
      number -= 16;
      modsyPort.write(str(number));
    } else if (24 < number && number < 33) {
      modsyPort.write('1');
      number -= 24;
      modsyPort.write(str(number));
    }

    modsyPort.write('>');
  }
}

void visualize_parameter(String message, char type) {

  if (type == 'b') {
    //b for bottom
    timing_now_popup = millis();
    popup_message = message;
  } else if (type == 't') {
    //t for top
    timing_now_popup_above = millis();
    popup_message_2 = message;
  }
}

int pick_parameter(int type) {
  int random_num;
  int parameter = 0;

  if (type == 1) {

    //This is the reload parameter case
    if (echoctrl) {
      random_num = int(random(0, echo_non_excluded.length));
      parameter = echo_non_excluded[random_num];
    } else {
      random_num = int(random(0, juno_non_excluded.length));
      parameter = juno_non_excluded[random_num];
    }

    if (parameter == fix_parameter) { //Or equal to one of the problematic parameters
      pick_parameter(type); //If its the same fun the function again.
    }
  } else if (type == 2) {

    //This is the fix parameter case
    if (echoctrl) {
      random_num = int(random(0, echo_non_excluded.length));
      parameter = echo_non_excluded[random_num];
    } else {
      random_num = int(random(0, juno_non_excluded.length));
      parameter = juno_non_excluded[random_num];
    }

    if (parameter == reload_parameter) { //Or equal to one of the problematic parameters
      pick_parameter(type); //If its the same fun the function again.
    }
  }

  return parameter;
}


void move_spawned() {

  calcHitbox('s', spaceship_x, spaceship_y); //Calc hitbox of friendly spaceship

  //Loop through all bullets and move them, also check the collision with the main device
  for (int j = 0; j < present_bullets.length; j++) {
    if (present_bullets[j] == 'b') {
      xpos_enemybullets[j] -= bullet_speed[j];

      calcHitbox('b', xpos_enemybullets[j], ypos_enemybullets[j]); //Calc hitbox of spawned element

      //Hit calculation
      if (hitbox_bullet[0] < hitbox_spaceship[1] && hitbox_bullet[1] > hitbox_spaceship[0] &&
hitbox_spaceship[2] < hitbox_bullet[3] && hitbox_spaceship[3] > hitbox_bullet[3] ) {
        //We have a hit, box calculation
        println("Bullet Hit");
        present_bullets[j] = 'n'; //Reset position to nothing
        xpos_enemybullets[j] = 0;
        if (set_shield == false) {
          health -= int(random(10, 30));
          health_bar.update_bar(health);
        }
      }
      if (hitbox_bullet[1] < 0) {
        println("Spawned element of map");
        present_bullets[j] = 'n'; //Reset position to nothing
        xpos_enemybullets[j] = 1023;
      }
    }
  }

  //Friendly bullets
  for (int a = 0; a < friendly_bullets.length; a++) {
    if (friendly_bullets[a] == 'b') {
      println("Move friendly bullet");

      xpos_friendlybullets[a] += friendly_bullet_speed;

      calcHitbox('b', xpos_friendlybullets[a], ypos_friendlybullets[a]); //Calc hitbox of spawned
element

      for (int i = 0; i < spawned.length; i++) {
        //Check if a friendly bullet has hit one of the spawned elements
        if (spawned[i] != 'n' ) {
          calcHitbox(spawned[i], spawned_xpos[i], spawned_ypos[i]); //Calc hitbox of spawned element

          if (hitbox_bullet[0] < hitbox_spawned[1] && hitbox_bullet[1] > hitbox_spawned[0] &&
hitbox_spawned[2] < hitbox_bullet[3] && hitbox_spawned[3] > hitbox_bullet[3] ) {
            println("Bullet hit");
            spawned[i] = 'n'; //Reset position to nothing
            spawned_xpos[i] = 1023;
          }
        }
      }



      if (hitbox_bullet[1] > 1023) {
        println("Own bullet of map");
        friendly_bullets[a] = 'n'; //Reset position to nothing
        xpos_friendlybullets[a] = 0;
      }
    }
  }

  //Loop through all spawned items and move them
  for (int i = 0; i < spawned.length; i++) {
    if (spawned[i] == 'e') {
      //enemyship movement
      spawned_xpos[i] -= spawned_speed[i];

      if (int(random(0, 5)) == 1) { //Change of 1 to 5
        // println("Shot called");
        possible_enemy_shot(spawned_xpos[i], spawned_ypos[i], spawned_speed[i]);
      }
    } else if (spawned[i] == 'a') {
      //Asteroid movement
      spawned_xpos[i] -= spawned_speed[i];
    }

    if (spawned[i] != 'n' ) {

      //Calculate hitbox
      calcHitbox(spawned[i], spawned_xpos[i], spawned_ypos[i]); //Calc hitbox of spawned element
```

```
      //rect(hitbox_spawned[0], hitbox_spawned[2], enemyship_hitbox_size[0],
enemyship_hitbox_size[1]);


      if (hitbox_spawned[0] < hitbox_spaceship[1] && hitbox_spawned[1] > hitbox_spaceship[0] &&
hitbox_spaceship[2] < hitbox_spawned[3] && hitbox_spaceship[3] > hitbox_spawned[3] ) {
        //We have a hit, box calculation
        println("There is a hit");
        spawned[i] = 'n'; //Reset position to nothing
        spawned_xpos[i] = 1023;


        if (set_shield == false) {
          health -= int(random(10, 30));
          health_bar.update_bar(health);
        }
      }


      if (hitbox_spawned[1] < 0) {
        println("Spawned element of map");
        spawned[i] = 'n'; //Reset position to nothing
        spawned_xpos[i] = 1023;
      }
    }
  }
}


void stopgame() {
  print("Stop the game");
  fill(100);
  noStroke();
  rect(width/3, height/4, width/3, height/2, 30);


  fill(255);
  textSize(28);
  textAlign(CENTER);
  text("Endgame", width/2, height/2);

  noLoop();
}

void calcHitbox(char type, int x_axis, int y_axis) {
  //This method is equiped to calculate the hitboxes of friendly ship 's', enemy ship 'e',..
  //, the astroid 'a'

  int left_boundry;
  int right_boundry;
  int top_boundry;
  int bottom_boundry;

  if (type == 's') {
    //Calc hitbox for own spaceship
    left_boundry = x_axis;
    right_boundry = x_axis + spaceship_hitbox_size[0];
    top_boundry = y_axis; //This might have to be flipped with the one bellow
    bottom_boundry = y_axis + spaceship_hitbox_size[1]; //This might have to be flipped with the one
above

    for (int i = 0; i < hitbox_spaceship.length; i++) {
      switch (i) {
      case 0:
        hitbox_spaceship[i] = left_boundry;
        break;
      case 1:
        hitbox_spaceship[i] = right_boundry;
        break;
      case 2:
        hitbox_spaceship[i] = top_boundry;
        break;
      case 3:
        hitbox_spaceship[i] = bottom_boundry;
        break;
      }
    }
  }

  if (type == 'e') {
    //Calc hitbox for enemy spaceship
    left_boundry = x_axis;
    right_boundry = x_axis + enemyship_hitbox_size[0];
    top_boundry = y_axis; //This might have to be flipped with the one bellow
    bottom_boundry = y_axis + enemyship_hitbox_size[1]; //This might have to be flipped with the
one above

    for (int i = 0; i < hitbox_spawned.length; i++) {
      switch (i) {
      case 0:
        hitbox_spawned[i] = left_boundry;
        break;
      case 1:
        hitbox_spawned[i] = right_boundry;
        break;
      case 2:
        hitbox_spawned[i] = top_boundry;
        break;
      case 3:
        hitbox_spawned[i] = bottom_boundry;
        break;
      }
    }
  }


  if (type == 'a') {
    //Calc hitbox for astroid
    left_boundry = x_axis;
    right_boundry = x_axis + astroid_hitbox_size[0];
    top_boundry = y_axis; //This might have to be flipped with the one bellow
    bottom_boundry = y_axis + astroid_hitbox_size[1]; //This might have to be flipped with the one
above


    for (int i = 0; i < hitbox_spawned.length; i++) {
      switch (i) {
      case 0:
        hitbox_spawned[i] = left_boundry;
        break;
      case 1:
        hitbox_spawned[i] = right_boundry;
        break;
      case 2:
        hitbox_spawned[i] = top_boundry;
        break;
      case 3:
        hitbox_spawned[i] = bottom_boundry;
        break;
      }
    }
  }
  if (type == 'b') {
    //Calc hitbox for bullet
    left_boundry = x_axis;
    right_boundry = x_axis + bullet_hitbox_size[0];
    top_boundry = y_axis; //This might have to be flipped with the one bellow
    bottom_boundry = y_axis + bullet_hitbox_size[1]; //This might have to be flipped with the one
above


    for (int i = 0; i < hitbox_bullet.length; i++) {
      switch (i) {
      case 0:
        hitbox_bullet[i] = left_boundry;
        break;
      case 1:
        hitbox_bullet[i] = right_boundry;
        break;
      case 2:
        hitbox_bullet[i] = top_boundry;
        break;
      case 3:
        hitbox_bullet[i] = bottom_boundry;
        break;
      }
    }
  }
}


void possible_enemy_shot(int x_pos, int y_pos, int current_speed) {
  //Change that there will be a shot is normally 1 to 5, this can be seen in the move section


  boolean bullet_spawned = false;


  for (int i = 0; i < present_bullets.length; i++) {
    if (present_bullets[i] == 'n' && bullet_spawned == false) {
      bullet_spawned = true;
      present_bullets[i] = 'b';
      xpos_enemybullets[i] = x_pos  + 40;
      ypos_enemybullets[i] = y_pos + 20;
      bullet_speed[i] = current_speed + 50;  //Bullet should go faster then enemy ship
    }
  }
}

void spaceship_shoots() {

  boolean bullet_spawned = false;

  for (int i = 0; i < friendly_bullets.length; i++) {
    if (friendly_bullets[i] == 'n' && bullet_spawned == false) {
      bullet_spawned = true;
      friendly_bullets[i] = 'b';
      xpos_friendlybullets[i] = spaceship_x  + 128;
      ypos_friendlybullets[i] = spaceship_y + 20;
      //Speed is already determined
    }
  }
}

void spaceship_shield() {
  set_shield = true;
  timing_now_shield = millis();
  power -= 10;
  power_bar.update_bar(power);
}

void assignSpaceShipFunctions() {
  //This function can be used to randomly assign the fire and shield buttons to a button on the
Modsy controller.

  fireButton = int(random(1, 5));
  shieldButton = int(random(1, 5));
```

```
if (fireButton == shieldButton) { //Recall the funciton if the parameters are the same.
  assignSpaceShipFunctions();
}

//print("Fire : ");
//println(fireButton);
//print("Shield : ");
//println(shieldButton);

if (enable_port_modsy) {
  modsyPort.write('<'); //to start the message
  modsyPort.write('o'); //random letter chosen for firebutton > o

  switch(fireButton) {
  case 1:
    modsyPort.write('1');
    break;
  case 2:
    modsyPort.write('2');
    break;
  case 3:
    modsyPort.write('3');
    break;
  case 4:
    modsyPort.write('4');
    break;
  }

  modsyPort.write('>'); //close the message

  modsyPort.write('<'); //to start the message
```

```
    modsyPort.write('d'); //random letter chosen for firebutton > d
    switch(shieldButton) {
    case 1:
      modsyPort.write('1');
      break;
    case 2:
      modsyPort.write('2');
      break;
    case 3:
      modsyPort.write('3');
      break;
    case 4:
      modsyPort.write('4');
      break;
    }
    modsyPort.write('>'); //close the message
  }
  //Here the first timing of the spaceship functions is also set
  reload_timing = int(random(reload_timing_boundries[0], reload_timing_boundries[1]));
  fix_timing = int(random(fix_timing_boundries[0], fix_timing_boundries[1]));
}

void keyPressed() {
  if (key == 's' || key == 'S') {
    run_game = true;
  }
}

//Healthbar class - Playful game interaction for the Modsy controller
//
//By Olivier Mathijssen
//05-07-2021
```

```
//
//Developed as part of the Bachelor Thesis the Playfulness of Modsy

class HealthBar {
  int value, max, x, y, w, h;
  color backing, bar;

  HealthBar(int xpos, color colour) {
    value = 2;
    max = 100;
    w = 50;
    h = 200;
    x = xpos;
    y = height/2-h/2;

    bar = color(220, 220, 220);
    backing = colour;
  }

  void draw() {
    fill(backing);
    noStroke();

    rect(x, y, w, h);

    fill(bar);
    rect(x, y, w, map(value, 0, max, 0, h));
  }

  void update_bar(int val) {
    value = 100 - val;
  }
}
```

J – Arduino program

J1 – Normal Modsy controller

```
//Demux adresses
//Multiplexer 1, selector for multiplexers
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

#include <USB-MIDI.h>
#include <Wire.h>
#include <er_oled.h>

#define LED_PIN    7
#define LED_COUNT 41

#define BUTTON_SENSE 5

Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);

USBMIDI_CREATE_DEFAULT_INSTANCE();

uint8_t oled_buf[WIDTH * HEIGHT / 8];

#define mux1_pinA 6
#define mux1_pinB 12
#define mux1_pinC 4

#define mux2_pinA A0
```

```
#define mux2_pinB A1
#define mux2_pinC A2

#define pot_sense_pin = A5;

String mapped_device = "Modsy";
String selected_device = " ";

boolean selected_mapped_change[2] = {false, false}; //0 Pos is mapped, 1 is for selected

//Of JUNO-106
const char string_1[] PROGMEM = "DCO RANGE";
const char string_2[] PROGMEM = "DCO LFO MOD";
const char string_3[] PROGMEM = "DCO PWM DEPTH";
const char string_4[] PROGMEM = "VCF CUTOFF FREQ";
const char string_5[] PROGMEM = "VCF RESONANCE";
const char string_6[] PROGMEM = "VCF KEY FOLLOW";
const char string_7[] PROGMEM = "EFFECT DEPTH";
const char string_8[] PROGMEM = " ";
const char string_9[] PROGMEM = "DCO PWM SOURCE";
const char string_10[] PROGMEM = "DCO PWM LEVEL";
const char string_11[] PROGMEM = "DCO SAW LEVEL";
const char string_12[] PROGMEM = "HPF CUTOFF FREQ";
const char string_13[] PROGMEM = "VCF ENV MOD";
const char string_14[] PROGMEM = "VCF LFO MOD";
const char string_15[] PROGMEM = " ";
const char string_16[] PROGMEM = " ";
const char string_17[] PROGMEM = "DCO SUB LEVEL";
const char string_18[] PROGMEM = "ARPEGGIO TYPE";
const char string_19[] PROGMEM = "ARPEGGIO STEP";
const char string_20[] PROGMEM = "LFO RATE";
const char string_21[] PROGMEM = "ENV1 ATTACK";
const char string_22[] PROGMEM = "ENV1 DECAY";
const char string_23[] PROGMEM = "ENV1 SUSTAIN";
const char string_24[] PROGMEM = "ENV1 RELEASE";
const char string_25[] PROGMEM = "ARPEGGIO SW";
const char string_26[] PROGMEM = " ";
const char string_27[] PROGMEM = " ";
const char string_28[] PROGMEM = " ";
const char string_29[] PROGMEM = "ENV2 ATTACK";
const char string_30[] PROGMEM = "ENV2 DECAY";
const char string_31[] PROGMEM = "ENV2 SUSTAIN";
const char string_32[] PROGMEM = "ENV2 RELEASE";

//Of JUNO-106 - Bram
//const char string_1[] PROGMEM = "DCO RANGE";
//const char string_2[] PROGMEM = "DELAY TYPE";
//const char string_3[] PROGMEM = "DCO PWM DEPTH";
//const char string_4[] PROGMEM = "VCF CUTOFF FREQ";
//const char string_5[] PROGMEM = "VCF RESONANCE";
//const char string_6[] PROGMEM = "VCF KEY FOLLOW";
//const char string_7[] PROGMEM = "EFFECT DEPTH";
//const char string_8[] PROGMEM = " ";
//const char string_9[] PROGMEM = "DCO PWM SOURCE";
//const char string_10[] PROGMEM = "DCO PWM LEVEL";
//const char string_11[] PROGMEM = "DCO SAW LEVEL";
//const char string_12[] PROGMEM = "HPF CUTOFF FREQ";
//const char string_13[] PROGMEM = "VCF ENV MOD";
//const char string_14[] PROGMEM = "VCF LFO MOD";
//const char string_15[] PROGMEM = " ";
//const char string_16[] PROGMEM = " ";
//const char string_17[] PROGMEM = "DCO SUB LEVEL";
//const char string_18[] PROGMEM = "ARPEGGIO TYPE";
//const char string_19[] PROGMEM = "ARPEGGIO STEP";
```

```
//const char string_20[] PROGMEM = "LFO RATE";
//const char string_21[] PROGMEM = "REVERB TYPE";
//const char string_22[] PROGMEM = "DCO NOISE LEVEL";
//const char string_23[] PROGMEM = "DELAY LEVEL";
//const char string_24[] PROGMEM = "REVERB LEVEL";
//const char string_25[] PROGMEM = "ARPEGGIO SW";
//const char string_26[] PROGMEM = " ";
//const char string_27[] PROGMEM = " ";
//const char string_28[] PROGMEM = " ";
//const char string_29[] PROGMEM = "ENV2 ATTACK";
//const char string_30[] PROGMEM = "ENV2 DECAY";
//const char string_31[] PROGMEM = "ENV2 SUSTAIN";
//const char string_32[] PROGMEM = "ENV2 RELEASE";

//Of ECHO
const char estring_1[] PROGMEM = "L Division";
const char estring_2[] PROGMEM = "R Division";
const char estring_3[] PROGMEM = " ";
const char estring_4[] PROGMEM = "Input Gain";
const char estring_5[] PROGMEM = "Feedback";
const char estring_6[] PROGMEM = " ";
const char estring_7[] PROGMEM = " ";
const char estring_8[] PROGMEM = " ";
const char estring_9[] PROGMEM = "Input Gain";
const char estring_10[] PROGMEM = "Feedback";
const char estring_11[] PROGMEM = " ";
const char estring_12[] PROGMEM = "Stereo Width";
const char estring_13[] PROGMEM = " ";
const char estring_14[] PROGMEM = " ";
const char estring_15[] PROGMEM = " ";
const char estring_16[] PROGMEM = " ";
const char estring_17[] PROGMEM = " ";
const char estring_18[] PROGMEM = " ";

const char estring_19[] PROGMEM = " ";

const char estring_20[] PROGMEM = "Output Gain";

const char estring_21[] PROGMEM = " ";

const char estring_22[] PROGMEM = " ";

const char estring_23[] PROGMEM = " ";

const char estring_24[] PROGMEM = " ";

const char estring_25[] PROGMEM = " ";

const char estring_26[] PROGMEM = " ";

const char estring_27[] PROGMEM = " ";

const char estring_28[] PROGMEM = "Channel Mode";

const char estring_29[] PROGMEM = " ";

const char estring_30[] PROGMEM = " ";

const char estring_31[] PROGMEM = " ";

const char estring_32[] PROGMEM = " ";



const char *const string_table[] PROGMEM = {string_1, string_2, string_3, string_4, string_5,
string_6, string_7, string_8, string_9, string_10,

                                        string_11, string_12, string_13, string_14, string_15, string_16, string_17,
string_18, string_19, string_20,

                                        string_21, string_22, string_23, string_24, string_25, string_26, string_27,
string_28, string_29, string_30,

                                        string_31, string_32

                                       };


const char *const estring_table[] PROGMEM = {estring_1, estring_2, estring_3, estring_4, estring_5,
estring_6, estring_7, estring_8, estring_9, estring_10,

                                         estring_11, estring_12, estring_13, estring_14, estring_15, estring_16,
estring_17, estring_18, estring_19, estring_20,

                                         estring_21, estring_22, estring_23, estring_24, estring_25, estring_26,
estring_27, estring_28, estring_29, estring_30,

                                         estring_31, estring_32

                                        };

char buffer[30];
```

```
boolean string_change[33];

boolean juno_mapping = true;

int pot_values[32] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
                    };//Holds all the values of the potentiometers

int button_values[6] = {0, 0, 0, 0, 0, 0};

//For playful interaction
long par_timer;

//boolean newData = false;
const byte numChars = 10;
char receivedChars[numChars];
String incoming_message;

int reload_parameter = 0;
int fix_parameter = 0;
int fix_led = 0;
int reload_led = 0;

int fire_button = 0;
int shield_button = 0;

//For on/off
boolean low = true;

void OnMidiSysEx(byte* data, unsigned length) {
  char incoming_chardata[length];
  String total_message;
```

```
boolean display_message = true;
boolean intro_message = false;
char char_par_num[2] = {'0', '0'};
String parameter_number_str;

for (uint16_t i = 0; i < length; i++) {

  if (data[i] == 240) {
    //Start message
    incoming_chardata[i] = ' ';
  } else if (data[i] == 247) {
    incoming_chardata[i] = '\0';
  } else {
    incoming_chardata[i] = char(data[i]);
  }

}

//Check for the type of message
if (incoming_chardata[1] == 'l') {
  display_message = false;
}  else if (incoming_chardata[1] == 'i') {
  intro_message = true;
} else {
  display_message = true;
} //Could add extra for value updating

//For the display or LED number
char_par_num[0] = char(incoming_chardata[2]);
char_par_num[1] = char(incoming_chardata[3]);

if (char_par_num[0] == '0') {
  parameter_number_str = String(char_par_num[1]);
} else {
  parameter_number_str = String(char_par_num);
}

int parameter_number = parameter_number_str.toInt();
total_message = String(incoming_chardata).substring(4);

if (intro_message == true) {
  switch (parameter_number) {
    case 55:
      led_vis(1);
      break;
    case 56:
      led_vis(2);
      break;
    case 57:
      led_vis(3);
      break;
    case 58:
      led_vis(4);
      break;
    case 59:
      led_vis(5);
      break;

    default:
      break;
  }
}

if (parameter_number > 0) {
    if (parameter_number == 40) {
      //mapped data
      selected_mapped_change[0] = true;
      mapped_device = total_message;

      if (total_message == "JUNO-106(VST2 64bit)") {
        juno_mapping = true;
        for (int i = 0; i < 33; i++) {
          string_change[i] = true;
        }
      } else if (total_message == "Echo") {
        juno_mapping = false;
        for (int i = 0; i < 33; i++) {
          string_change[i] = true;
        }
      }

    } else if (parameter_number == 41) {
      //selected data
      selected_mapped_change[1] = true;
      selected_device = total_message;
    }

  }

}

void setup() {

  Serial.begin(9600);

  Wire.begin();
  //Lister to MIDI on channel 1
  MIDI.begin(1);
  MIDI.setHandleSystemExclusive(OnMidiSysEx);

  par_timer = millis();

  strip.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
  strip.show();            // Turn OFF all pixels ASAP
  strip.setBrightness(50); // Set BRIGHTNESS to about 1/5 (max = 255)

  //Mux 1
  pinMode(mux1_pinA, OUTPUT);
  pinMode(mux1_pinB, OUTPUT);
  pinMode(mux1_pinC, OUTPUT);

  //Mux 2
  pinMode(mux2_pinA, OUTPUT);
  pinMode(mux2_pinB, OUTPUT);
  pinMode(mux2_pinC, OUTPUT);

  //Buttonsense
  pinMode(BUTTON_SENSE, INPUT);

  initiate_displays();
  initiate_leds();

}

void initiate_displays() {

  for (int i = 0; i < 33; i++) {
```

```
  string_change[i] = true;
}

par_timer = millis();

int mux = 0;
int row_num = 0;
int corrected_row_num = 0;

for (byte byteCounter = 0; byteCounter < 5; byteCounter++) {
  mux++;
  row_num = 0;

  for (byte byteCounterTwo = 0; byteCounterTwo < 8; byteCounterTwo++) {
    row_num++;

    select_adress(byteCounter, byteCounterTwo);

    if (mux == 1) {
      if (row_num == 1 || row_num == 2) {
        er_oled_begin();
        er_oled_clear(oled_buf);

        command(0xa6);//--set normal display
        er_oled_string(10, 2, "--", 15, 1, oled_buf);

        er_oled_display(oled_buf);
      }
    }

    if (mux > 1 && mux < 6) {
      /* display an image of bitmap matrix */
```

```
        er_oled_begin();
        er_oled_clear(oled_buf);

        command(0xa6);//--set normal display

        er_oled_string(10, 2, "-", 15, 1, oled_buf);

        er_oled_display(oled_buf);
      }

    }
  }

  //Serial.println(F("Done with screen initialization"));

}

void initiate_leds() {
  uint32_t color = strip.Color(0, 127, 0);
  int wait = 100;

  for (int a = 0; a < 10; a++) { // Repeat 10 times...
    for (int b = 0; b < 3; b++) { //  'b' counts from 0 to 2...
      strip.clear();         //   Set all pixels in RAM to 0 (off)
      // 'c' counts up from 'b' to end of strip in steps of 3...
      for (int c = b; c < strip.numPixels(); c += 3) {
        strip.setPixelColor(c, color); // Set pixel 'c' to value 'color'
      }
      strip.show(); // Update strip with new contents
      delay(wait);  // Pause for a moment
    }
  }
  for (int i = 0; i < strip.numPixels(); i++) {
    strip.setPixelColor(i, strip.Color(0, 0, 0));
    strip.show();
  }

}

void led_vis(int type) {
  //Led visualisaties voor de Robster

  switch (type) {
    case 1:
      for (int i = 0; i < strip.numPixels(); i++) { // For each pixel in strip...
        strip.setPixelColor(i, strip.Color(61, 79, 161));         //  Set pixel's color (in RAM)
        strip.show();                          //  Update strip to match
        delay(50);                           //  Pause for a moment
      }
      for (int i = 0; i < strip.numPixels(); i++) { // For each pixel in strip...
        strip.setPixelColor(i, strip.Color(0, 0, 0));         //  Set pixel's color (in RAM)
        strip.show();                          //  Update strip to match
        delay(50);                           //  Pause for a moment
      }

    case 2:
      //Top right parameter ON
      strip.setPixelColor(1, strip.Color(61, 79, 161));
      strip.show();

      break;

    case 3:
```

```
    //Top right parameter OFF
    strip.setPixelColor(1, strip.Color(0, 0, 0));
    strip.show();

    break;

   case 4:
    //All parameters ON
    for (int i = 1; i < strip.numPixels(); i++) {
     strip.setPixelColor(i, strip.Color(61, 79, 161));
     strip.show();
    }
    break;

   case 5:
    //All parameters OFF
    for (int i = 1; i < strip.numPixels(); i++) {
     strip.setPixelColor(i, strip.Color(0, 0, 0));
     strip.show();
    }
    break;

  }
}

void loop() {
  // put your main code here, to run repeatedly:
  MIDI.read(); //MIDI read
  MidiUSB.flush();

  recvWithStartEndMarkers();
  loop_muxes();
}


void loop_muxes() {
  int mux = 0;
  int row_num = 0;
  int corrected_row_num = 0;

  for (byte byteCounter = 0; byteCounter < 5; byteCounter++) {
    mux++;
    row_num = 0;

    for (byte byteCounterTwo = 0; byteCounterTwo < 8; byteCounterTwo++) {
      row_num++;

      select_adress(byteCounter, byteCounterTwo);

      corrected_row_num = convert_element_num(mux, row_num);

      if (mux == 1) {
        switch (row_num) {
          case 1:
            //For the currently selected display
            if (selected_mapped_change[1] == true) {
              selected_mapped_change[1] = false;
              er_oled_clear(oled_buf);
              er_oled_string(10, 2, selected_device.c_str(), 15, 1, oled_buf);
              er_oled_display(oled_buf);
            }
            break;
          case 2:
            //For the mapped display, during bytecounting this results in the first element
            if (selected_mapped_change[0] == true) {
              selected_mapped_change[0] = false;
              er_oled_clear(oled_buf);
              er_oled_string(10, 2, mapped_device.c_str(), 15, 1, oled_buf);
              er_oled_display(oled_buf);
            }
            break;
          case 3:
            //Mapping button
            readout_button(1);
            break;
          case 4:
            //Page button
            readout_button(2);
            break;
          case 5:
            //FButton 1
            readout_button(3);
            break;
          case 6:
            //FButton 2
            readout_button(4);
            break;
          case 7:
            //FButton 3
            readout_button(5);
            break;
          case 8:
            //FButton 4
            readout_button(6);
            break;
          default:
```

```
          break;

        }
      }

      if (mux > 1 && mux < 6) {

        readPotVal(corrected_row_num);

        if (millis() > par_timer + 800) {
          if (string_change[corrected_row_num - 1]) {
            //If the display value has changed...
            string_change[corrected_row_num - 1] = false;
            //Serial.print("Update Display");

            er_oled_clear(oled_buf);

            if (juno_mapping == true) {
              strcpy_P(buffer, (char *)pgm_read_word(&(string_table[corrected_row_num - 1]))); //
Necessary casts and dereferencing, just copy.
            } else {
              strcpy_P(buffer, (char *)pgm_read_word(&(estring_table[corrected_row_num - 1]))); //
Necessary casts and dereferencing, just copy.
            }
            er_oled_string(10, 2, buffer, 15, 1, oled_buf);
            er_oled_display(oled_buf);
          }
        }

      }
    }
  }
}
```

```
void readout_button(int id) {

 int button_readout = digitalRead(BUTTON_SENSE);

 if (button_readout != button_values[id - 1]) {
  Serial.println("Buttonpressed");
  if (button_readout == HIGH) {
   if (id == 1) {
    //Mapping button
    MIDI.sendNoteOn(42, 127, 1);
   } else if (id == 2) {
    //Mapping page button
    MIDI.sendNoteOn(36, 127, 1);
   } else {
    //This will send note with velocity 64, 65, 66 and 67
    MIDI.sendNoteOn(id + 61, 127, 1);
   }
  }
  button_values[id - 1] = button_readout;
 }
}

void readPotVal(int select_num) {
 int pot_readout;
 int buffer_pot = 3;
 int samples = 7;



 pot_readout = 0;
 for (int i = 0; i < samples; i++) {
  pot_readout += analogRead(A5);
  }
  pot_readout = pot_readout / samples;

  if (select_num > 24 && select_num < 29) {
    if (pot_readout < 500) {
      pot_readout = 0;
    } else if (pot_readout >= 500) {
      pot_readout = 1023;
    }
  }

  if (pot_readout > pot_values[select_num - 1] + buffer_pot || pot_readout < pot_values[select_num
- 1] - buffer_pot) {

    int mapperoutput = map(pot_readout, 0, 1023, 0, 127);

    if (select_num == reload_parameter) {
      Serial.println("pr");
      strip.setPixelColor(reload_led, strip.Color(0, 0, 0));
      strip.show();
    } else if (select_num == fix_parameter) {
      Serial.println("pf");
      strip.setPixelColor(fix_led, strip.Color(0, 0, 0));
      strip.show();
    } else if (select_num == fire_button) {
      Serial.println("g"); //g for gun
    } else if (select_num == shield_button) {
      Serial.println("s"); //s for shield
    }

    if (select_num > 0 && select_num < 9) {
      MIDI.sendControlChange(select_num + 15, mapperoutput, 1);
    } else if (select_num > 8 && select_num < 17) {
```

```
 MIDI.sendControlChange(select_num + 15, mapperoutput, 1);
} else if (select_num > 16 && select_num < 25) {
 MIDI.sendControlChange(select_num + 31, mapperoutput, 1);
} else if (select_num > 24 && select_num < 33) {
 if (select_num > 24 && select_num < 29) {
  //Special algorithm for buttons, send high or low
  if (mapperoutput == 127) {
   if (low == true) {
    Serial.println("Send 127");
    MIDI.sendControlChange(select_num + 31, 127, 1);
    low = false;
   } else {
    Serial.println("Send 0");
    MIDI.sendControlChange(select_num + 31, 0, 1);
    low = true;
   }
  }

 } else {
  MIDI.sendControlChange(select_num + 31, mapperoutput, 1);
 }
}

pot_values[select_num - 1] = pot_readout;

er_oled_clear(oled_buf);
er_oled_string(10, 2, String(mapperoutput).c_str(), 15, 1, oled_buf);
er_oled_display(oled_buf);

par_timer = millis();
string_change[select_num - 1] = true;
```

```
 }

}

int convert_element_num(int mux, int parameter_in_row) {
 //This whole method would not have been neccessary with good hardware design
 int display_num;

 if (parameter_in_row < 3) {
  display_num = 9 - parameter_in_row;
 } else {
  display_num = parameter_in_row - 2;
 }

 switch (mux) {
  case 1:
   display_num += 32;
   break;
  case 2:
   display_num += 24;
   break;
  case 3:
   display_num += 16;
   break;
  case 4:
   display_num += 8;
   break;
  case 5:
   //Nothing needs to happen
   break;
 }
```

```
  return display_num;
}

void select_adress(byte address_mux1, byte address_mux2) {

  int M1S0 = bitRead(address_mux1, 0);//Least significant bit
  int M1S1 = bitRead(address_mux1, 1);
  int M1S2 = bitRead(address_mux1, 2);//Most significant bit

  int M2S0 = bitRead(address_mux2, 0);
  int M2S1 = bitRead(address_mux2, 1);
  int M2S2 = bitRead(address_mux2, 2);

  digitalWrite(mux1_pinA, M1S0);
  digitalWrite(mux1_pinB, M1S1);
  digitalWrite(mux1_pinC, M1S2);

  digitalWrite(mux2_pinA, M2S0);
  digitalWrite(mux2_pinB, M2S1);
  digitalWrite(mux2_pinC, M2S2);
}

void recvWithStartEndMarkers() {

  static boolean recvInProgress = false;
  static byte ndx = 0;
  char startMarker = '<';
  char endMarker = '>';
  char rc;

  while (Serial.available() > 0) {
    rc = Serial.read();
```

```
      if (recvInProgress == true) {
        if (rc != endMarker) {
          receivedChars[ndx] = rc;
          ndx++;
          if (ndx >= numChars) {
            ndx = numChars - 1;
          }
        } else {
          receivedChars[ndx] = '\0'; // terminate the string
          recvInProgress = false;
          ndx = 0;

          incoming_message = String(receivedChars);

          check_message(receivedChars);
        }
      } else if (rc == startMarker) {
        recvInProgress = true;
      }
    }

}

void check_message(String incoming_message) {

  int par_number = 0;
  int led_num = 0;

  int button_num = 0;

  char row_char = incoming_message[1]; //For the row
  char element_char; //For the number in the row

  switch (row_char) {
      case '1':
        par_number = 0;

        led_num = 35;
        button_num = 25;
        break;
      case '2':
        par_number = 8;

        led_num = 33;
        button_num = 26;
        break;
      case '3':
        par_number = 16;

        led_num = 31;
        button_num = 27;
        break;
      case '4':
        par_number = 24;

        led_num = 29;
        button_num = 28;
        break;
      default:
        //Serial.println("No correct row");
        break;
  }
```

```
if (incoming_message[0] == 'o') {

  //O is send first, remove all LEDS from the field
  for (int i = 0; i < strip.numPixels(); i++) {
    strip.setPixelColor(i, strip.Color(0, 0, 0));
    strip.show();
  }

  //For assignment of the firebutton
  fire_button = button_num;

  Serial.print("-Fire fire: ");
  Serial.println(String(fire_button));
  //Set the new led to a colour
  strip.setPixelColor(led_num, strip.Color(230, 3, 5)); //Red colour for firing
  strip.show();

} else if (incoming_message[0] == 'd') {
  //For assignment of the shieldbutton
  shield_button = button_num;

  Serial.print("-Shield");
  Serial.println(String(shield_button));

  //Set the new led to a colour
  strip.setPixelColor(led_num, strip.Color(7, 6, 230)); //Blue colour for shield
  strip.show();

} else {

  element_char = incoming_message[2];
```

```
switch (element_char) {
 case '1':

   if (row_char == '2' || row_char == '4') {
     led_num = par_number + 8;
     par_number += 1;
   } else {
     par_number += 1;
     led_num = par_number;
   }

   break;
 case '2':

   if (row_char == '2' || row_char == '4') {
     led_num = par_number + 7;
     par_number += 2;
   } else {
     par_number += 2;
     led_num = par_number;
   }

   break;
 case '3':

   if (row_char == '2' || row_char == '4') {
     led_num = par_number + 6;
     par_number += 3;

   } else {
     par_number += 3;
     led_num = par_number;
```

```
  }

  break;
case '4':


  if (row_char == '2' || row_char == '4') {
    led_num = par_number + 5;
    par_number += 4;
  } else {
    par_number += 4;
    led_num = par_number;
  }
  break;
case '5':

  if (row_char == '2' || row_char == '4') {
    led_num = par_number + 4;
    par_number += 5;
  } else {
    par_number += 5;
    led_num = par_number;
  }

  break;
case '6':


  if (row_char == '2' || row_char == '4') {
    led_num = par_number + 3;
    par_number += 6;
  } else {
      par_number += 6;
      led_num = par_number;
    }

    break;
  case '7':

    if (row_char == '2' || row_char == '4') {
      led_num = par_number + 2;
      par_number += 7;
    } else {
      par_number += 7;
      led_num = par_number;
    }

    break;
  case '8':

    if (row_char == '2' || row_char == '4') {
      led_num = par_number + 1;
      par_number += 8;
    } else {
      par_number += 8;
      led_num = par_number;
    }

    break;
  default:
    //Serial.println("No correct element");
    break;
}
```

```
  if (incoming_message[0] == 'f') {
   //First set the past led to 0
   strip.setPixelColor(fix_led, strip.Color(0, 0, 0));
   //Then set the new led to a colour
   strip.setPixelColor(led_num, strip.Color(50, 205, 10));
   strip.show();
   //update the fix parameter
   fix_led = led_num;
   fix_parameter = par_number;

  } else if (incoming_message[0] == 'r') {
   //First set the past led to 0
   strip.setPixelColor(reload_led, strip.Color(0, 0, 0));
   //Then set the new led to a colour
   strip.setPixelColor(led_num, strip.Color(255, 255, 10));
   strip.show();
   reload_led = led_num;
   reload_parameter = par_number;
  }

 }
}
```

J2 – Playful extension Modsy controller

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

//Pin for LED data
#define LED_PIN    7
//Amount of LEDs
#define LED_COUNT 16

Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);

//Control for Joystick
int VRx = A0; //Pin of the X-axis joystick
int VRy = A1; //Pin of the Y-axis joystick
int SW = 2;   //Pin for switch of joystick?
int xPosition = 0;
int yPosition = 0;
int previous_xPosition = 0;
int previous_yPosition = 0;
int SW_state = 0;
int mapX = 0;
int mapY = 0;

int health_level;
int power_level;

const byte numChars = 20;
char receivedChars[numChars];   // an array to store the received data
char  dataNumber = 0;
boolean newData = false;
```

```
char char_par_num[2];
String incoming_message;

unsigned long time_now_joystick;
int period = 100; //period of 0.1Hz

char lastmove = 'n';

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);




  //Setting up joystick control
  pinMode(VRx, INPUT);
  pinMode(VRy, INPUT);
  pinMode(SW, INPUT_PULLUP);

  strip.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
  strip.show();            // Turn OFF all pixels ASAP
  strip.setBrightness(50); // Set BRIGHTNESS to about 1/5 (max = 255)

  set_leds();

}

void loop() {
  // put your main code here, to run repeatedly:

  readJoystick();
  recvWithStartEndMarkers();

}

void recvWithStartEndMarkers() {

  static boolean recvInProgress = false;
  static byte ndx = 0;
  char startMarker = '<';
  char endMarker = '>';
  char rc;

  while (Serial.available() > 0) {
    rc = Serial.read();

    //Serial.println("Ola");

    if (recvInProgress == true) {
      if (rc != endMarker) {
        receivedChars[ndx] = rc;
        ndx++;
        if (ndx >= numChars) {
          ndx = numChars - 1;
        }
      } else {
        receivedChars[ndx] = '\0'; // terminate the string
        recvInProgress = false;
        ndx = 0;

        incoming_message = String(receivedChars);
        check_message(incoming_message);
        //Serial.println(incoming_message);
      }
    } else if (rc == startMarker) {
      recvInProgress = true;
    }
  }

}

void set_leds() {
  for (int j = 8; j < strip.numPixels(); j++) {
    //For the health bar
    strip.setPixelColor(j, strip.Color(50, 205, 10));         //  Set pixel's color (in RAM)
    strip.show();
  }
  for (int i = 0; i < 8; i++) {
    //For the power bar
    strip.setPixelColor(i, strip.Color(255, 255, 10));        //  Set pixel's color (in RAM)
    strip.show();                          //  Update strip to match
  }
}


void check_message(String incoming_msg) {
  String parameter_number_str;
  char level = incoming_msg.charAt(1);

  Serial.println(level);



  if (incoming_msg.charAt(0) == 'h') {
    switch (level) {
```

```
case '1':
 for (int i = 8; i < 9; i++) { // For each pixel in strip...
  strip.setPixelColor(i, strip.Color(50, 205, 10));       //  Set pixel's color (in RAM)
  strip.show();                      //  Update strip to match
 }
 for (int i = 9; i < 16; i++) { // For each pixel in strip...
  strip.setPixelColor(i, strip.Color(0, 0, 0));       //  Set pixel's color (in RAM)
  strip.show();                      //  Update strip to match
 }

 break;

case '2':
 for (int i = 8; i < 10; i++) { // For each pixel in strip...
  strip.setPixelColor(i, strip.Color(50, 205, 10));       //  Set pixel's color (in RAM)
  strip.show();                      //  Update strip to match
 }
 for (int i = 10; i < 16; i++) { // For each pixel in strip...
  strip.setPixelColor(i, strip.Color(0, 0, 0));       //  Set pixel's color (in RAM)
  strip.show();                      //  Update strip to match
 }

 break;

case '3':
 for (int i = 8; i < 11; i++) { // For each pixel in strip...
  strip.setPixelColor(i, strip.Color(50, 205, 10));       //  Set pixel's color (in RAM)
  strip.show();                      //  Update strip to match
 }
 for (int i = 11; i < 16; i++) { // For each pixel in strip...
  strip.setPixelColor(i, strip.Color(0, 0, 0));       //  Set pixel's color (in RAM)
  strip.show();                      //  Update strip to match
  }

  break;

 case '4':
  for (int i = 8; i < 12; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(50, 205, 10));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
  }
  for (int i = 12; i < 16; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(0, 0, 0));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
  }

  break;

 case '5':
  for (int i = 8; i < 13; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(50, 205, 10));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
  }
  for (int i = 13; i < 16; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(0, 0, 0));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
  }

  break;

 case '6':
  for (int i = 8; i < 14; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(50, 205, 10));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
```

```
      }
      for (int i = 14; i < 16; i++) { // For each pixel in strip...
        strip.setPixelColor(i, strip.Color(0, 0, 0));         //  Set pixel's color (in RAM)
        strip.show();                          //  Update strip to match
      }

      break;

    case '7':
      for (int i = 8; i < 15; i++) { // For each pixel in strip...
        strip.setPixelColor(i, strip.Color(50, 205, 10));         //  Set pixel's color (in RAM)
        strip.show();                          //  Update strip to match
      }
      for (int i = 15; i < 16; i++) { // For each pixel in strip...
        strip.setPixelColor(i, strip.Color(0, 0, 0));         //  Set pixel's color (in RAM)
        strip.show();                          //  Update strip to match
      }

      break;

    case '8':
      for (int i = 8; i < 16; i++) { // For each pixel in strip...
        strip.setPixelColor(i, strip.Color(50, 205, 10));         //  Set pixel's color (in RAM)
        strip.show();                          //  Update strip to match
      }
      break;

    default:
      break;

  }
} else if (incoming_msg.charAt(0) == 'p') {
```

```
switch (level) {
 case '1':
  for (int i = 7; i < 8; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(255, 255, 10));        //  Set pixel's color (in RAM)
   strip.show();                         //  Update strip to match
  }
  for (int i = 0; i < 7; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(0, 0, 0));        //  Set pixel's color (in RAM)
   strip.show();                         //  Update strip to match
  }                         //  Update strip to match
  break;

 case '2':
  for (int i = 6; i < 8; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(255, 255, 10));        //  Set pixel's color (in RAM)
   strip.show();                         //  Update strip to match
  }
  for (int i = 0; i < 6; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(0, 0, 0));        //  Set pixel's color (in RAM)
   strip.show();                         //  Update strip to match
  }

  break;

 case '3':
  for (int i = 5; i < 8; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(255, 255, 10));        //  Set pixel's color (in RAM)
   strip.show();                         //  Update strip to match
  }
  for (int i = 0; i < 5; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(0, 0, 0));        //  Set pixel's color (in RAM)
   strip.show();                         //  Update strip to match
  }

  break;

 case '4':
  for (int i = 4; i < 8; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(255, 255, 10));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
  }
  for (int i = 0; i < 4; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(0, 0, 0));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
  }

  break;

 case '5':
  for (int i = 3; i < 8; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(255, 255, 10));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
  }
  for (int i = 0; i < 3; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(0, 0, 0));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
  }

  break;

 case '6':
  for (int i = 2; i < 8; i++) { // For each pixel in strip...
   strip.setPixelColor(i, strip.Color(255, 255, 10));        //  Set pixel's color (in RAM)
   strip.show();                        //  Update strip to match
```

```
    }
    for (int i = 0; i < 2; i++) { // For each pixel in strip...
      strip.setPixelColor(i, strip.Color(0, 0, 0));         //  Set pixel's color (in RAM)
      strip.show();                          //  Update strip to match
    }

    break;

  case '7':
    for (int i = 1; i < 8; i++) { // For each pixel in strip...
      strip.setPixelColor(i, strip.Color(255, 255, 10));         //  Set pixel's color (in RAM)
      strip.show();                          //  Update strip to match
    }
    for (int i = 0; i < 1; i++) { // For each pixel in strip...
      strip.setPixelColor(i, strip.Color(0, 0, 0));         //  Set pixel's color (in RAM)
      strip.show();                          //  Update strip to match
    }

    break;

  case '8':
    for (int i = 0; i < 8; i++) { // For each pixel in strip...
      strip.setPixelColor(i, strip.Color(255, 255, 10));         //  Set pixel's color (in RAM)
      strip.show();                          //  Update strip to match
    }
    break;

  default:
    break;

 }
}
```

```
}
void readJoystick() {

  if (millis() >= time_now_joystick + period) {
    time_now_joystick = millis();

    xPosition = analogRead(VRx);
    yPosition = analogRead(VRy);

    //  Serial.print("X pos: ");
    //  Serial.println(xPosition);
    //  Serial.print("Y pos: ");
    //  Serial.println(yPosition);

    SW_state = digitalRead(SW);

    if (xPosition > 900) {
      Serial.println("ju");
    }
    if (xPosition < 100) {
      Serial.println("jd");
    }
    if (yPosition > 900) {
      Serial.println("jr");
    }
    if (yPosition < 100) {
      Serial.println("jl");
    }
  }

}
```