AALBORG UNIVERSITET

Department of Architecture,
Design and Media Technology

Medialogy, 3. Semester

Title: Music creation through image processing

Project period: Week 36 to 47

Semester theme: Visual Computing-Human Perception

Supervisors: Ramin Irani

Projectgroup no. 15336

Members:

Mads Petersen Schmidt

Mathias Wessmann

Mikkel Damgaard Vindum

Nicolai Bruhn Lauritsen

Thor Vest Tidemand

Vytautas Kulnickis

Abstract:

This report contains the project made by group MTA15336 with the semester theme: "Visual Computing-Human Perception."

The group has developed a prototype using the OpenCV library of C++ and a musical interface developed in the Unity Engine. The purpose of this report is to discover whether it is possible to develop a digital prototype, which is controlled by utilizing hand gestures, to manipulate the cursor on-screen, simulate several key press, and create short musical sequences in the implemented music creator. The system was tested on several users to evaluate how well the hand recognition performed. Based on the information from the evaluation a more advanced version of the system was made to better recognize the hand movement, gestures and fingers.

Table of content

1. Introduction

Creating music on computers using current solutions can be a complex process for beginners and people without experience. The entry level can be too demanding in terms of knowledge of musical composition, and technical prowess.

In this project, an analysis of current existing solutions for creating music, both simple and complex, will be carried out. Based on this, a solution will be developed, with the aim of making creation of simple music possible for a user with no previous experience. This will be done by implementing hand recognition to control the system intuitively, combined with creating an easy to understand interface in the music creation software.

Once a solution has been designed, it should be implemented primarily in C++. A prototype should be developed through this implementation, with enough features that it can later be evaluated upon.

This report also introduces new areas of investigation that could either be explored on their own merit, or as an extension of this project.

2. Initial Problem Statement

2.1 Motivation

Today composing and playing music is integrated into our growing digital lifestyles through different types of software such as FL Studio [18] and Garage Band [1]. Two of the important aspects of digital music are design and technology. The technology for a way to create it, listen to it and share it, and design as an important aspect for structuring and navigating naturally. The current solutions are not necessarily easy to navigate and can be conveyed in a matter, that makes it hard to learn and even harder to master. The existing solutions makes it difficult for beginners to pick up composing and allowing them to make simple tracks, in order to learn how to start making music of their own.

This project aims to simplify the way the user can create music, by allowing them to utilize their hands to “grab” instruments they want and place them in an ordered environment, to create pieces of music. The user can control the program through hand gestures and movement. The goal is to teach the user how different instruments sound, how these can work together to make a simple piece of music. This should provide an easy to use interface for the user, through intuitive controls.

2.2 Research question

How can we use image processing to make use of hand gestures to drag, drop, and control other functions in an application?

2.3 Description

To create a solution that fulfils these goals, the group will design a canvas where you drag or point where different sounds should be played on a timeline. This should be controlled by using hand gestures to point out where the sounds should be played. The solution should be able to play different tones, and ideally have implemented several different instruments. This should be presented in an easy to understand user interface that does not limit the user based on knowledge of music theory or complex music creation software.

3. Background Research

3.1 Public interfaces

Public interfaces can be used to create awareness about a subject [5]. If people saw a public activity with an interactive interface they could become aware of something they otherwise would not. In an article about awareness in workspaces, Dourish and Bellotti describes awareness as “*an understanding of the activities of others, which provides a context for your own activities”* [5]. This context can be different types of information, which could be relevant for your work or leisure activities. An important social activity in a community is community meetings, which these days also share the information from the meetings on online social networks. One of the problems is that awareness about the meetings are not prioritized to public shared displays, therefore lessening the attention and knowledge about the meetings. A solution to this has been made by Michael Koch [10], in the form of public interfaces that provides an interactive visualization of the meetings, called ‘The Meeting Mirror’. The meeting mirror is a public interactive interface where community members can see check in and see information about other participants and participate in the meetings. They have implemented and used the interfaces for two events, and feedback about usefulness have been positive.

Public collaboration activities is something that can bind people together. Activities that interest people that are passing by makes them want to join in. It should be something that allows for anyone to interact and possibly collaborate with other people that they might have never met before. Sociological work states that no active interaction among members of a community is required, but rather the possibility to interact is needed [3]. This could be an activity which applies mutual collaboration between people, where people would want to exchange knowledge or get the will to help each other. A community should therefore not be explained as a group of people who have the possibility to communicate, but rather wanting to communicate and collaborate with each other.

Public activities are something that binds people together. If this was done with a public interface, where people were allowed to work together to create music for others to enjoy, they would indirectly exchange knowledge and possibly engage the people passing by. By taking something as the V Motion Project [7] and making it possible for people to join in and contribute to the activity, it could attract and entertain people who were just walking by. When leaving a personal touch on a project, made by many other people, it becomes a medium for indirect communication and exchange of content. Public activities like that could be used just for ‘fun’, or it could even be used as a way to pass time if waiting for a train.

3.2 State of the art

FL Studio is an audio production software for users to make their own beats and songs from a variety of tools and sounds. It is mainly used by experienced music creators and electronic artists. The overall interface of the program contains many smaller windows, each with its own function for editing as seen in figure 1. The software, which contains several thousands of different sound samples and audio tracks, also allows downloading and adding of other sound packages, further enlarging the audio library.

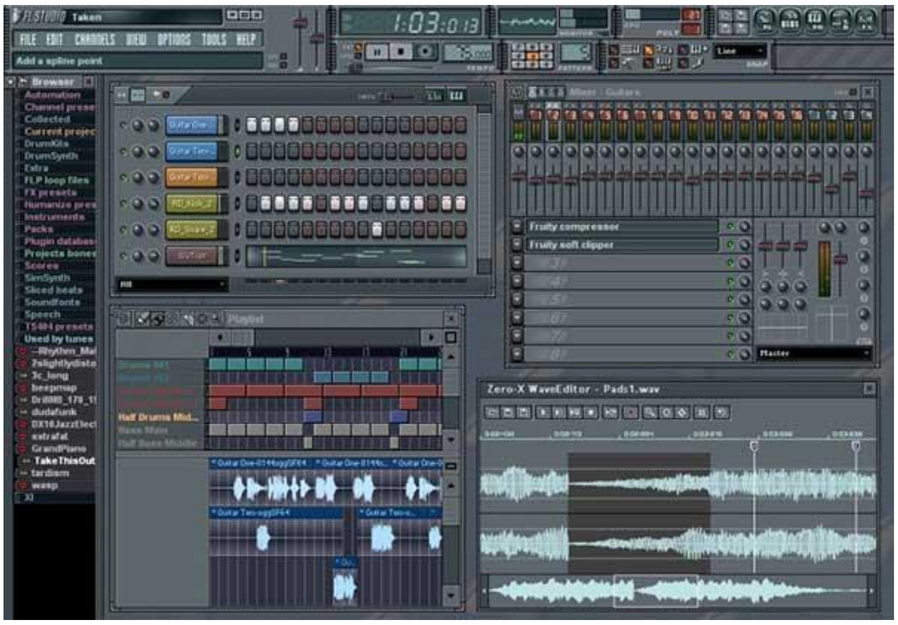

FL STUDIO
FILE EDIT CHANNELS VIEW OPTIONS TOOLS HELP
Browser
Playlist
Zero-X WaveEditor - Pads1.wav

Figure 1 – FL Studio screenshot of interface.

V Motion Project [7]

The V Motion Project is a Kinect system, which involves making music with the movements of a Kinect controller. The Kinect system detects the movement of the user, while an interface is projected on to a wall, allowing the user to see his own "data shadow" and can move the hands to touch a button that plays a sound. The systems works by connecting a Kinect camera to the music software Ableton Live, which is used by DJs and musicians to remix their songs during live performances. A set of different gestures has been made and allocated to their own function in the program, so that the user controls different tools by doing the different gestures.

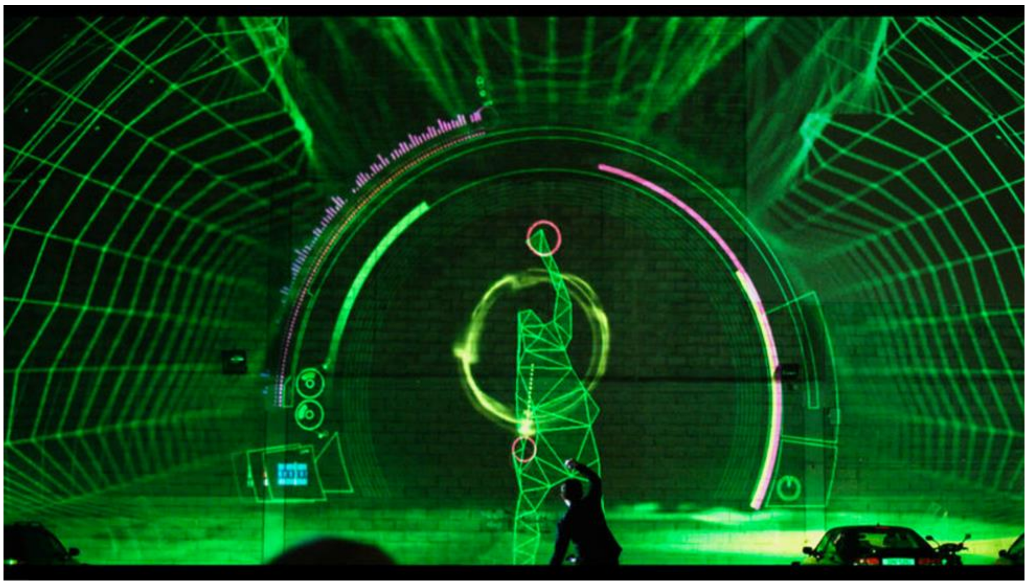


Figure 2 V Motion Project

Technitone Web

Technitone is a web-audio composing and sequencing software, which lets the user create music with a drag-and-drop function to a grid, which plays the tones in sequence. The software contains 23 different instruments and tones that can be used and allows the user to record custom sound samples and upload them. After being placed on the grid, each different tone can be adjusted to a setting preferred by the user, be it volume, pitch, distortion or delay changes. This allows for great creativity and can be shared with others [19].

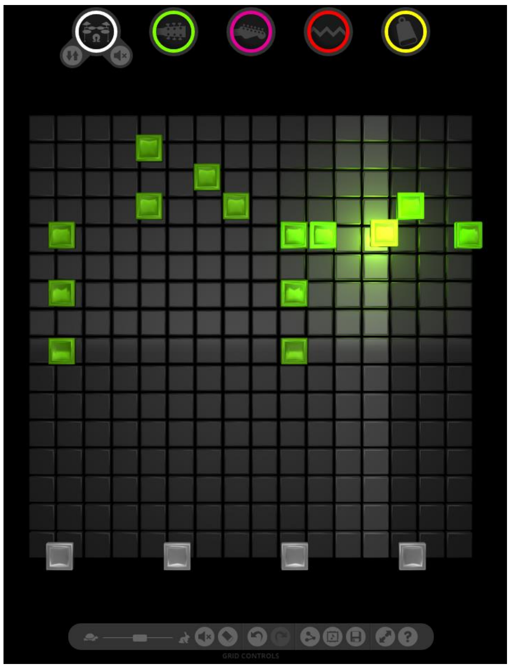


Figure 3- Tecninote Web

Piano Staircase [12]

The Piano Staircase is a project made famous by the website ‘thefuntheory.com’, a website that promotes fun and intuitive projects, which can change people’s behavior in terms of health. This project was an experiment to see if they could get more people to take the stairs instead of the escalator in a Swedish train station. To do that they covered the stair with electronic pressure sensors and sheets to make the staircase look like a piano, and when a person stepped on a key, a note from the piano played. With this they turned the entire staircase into a giant piano, which played music when people walked on it. They used this to influence people to be more physically active by incorporating a different and fun way to climb the stairs. This is also known as the ‘nudge theory [17]. While the Piano Staircase was active, about 66% more people chose to use the stairs over the escalator.

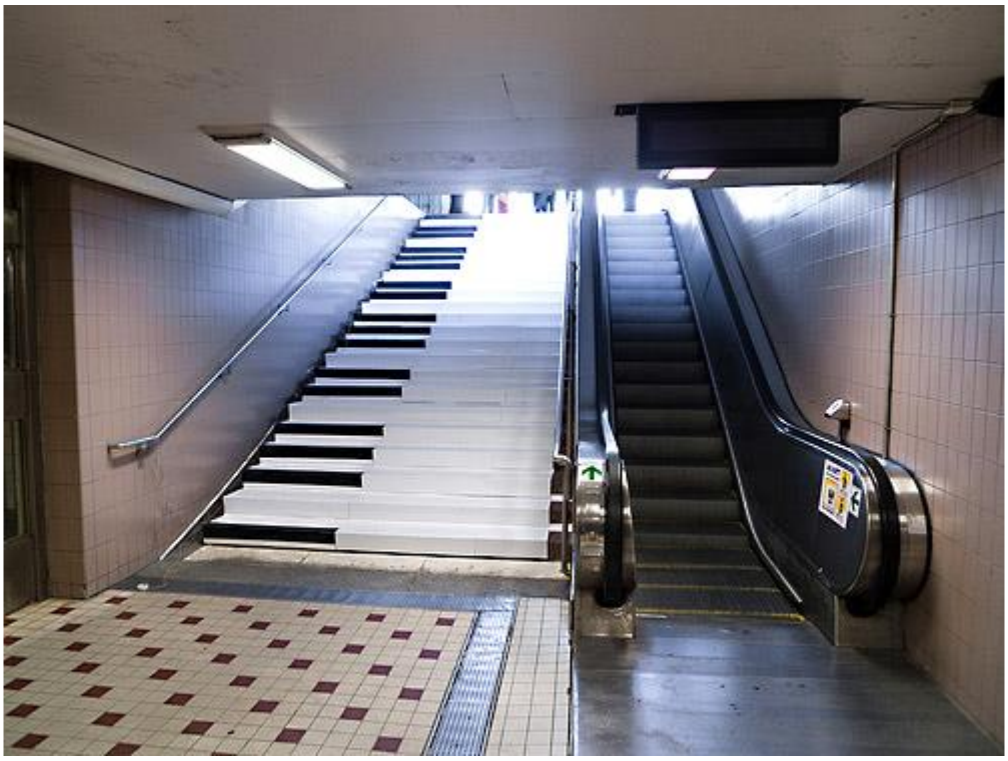


Figure 4 - Piano Staircase

3.3 Competitive analysis

This will be an analysis of existing systems and designs trying to enable users to create music through a digital interface and setup. Several similar systems currently exists that to some extent touches upon this idea. This analysis will function as a walkthrough of existing projects or designs competing with this idea. This is done not only to see the difference, but also to ensure this project will not run into pitfalls of the past.

AudioCrawl

AudioCrawl was built by a web audio enthusiast Adam Chambers and the team of Digital Surgeons; [20] a design and innovation firm. AudioCrawl shows a visual approach to creating audio, which can be utilized by more users at the same time. The goal of this project is to help build a community hub for anyone interested in creating music.

AudioCrawl Soundboard [21]

This is a different and minimalistic interface for making sounds and music, combined with an easy to use interface. There are 10 different sound channels that will constantly repeat itself with its corresponding sound. You drag them into the inner circle to get the sound to start playing. The user can decide what sounds will be active at any given time, by dragging the corresponding buttons into the inner circle. It is a collaborative experience where other users might enter and start manipulating the soundboard after their own ideas.

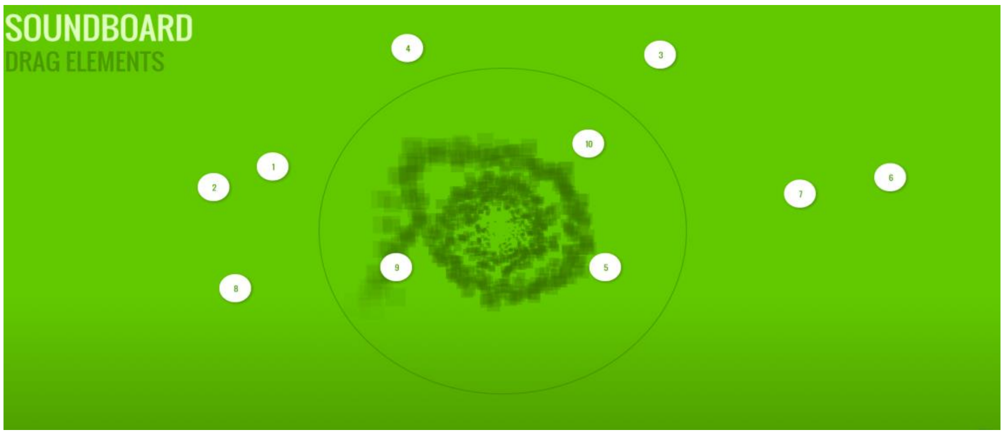


Figure 5 - screenshot of the soundboard on AudioCrawl.

The speed, volume and effect of the sound channels changes from where in the inner circle you place them, closer or further from the center. The graphical representation is that of a vortex, moving based on the sounds that are created from the software. After using the software for a short time, the actions and reactions there became slightly repetitive, since there was only a limited amount of different combinations between the sounds, combined with a short repeat time.

A possible flaw in the soundboard software, is the negative effects others user may have on the experience. Since more than one user can be active at a time, they affect each other's works. Should these users have conflicting ideas about what to create, they may negatively affect each other.

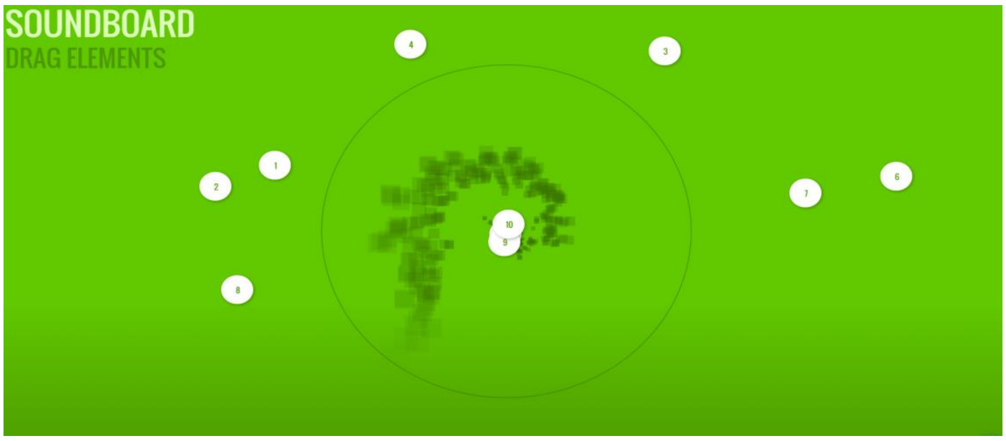


Figure 6 another screenshot of the soundboard on AudioCrawl.

Tonematrix [22]

This concept is one of the simpler tone-creation tools the group has found during their research. The basic concept is that a grid is laid out in front of you, and by clicking the different boxes that make up the 16x16 grid, they become active and play a sound depending on their position on the grid. The x-axis represent time(beat) and the y-axis musical notes ranging from low to higher pitch. The program runs on a timer, that initiates as boxes become active and runs a total of 16 beats before looping itself to the beginning.

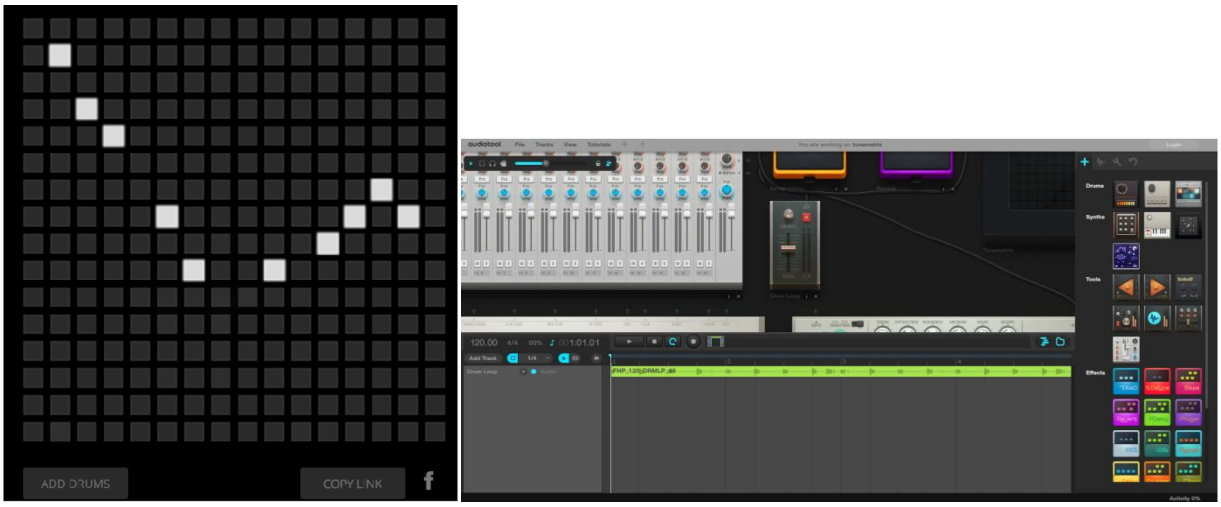
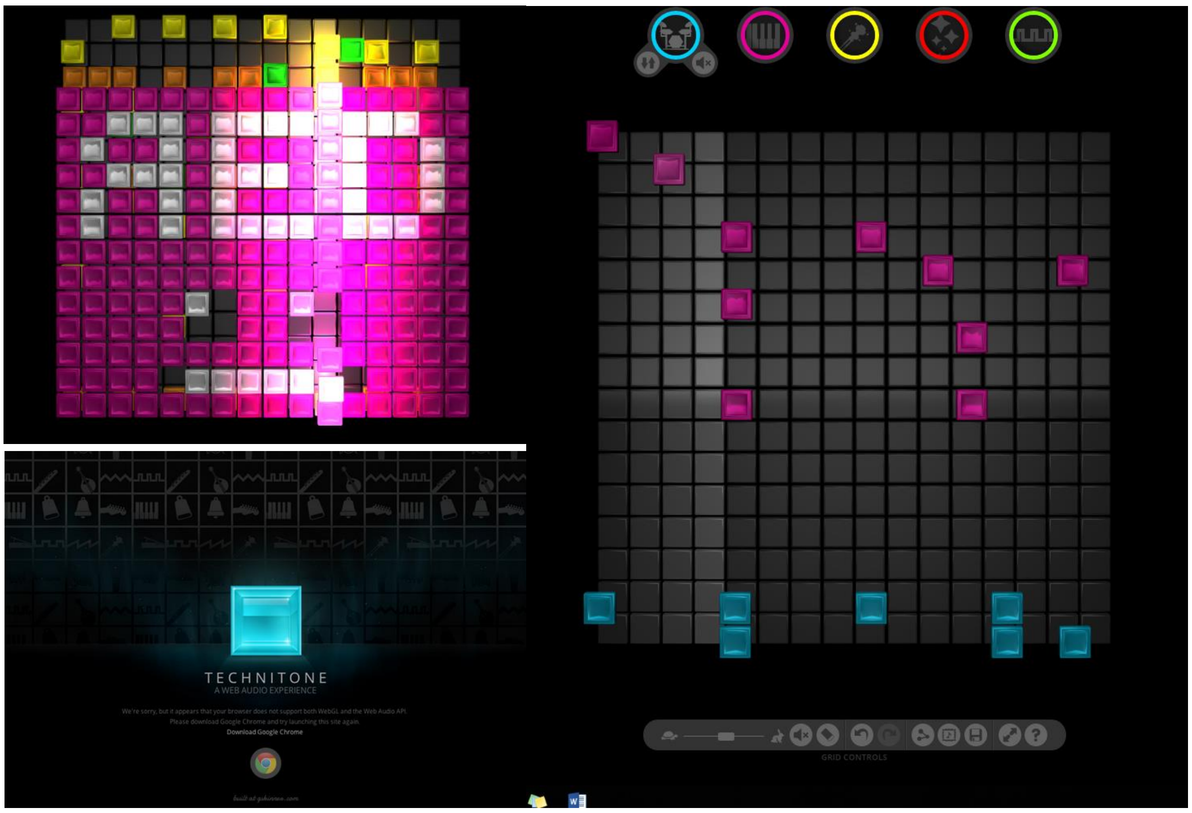


Figure 7 - Pictures of both the Tonematrix (left) front and the Audiotool (right) software beyond the front page.

The Tonematrix is very limited in the sense that making multiple tracks or developing a complex music track with multiple layers is unavailable unless the user goes into the much more complex software behind the front. The Tonematrix itself promotes a much deeper and more complex music creation software called "Audiotool", and is available behind the "Add Drums" button on this front page. Picking up how to use the Tonematrix is very simple, because of the minimalistic interface where everything is very deliberate and conveys exactly what it needs to in order to introduce the user into melody creation. Tools such as Audiotool requires the user to have knowledge of composition and of utilizing complex music software (different synthesizers, effect, balance-levels, etc.), however the Tonematrix enables users with no or very little experience to pick up and immediately start creating their very own music without the risk of it sounding bad.

AudioCrawl Technitone [19]

Technitone is an audio-tool for creating Web Audio easily through a browser on the internet. It is built by gskinner [23] and is a fusion of WebGl, Canvas, CSS3, JavaScript and the Web Audio API . You create music in the program by placing tones on a grid, which then plays the corresponding sounds in much the same way as Tonematrix. Among the options you can choose your instrument among 15 different kinds and personalize the sound further, by adjusting audio filters and effects at the interface. There is also an option for creating your own sounds by recording it through the computers microphone. The software offers simple ways to save, share and edit existing compositions made with the program. Players can view others' saved tracks and use them to start their own collection. If you go to the initial Technitone.com you can also collaborate with up to five users in a real-time multi-user music session.



We're sorry, but it appears that your browser does not support both WebGL and the Web Audio API.
Please download Google Chrome and try launching this site again.

Figure 8 - different overviews of Technitone.

Technitone focuses on different aspects than just the sound, such as the visual aspect, along with the technology and design. The team behind Technitone also reveals parts of their code and their knowledge to others interested in the software. Technitone applies several animations to keep the users engaged while using the program, such as tilting or in other ways manipulate the screen, should the user be idle for a short time. Technitone implements a slightly more complex interface than Tonematrix that could at times appear confusing to the user. The amount of options immediately available to user can seem overwhelming at first glance, and may discourage some from continue using it. Many aspects was left up to the user to figure out, like what effect would produce what results. A more simplified interface might help on this, as well as a minor tutorial or tips as the user engages in the program.

The V Motion Project

This V Motion Project was a collaboration between several people, including music producers and dancers, to make a system that turns motion into music and graphical visuals. Among the involved was Custom Logic and Jeff Nusz; tech lead on Google's Data Arts Team in San Francisco [8].

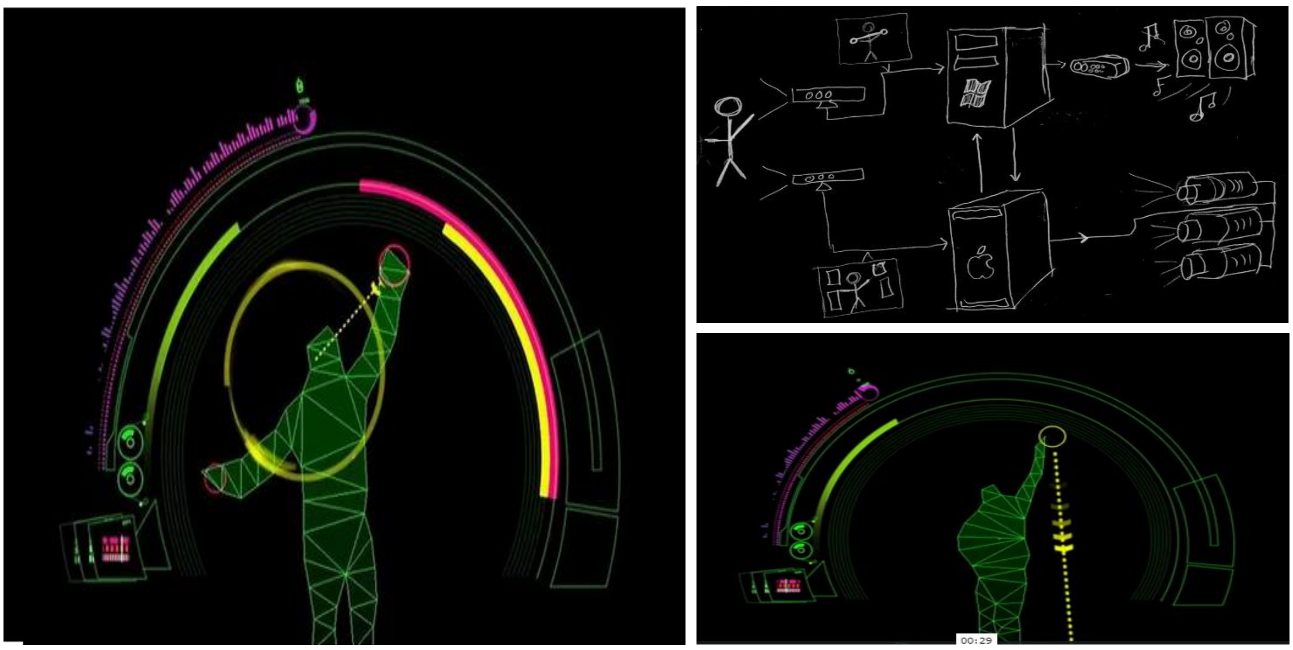


Figure 9 –shots of the V Motion Project in use and a sketch.

The virtual "keyboard" displayed by a projector acts as way to affect the music in a natural and intuitive way. The user engaging in the activity is also represented by the projector, to give him or her a sense of awareness. The user can control the bass, the low-frequency oscillation (used to create dubstep “wobble”) or simply create drums and other sequences for the song. With experience, users can create entire songs, as in the example by a Hip-Hop artist: “Can’t Help Myself” [15].

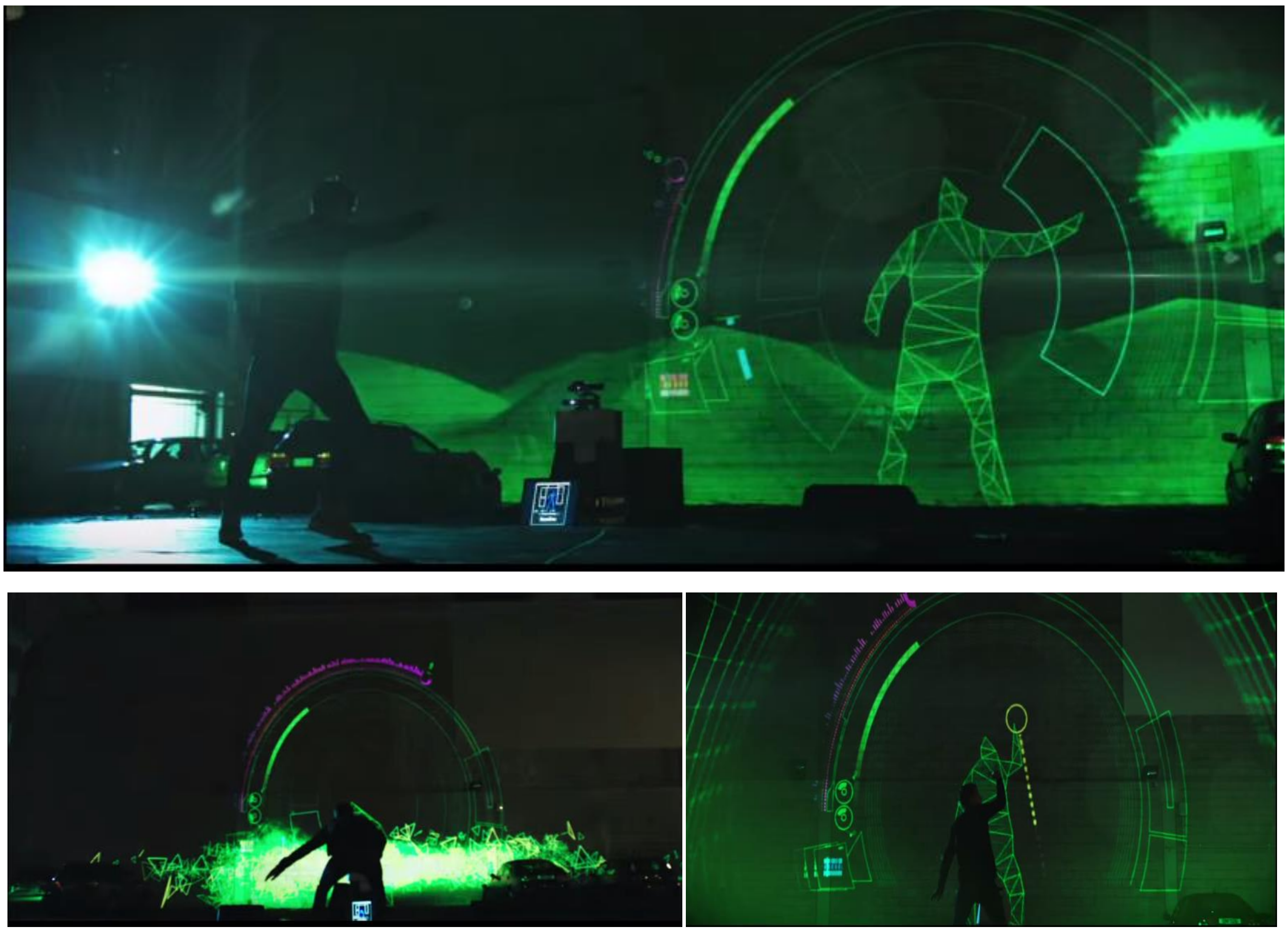


Figure 10 – shots from a video of the V Motion Project in use.

The interface is artistic and visually pleasing, however the system is complex and difficult to learn. Furthermore, the system requires a large setup, both in terms of machines needed, as well as sheer space. It seems as if the system is more oriented towards performers looking to create interesting shows, than to the average user. Nonetheless, the design itself with a virtual "keyboard" to each side of the user seems easy to understand, and must be natural to interact with. In regards to this project, it is clear that the complexity of the V Motion Project is considerably higher than the goals of this project, yet it is an interesting case in terms of how to visualize music.

4. Final Problem Statement

4.1 Motivation

Today, creating music in a digital environment is a complex process. The entry level of current existing solutions can seem intimidating to users with little-to-no knowledge of music creation software. As can be seen in examples like Audiotool and FL Studio, the user is presented with a huge variety of different tools that the user may not be acquainted with beforehand. However, solutions like ToneMatrix and Technitone are examples of ways to create music immediately, with no knowledge of music creation software nor theory.

At the same time, the V Motion Project inspired the group the rethink the control scheme usually associated with traditional music creation software. If the goal is to create a solution that allows the user to jump immediately into creating music with no barriers, why then impose the usage of tools such as the mouse and keyboard onto the user? The V Motion Project suggest an interesting approach to this, by using image processing to allow the user to manipulate the music creation software with no external tools.

Systems like ToneMatrix [22] and Technitone [19] are sources of inspiration for the system and its interface. Furthermore, projects like Piano Stairs [12] and The V Motion Project [15] inspired the group to investigate how to combine an interactive interface with social interactions. As Koch discusses [10], socially constructed 'displays' that bind people together for a moment by in a public environment, can help support community awareness.

Considering the above sentiments, the focus of this project will remain on the navigation and interaction of the solution, not the music or theory. The music and the theory behind it, at least for now, should only remain a tool to bring forth his process. Music creation should be an intuitive creative process, and should not require heavy investments in time, knowledge or experience. It should be about simply extending your hand and start creating music, where the process could be used for social experiments.

4.2 Research question

How can we make an interactive way to create small music sequences by using intuitive controls implemented through image processing, and possibly use this to support social activities in a public space?

4.3 Success criteria

- **Provide an easy-to-use interface**
  The interface should be easy to use and the program should be easy to get accustomed with.
- **Implement a responsive and precise navigation system using Image Processing**
  Implement image processing concepts into the project and create controls for the solution, using C++ with OpenCV libraries.
- **Recognizing different hand gestures by counting the amount of fingers**
  Implement hand recognition into the program with the help of OpenCV to be able to count the fingers shown, for use in gesture recognition.
- **Include several hand gestures**
  To properly control the actions of a music creation program, the hand recognition software should implement several different hand gestures. These should be used to control individual commands in the music program.
- **The controls of the music creation program should be intuitive and natural**
  To keep users engaged whilst using the solution, the controls implemented through the image processing software should feel intuiative and natural to the user. The controls should aid the user in getting accustomed with the use of the system.
- **Evaluate the prototype in a public area**
  In order to investigate whether the solution can support social activities in a public space, the system should be evaluated in a public area.

5. Design

5.1 Conceptual model

Designing a solution can be split roughly into two parts: conceptual design and physical design [2]. Conceptual design is concerned about creating an abstract conceptual model that can help the users understand and navigate the solution. Physical design, on the other hand, is more concerned about how to create the specific parts of the design, and how to implement these features into the solution. This chapter is going to look at the conceptual model for this project.

When creating a conceptual model, the important part is to make it resemble the user's mental model as closely as possible. A mental model, is the idea users have of how to use a system and its' functions. This could be things such as knowing that the icon for the "save" button in a word processing program should be a floppy disk, or expecting that a software application has a menu that allows you to adjust certain aspects of that program. In order to create a design that is easy to understand for the users, it is therefore of great importance that the conceptual models that the designers use has to closely match the mental model of the users. Otherwise the mixed expectations of what the system should be containing/doing may lead to confusion for both designers and users [2].

Objects

To help achieve this match, a better understanding of the solution can be formed by considering what objects and actions the solution should exist of, with regard to what users might expect. This will be done by making a list of objects and their attributes, their actions and how they can be manipulated (if they can be manipulated at all).

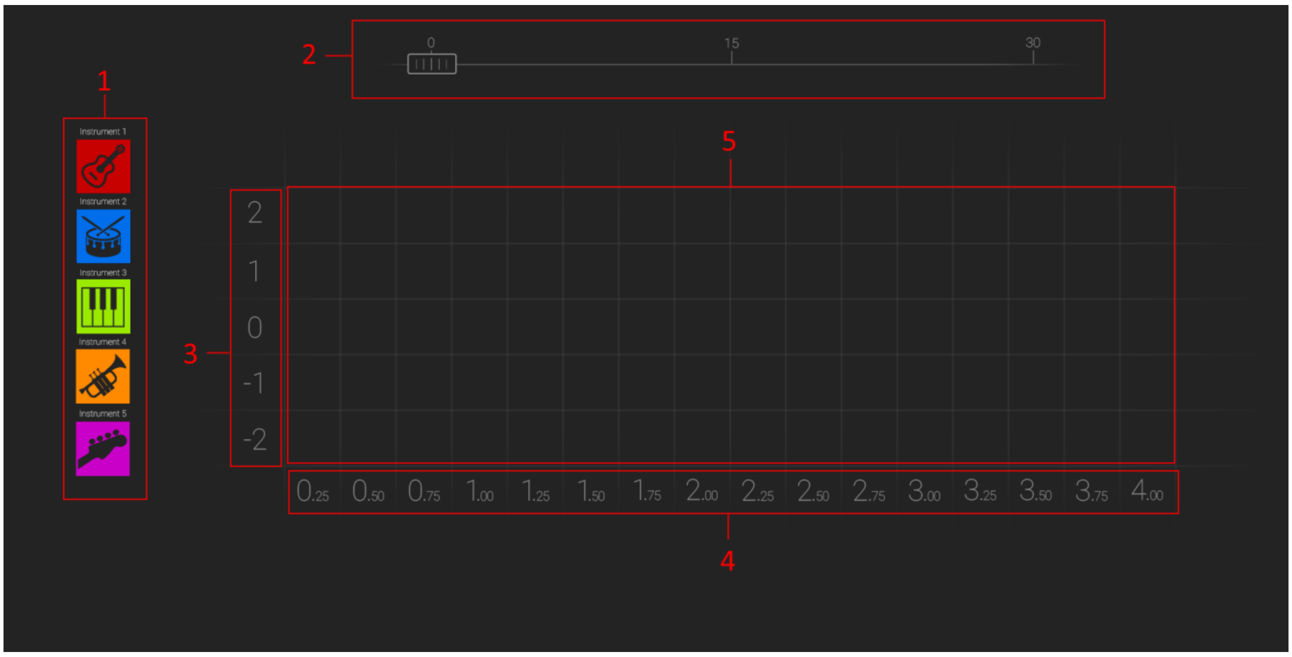


Figure 11 - an early design samples, with highlights of the different types of objects present in the scene.

At first, it must be considered what objects make up the solution. Looking at figure 11, one of the early design samples can be seen, with different group of objects highlighted. These objects are:

1) The instrument blocks
2) The slider bar that controls the view of the scene
3) Indicators for pitch level
4) Indicators for beats
5) The grid where the instrument blocks may be placed
6) Music player, this object is not present on the screen but is still essential to the solution

It should also be considered what actions each of these objects have. The instrument blocks should be able to attach to the grid when placed over a valid grid section, as well as notifying the music player that a tone of the instrument the block represents is now active at the position of the block. The blocks should also be capable of letting the music player know what pitch of the instrument that should be player, based on its position.

The slider bar should be able to change the view of the grid, by either moving the grid left or right. This allows the user to expand the piece of music they are creating, as they are not limited to the beats that can initially be seen.

The indicators for pitch level have no actions attached to them, as they should remain static. The indicators for the beats however, should change their values to match the grid accordingly, in case the slider bar is used to change the view of the grid.

The only action available for the grid, is that it has to be manipulated by the input of the slider bar as was discussed earlier.

Lastly, the music player should have several actions. It must be able to start playing the music, which should be based on what active blocks are current on the grid and their positions. The music player also needs to be able to pause the playing of the tones again. Furthermore, once the last beat of the grid is played, the music player should be able to return to the first beat of the grid and start playing from there again, effectively creating a repeat function.

Object	Actions	Manipulation
Instrument blocks	Attach to grid, notify music player when active (beat, pitch)	Drag-able
Slider bar	Move grid based on input	Drag-able
Pitch indicators	None	None
Beat indicators	Change according to movement of grid	Through slider bar (indirect)
Grid	Move based on input from slider	Through slider bar (indirect)
Music player	Play all active tones, pause all active tones, repeat	Hand gestures

Figure 12 - table showing the different objects, their actions and their manipulations.

Furthermore, it must also be considered how each different object can be manipulated, if that is the case at all.

The instrument blocks should be drag-able, meaning that the user should be able to grab and move them around until the user releases them again, at which point they should either attach to a grid spot if they were released over a valid grid spot (e.g. there are no current instrument of that type already placed at that location), or disappear.

The slider bar should likewise be drag-able, however only on the X-axis, and not outside of its bounds (the user is not capable of removing the slider bar).

Neither of the indicators should be able to be manipulated directly, however the beat indicators should be able to be manipulated indirectly from the movement of the grid, as the value of the beat indicators should change if the grid moves to show the corresponding movement.

The grid should also only be manipulated indirectly, based on the manipulation of the slider bar. When the slider bar is move in either of the X-axis directions, the grid should move accordingly.

The music player should be manipulated through hand gestures. This should at least include gestures for play/pause, but could possible also include more gestures like volume control.

5.2 Design samples

Grid Design – early samples

The grid design was initially seen as a simple way of creating music and making editing more natural and visually intuitive. You could simply choose an instrument/sound and then drag and drop it where you wanted. Instead of composing music, this was a way of creating sound sequences and melodies. This opened up the possibility to place the prototype in a public or an open space such as libraries, schools or pubs. For this to happen, the grid should be simple to navigate while having high visibility and an appealing design. Also the two axis should be easy to understand and give the user a way of making a unique track on their own. While the drop-mechanics and the ability to have more sound objects at the same spot both were appealing concepts, the general idea was to keep the grid simple and intuitive while still keeping many creative possibilities. The prototype should be able to process this interactive system, where the user utilizes gestures as the creative tool.

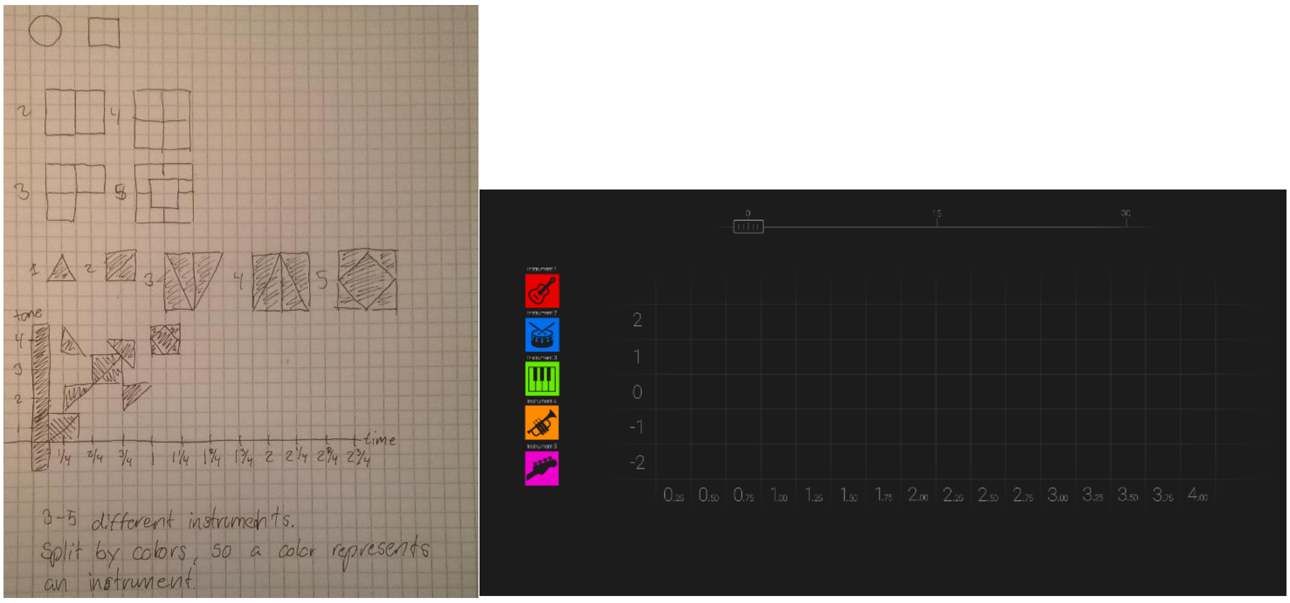


Figure 13 - example of the initial design layout, where the focus was on the possibility of having more sounds at the same spot. This would make it a lot more creative and "free" interface, which were some of the original thoughts and concepts. One initial idea was a y-axis of tone/pitch levels and an x-axis of a time bar that would then repeat itself after ending. In this way, you could create a nice melody and play it in a loop.

The initial thoughts of keeping the layout simple and natural meant that a grid-system with many objects should still be easy to grasp. The many sound-objects could be easily recognized by perhaps a unique color for each instrument (see figure 13). This combined with the concept of being able to place multiple sound objects in the same field would add more choices to the music creation, while giving a clear overview if you later wanted to find certain sounds to edit or remove.

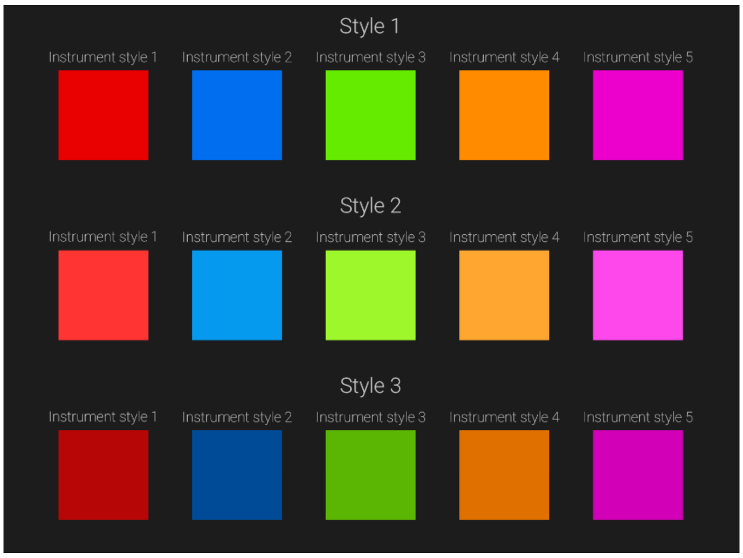


Figure 14 - a sample of how to illustrate both instruments and their varying styles.

As stated above, each instrument is assigned its own color. To develop a more detailed version of the grid, each instrument could also get five variations of styles. This would make it harder to keep simple, but it would add more complexity to the music created. As an example, instrument 1 could keep to red and its different variation in styles could be displayed in brighter and darker versions of red. It might still get confusing and color blind users could run into issues distinguishing the different variations.

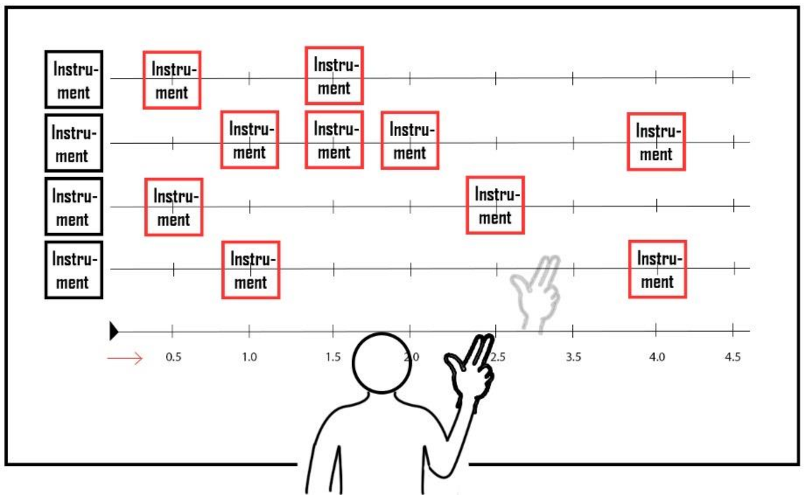


Figure 15 - another initial idea of the Grid Design. This focuses on an idea of how the board could look like when a user is interacting with it. Instruments have already been placed over the grid fields.

Hand Gestures – early samples

The initial approach to the hand gestures was to generate many different ideas and then choose a set of 'core' gestures to control the prototype. Like the color variations and styles of the instruments, more hand gestures could likewise be implemented once the product has been evaluated and the key factors has been tested.

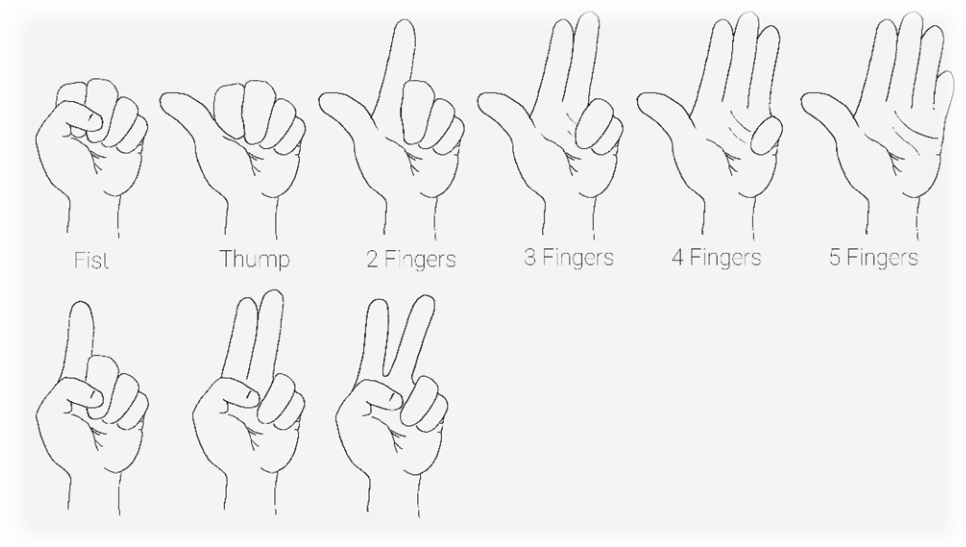


Figure 16 - some different initial hand gestures. They were all perceived as 'easy' and natural for the user to use.

The deciding factor was that the few chosen gestures should not be too similar, but still also not be too vague or difficult to perform. Difficult hand gestures could cause issues for the users, causing the prototype to be exhausting to use. Also it would later be necessary that the chosen gestures were recognizable by camera through computer vision. Whether the different gestures were recognized by the computer could also change the result, from either basing it on the number of fingers or color-coding the fingertips and thereby recognize individual fingers.

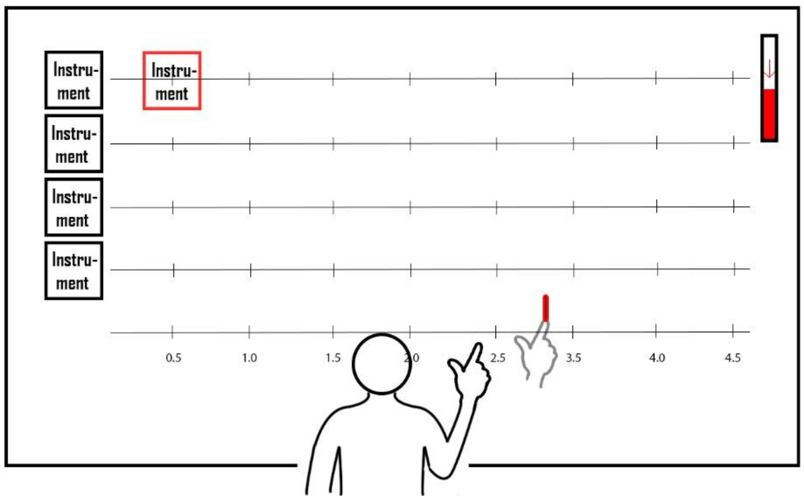


Figure 17 - An example of a possible 'volume control' gesture utilized by a user. The user would simply point his finger upwards while holding their thumb to the side. This would then be recognized and the volume bar would be revealed. Then the user moves the hand up or down to either increase or decrease the volume.

Grid Design – late samples

During the design and implementation of the grid, it went through various changes. The grid was kept simple and intuitive. Instead of 'dragging' objects unto the grid, this time the user simply had to click the fields on the grid to activate them, allowing them to play a sound. However – nothing would happen before you started playing the generated tone-sequence. You could do this by either pressing the 'play' button or use the specific hand gesture for this function. Then it would be up to the user to either have the tones playing while removing/adding them or choose to work in silence and then 'play' the sequence afterwards. This new grid design was even more simple by changing the y-axis to five rows of 'C', 'D', 'F', 'G' and 'A' key tones. Originally, the first prototype was using a xylophone sound, but this was not initially considered the final implementation. The x-axis was changed to 16 columns representing 16 equal increments of time; with 8 beats per second (or each increment representing a 1/8th of a second (or) a total of 2 seconds when the grid runs all the way through).

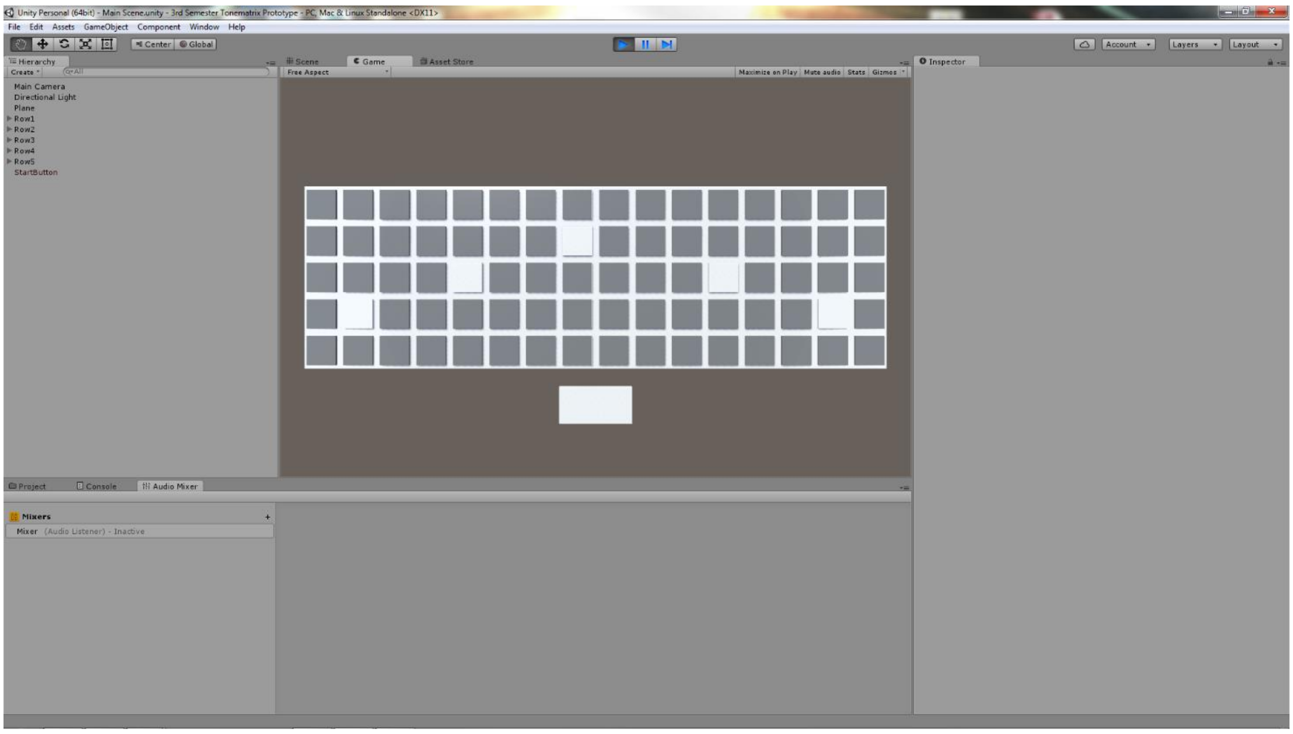

Unity Personal (64bit) - Main Scene.unity - 3rd Semester Tonematrix Prototype - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Global
Hierarchy
Scene
Game
Asset Store
Inspector
Account
Layers
Layout
Free Aspect
Maximize on Play Mute audio Stats Gizmos
Main Camera
Directional Light
Plane
Row1
Row2
Row3
Row4
Row5
StartButton
Project
Console
Audio Mixer
Mixers
Mixer (Audio Listener) - Inactive

Figure 18 - a later design sample of the prototype grid generated in Unity. The user drag their hand up to a gray field and press it using the correct gestures to 'activate' it and turn it white. When playing the tone sequence this white 'activated' field will be a musical increment in the melody along with the other 'activated' fields. The user can also add and remove 'activated' white fields while playing the sequence.

Hand Gestures – late samples

It was also discussed whether the chosen hand gestures, to keep the amount at a minimum, should cover more functions than just one each – for example, the grab gesture could mean 'grabbing' a sound object. Likewise, if the hand was in a specific area, the grab gesture could instead mean 'play'. The same idea was considered during the discussion about the release gesture, but then it would be 'release object/stop playing'. This way, one gesture could have one main function, and then another - more specific - function could apply only within a special area on-screen. This would go well with the earlier ideas of making the volume bar appear if the 'volume gesture' was recognized within a certain area of the interface. However, the opposite concept could be simply to 'grab/press' a button for playing/pausing/stopping the music sequence. This could likewise be used to implement new 'play/stop' gestures, which should be natural and easy for the users to use as well.

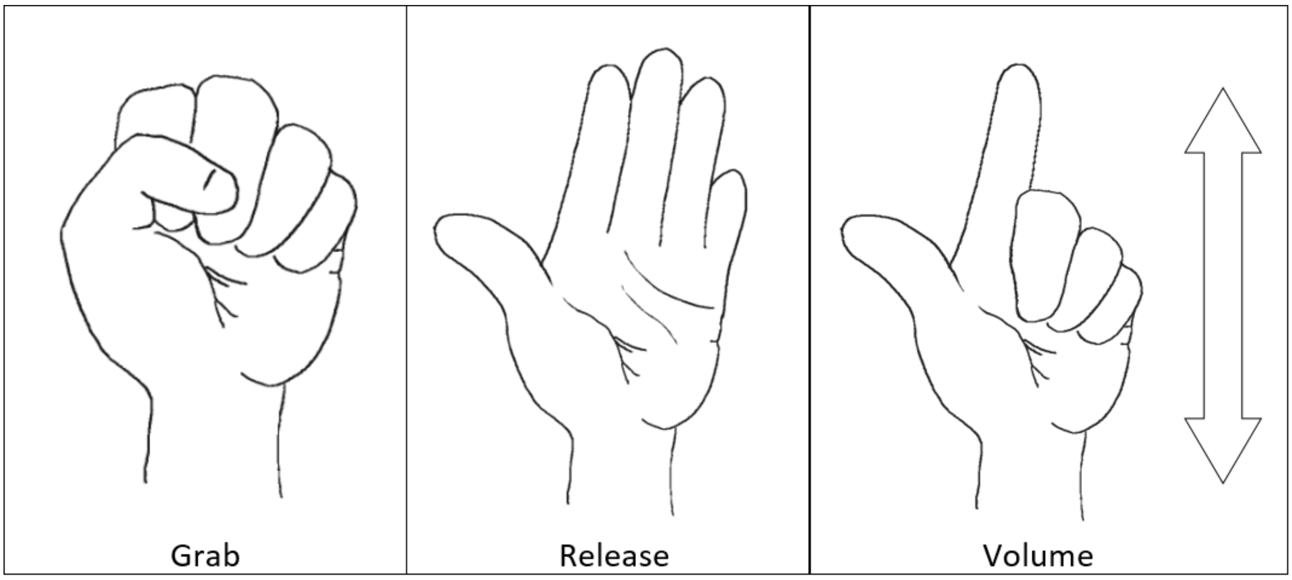


Figure 19 - the three chosen core-hand gestures to control the program. They are all easy to perform and easily distinguished between each other by the prototype.

A later decision was that three gestures would be enough to keep the prototype going. The project was also focusing on the ongoing concept to enable the user to play/pause the tone sequence while creating it – to listen while editing and making changes. If the user had to move their hand down or up to press, a special button to hear the tone sequence it would disturb the flow, so the decision to loop the track was then implemented after the user presses the play-button once. The project later implemented samples of three gestures, which were actually three variants of the same natural movement – Opening your hand, grabbing with your hand and closing your hand.

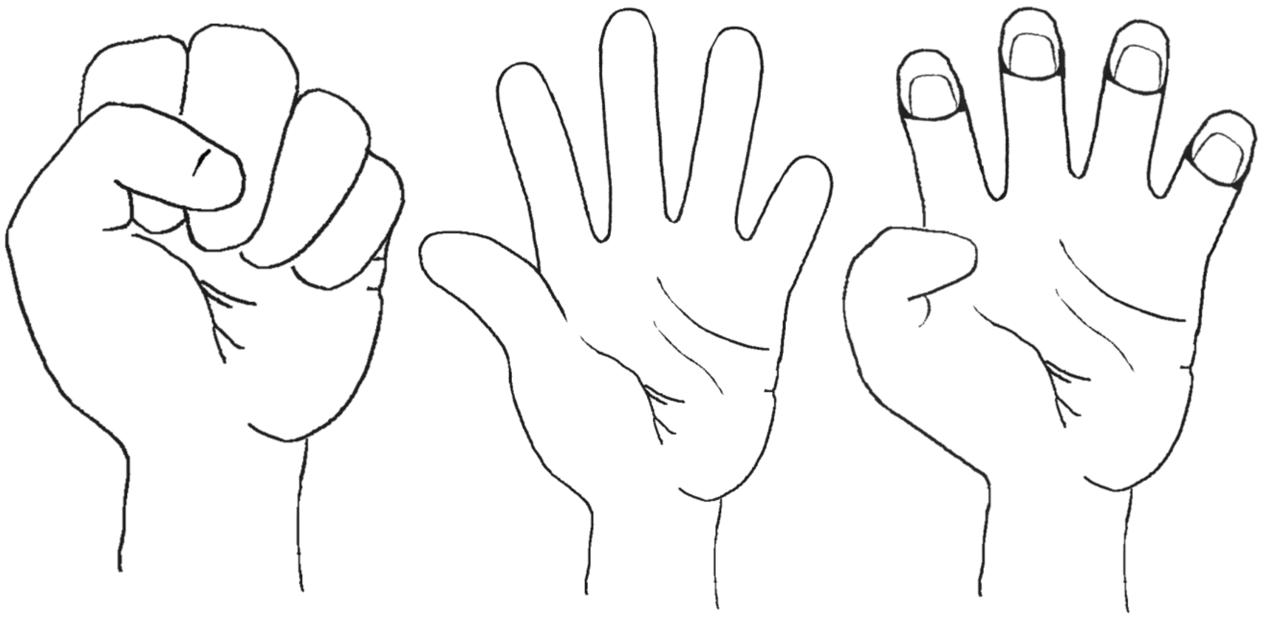


Figure 20 - an example of the three latest hand gestures. To the left, the 'open' hand gesture used to navigate the grid design and releasing 'clicked' buttons/bars. In the middle, a 'click' gesture to indicate when the user is activating a field on the grid or holding a bar to change volume. To the right, a 'play/stop' gesture to intentionally play the tone sequence, or stop playing it, while editing the grid.

5.3 Storyboard

This storyboard will function as an early walkthrough of the key concept in the project assisted by illustrating scenes with following explanations. It is made to ensure a higher understanding of the project: primarily about how it should work and what it should do, but also as a part of the early design-phase. It forces us to picture our thoughts, brainstorms and current studies into a connected story of scenes that illustrates its functionality. To show how the user interact with the design from beginning to end. It is planned to later make a more up-to-date storyboard with more precise screenshots and a more in-depth structure of all the chosen gestures and navigational options. For now, this early storyboard will focus on just the key concepts and early hand signs. The more specific parts of the interface and the order of events may be changed.

Early storyboard

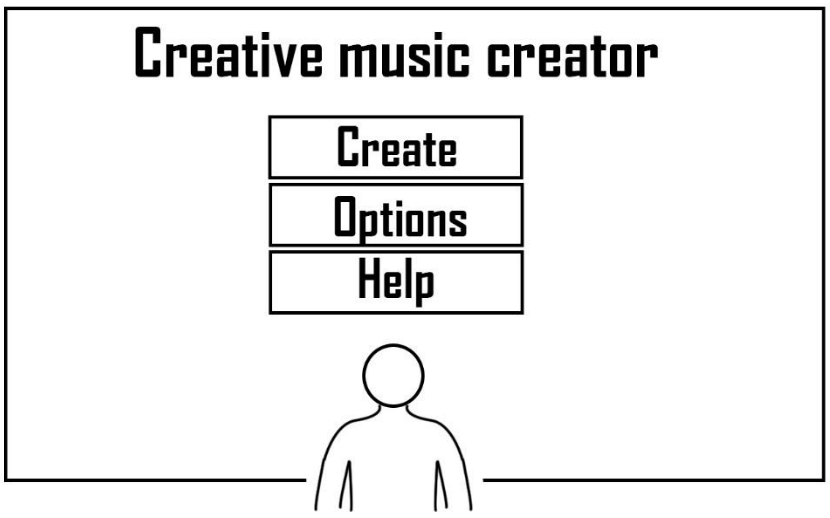


Figure 21 - shows a user standing in front of the creative screen and looking at the interface. Here it can be seen which different navigational options the user has. The title and the titles of the interfaces options should just be seen as examples.

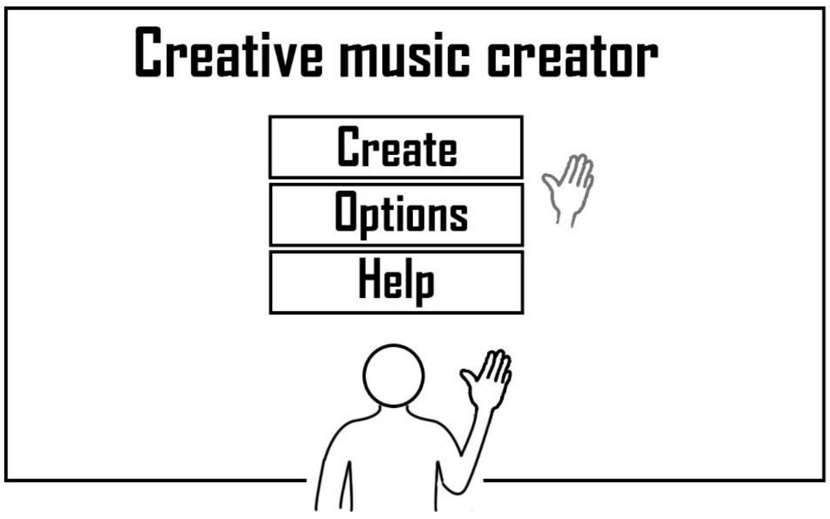


Figure 22 - here the user raises a hand to see the interactive version of it appearing on the creative screen. The user has a glove on, and this glove is used to follow and locate his hand so that it can be used to navigate the interface. Also this is the first designed hand gesture, where the user can move the hand freely around the screen.

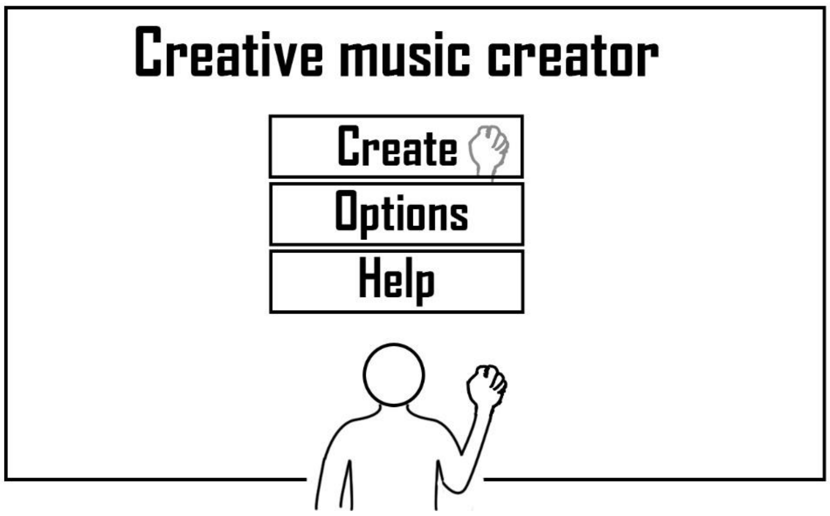


Figure 23 - now the user uses another hand gesture. By moving the hand and closing it “Create” is chosen. The user could have done the exact same gesture to enter “Options” or “Help”.

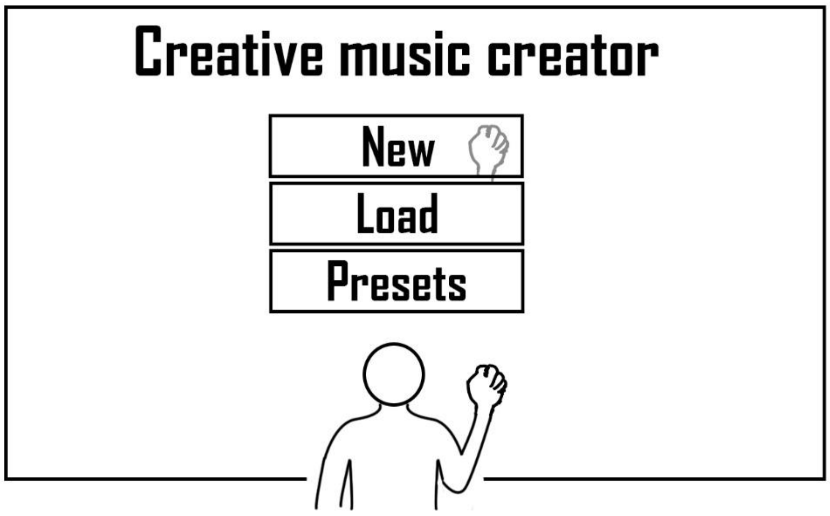


Figure 24 - again the user utilizes the same hand gesture as before to now choose between:” New”, ”Load” and “Presets”. “New” is for creating a new piece of music. “Load” is for opening a past project and continue working on it. “Presets” could have multiple options, perhaps for inspiration for veteran users, as a guiding start for beginners or just to listen to previous work.

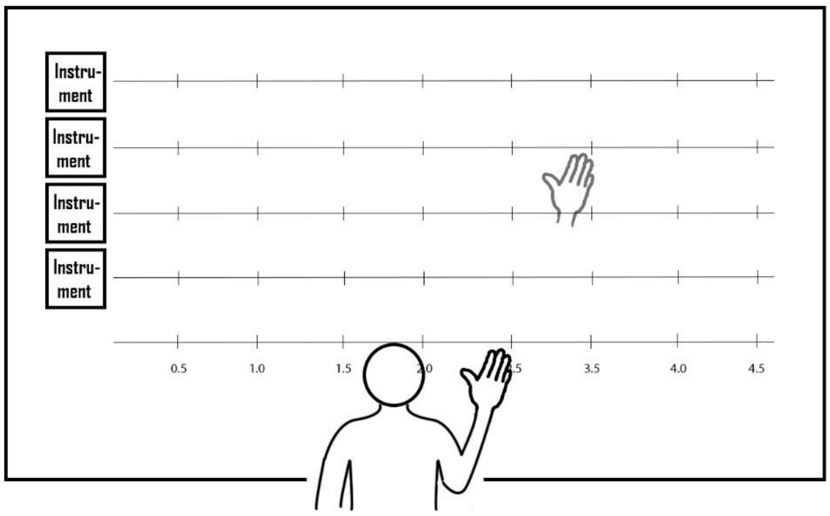


Figure 25 - now the user is seen in front of the music grid because the option ”New” was chosen. To the left the optional instruments are seen. Below the grid there is also a set of numbers indicating the time in which the chosen piece of instrument sound will be played during the audio sequence.

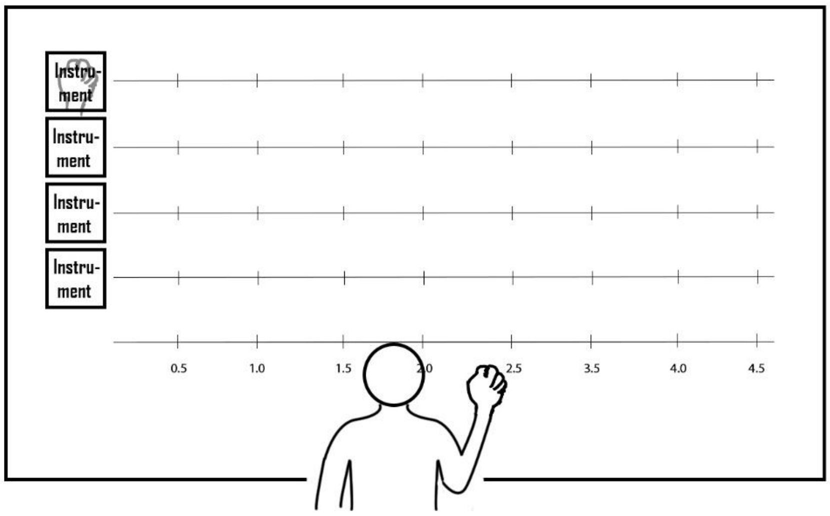


*Figure 26 - s*how the user moving the interactive hand to a chosen instrument. Again the user utilizes the closed hand gesture to press the button.

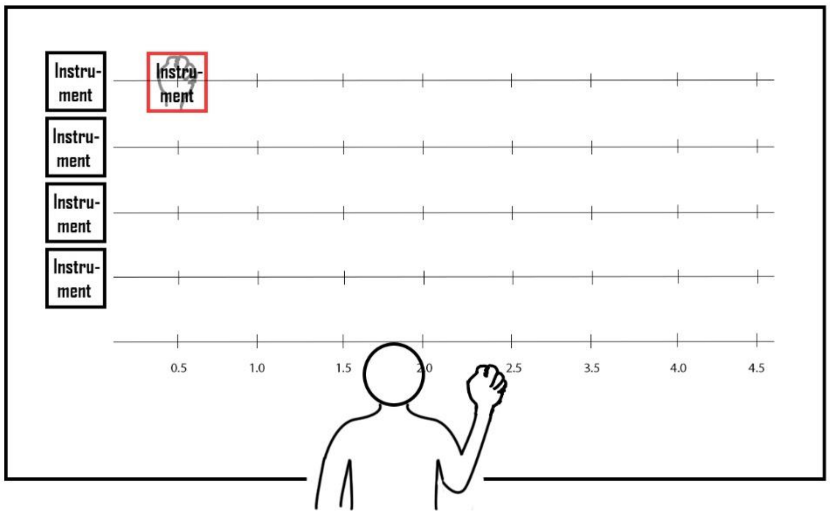


Figure 27 - instead of pushing it the user now is seen dragging it. The user is doing this by keeping the interactive hand closed while dragging it across the grid.

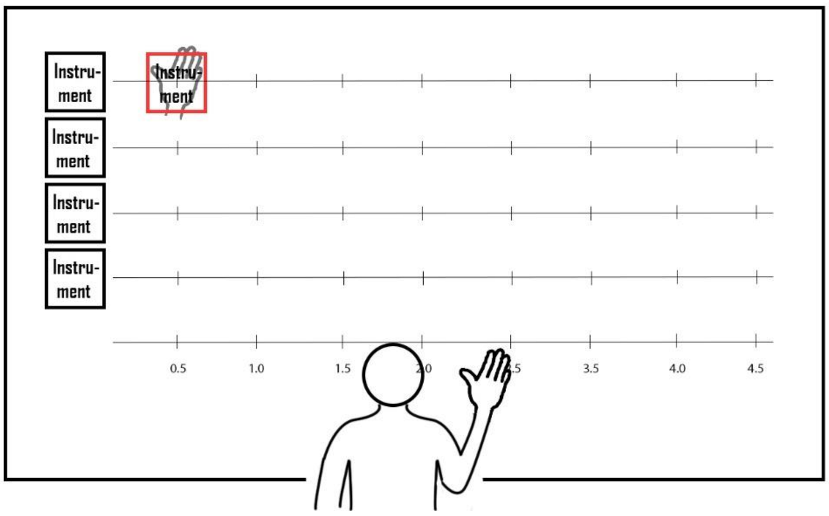


Figure 28 - by opening the hand the chosen instrument is placed on the grid. So now the user has used the two hand gestures to take, drag and place a music object.

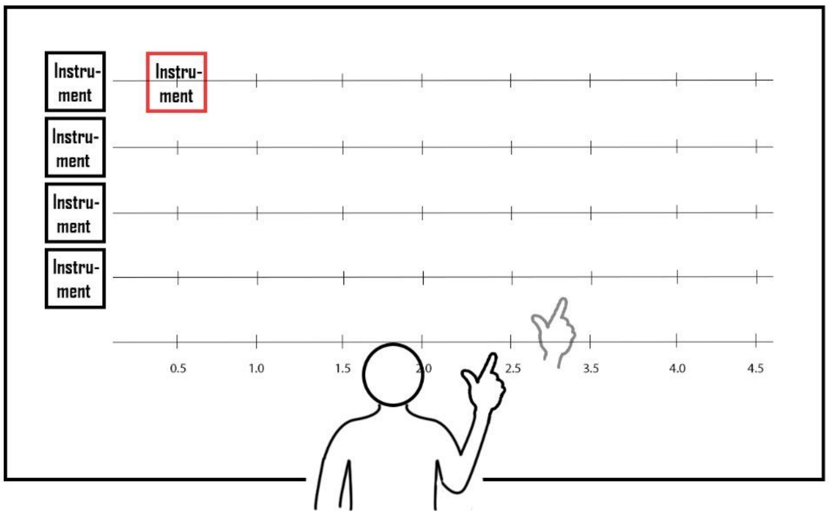


Figure 29 - this is the third hand gesture used to interact with the volume of the music.

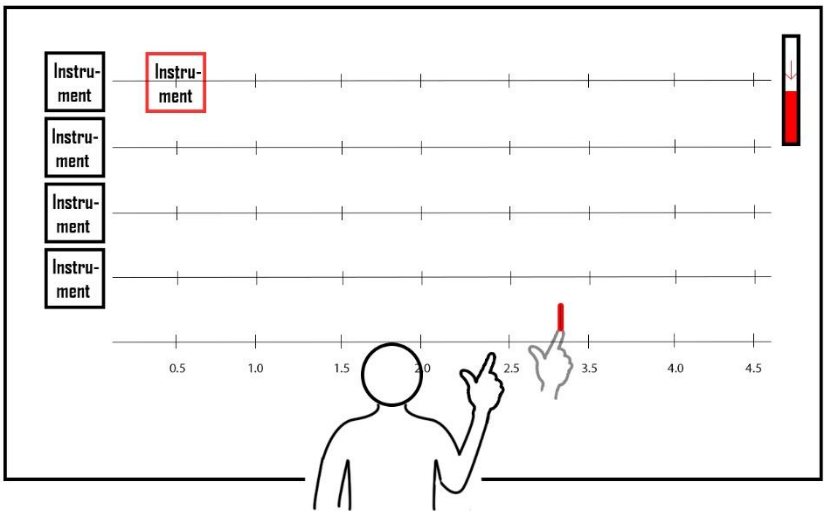


Figure 30 - show the user using the ”volume control” hand gesture to first activate the volume function and then dragging the hand down to decrease the volume output. It the user wanted to increase the volume output, the user would simply make the same gesture and drag the hand upwards instead.

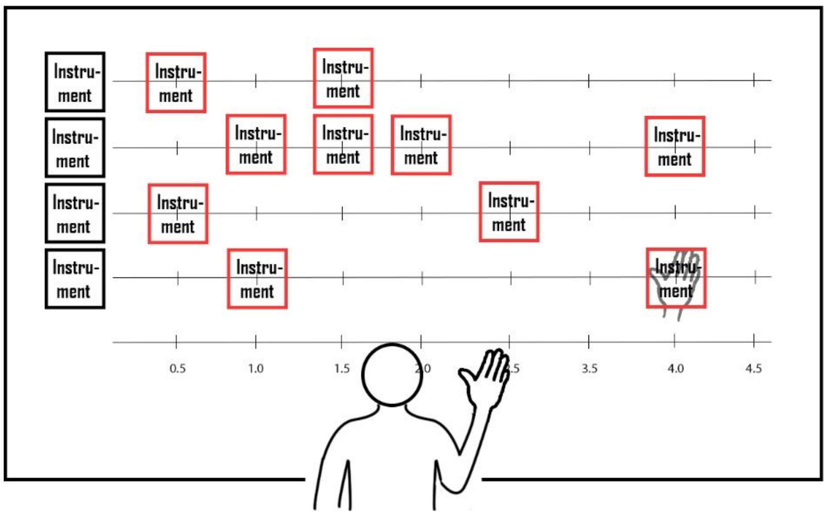


Figure 31 - here the user is seen having already created a fully functional music sequence following the steps showed before.

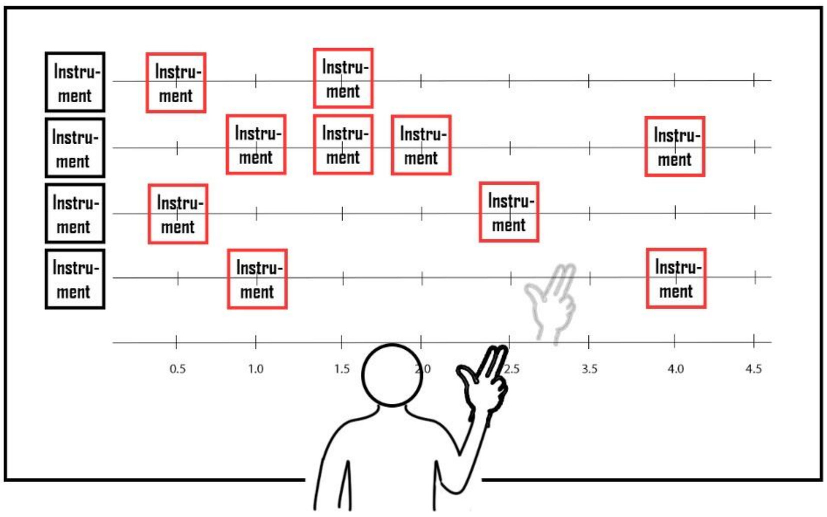


Figure 32 - this is a fourth hand gesture used by the user to play the music sequence. Is can be explained as a "play" command.

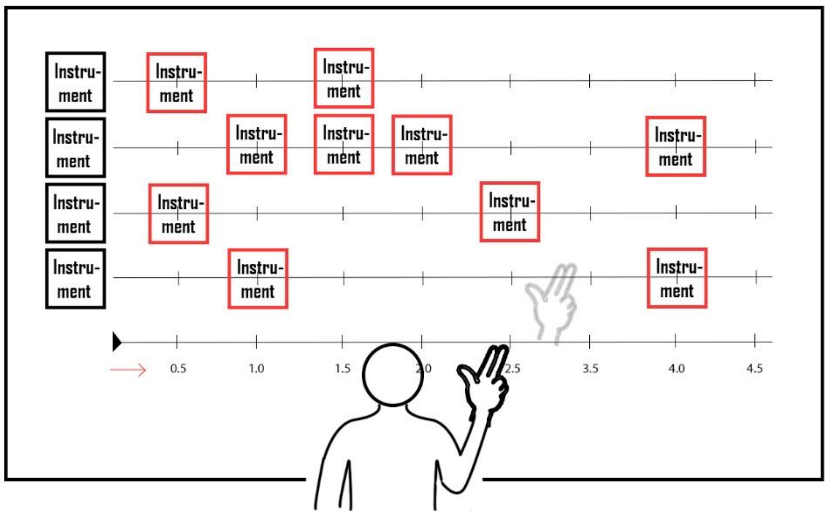


Figure 33 - when the "play" gesture is activated the time bar below the grid will appear and show while the music sequence is playing. The sequence will play from 0.0 to the end of the creation, from left to right.

5.4 Tools used

For the tools used for developing the prototype, the group used Visual Studio to handle the image processing part of the prototype, and Unity for developing the music creator or "tone matrix". Visual studio provides many different libraries, and for image processing purposes. Visual studio is an IDE (Integrated development environment) where you can develop software and web applications. The program itself supports many different languages, but for this project C++ was used. C++ is object oriented and is mainly used for programs that need a lot of processing power to work and therefore the group used this together with OpenCV to help process the OpenCV portion of the prototype. The gesture detection itself was developed using OpenCV.

Open Source Computer Vision (OpenCV) is a library within Visual Studio that is aimed at computer vision through a camera for its processes. This is what the group utilizes to make the program recognize the hand gestures presented in front of the camera. OpenCV has many different applications, but with computer vision one can develop many different things, such as facial recognition, which helps surveillance all over the world today, but also motion tracking which is used in movies and videogames in the industry today to make computer-generated models seem more realistic through a more accurate movement tracking.

The group also decided to develop a simple game in which to test the hand gesture recognition controller and for the music creator, the group used the Unity engine. Unity is a cross-platform game engine, which is capable of developing games for PC and many other platforms, and the group decided to utilize this engine, because it is easy to work with and develop within. The game itself is programmed in C# and this language is mainly used for scripting. The game itself consists of multiple scripts interacting with each other, game objects with different properties controlled through these scripts and the sound samples of a xylophone to help create the user's music track [14].

5.5 Lo-Fi testing

The Low-Fidelity test was performed with one participant at a time. The test was made to get feedback on the main gestures and the overall interface design of the program. The tools used for the test were a laptop, a webcam and some pictures with the gestures and the interface. The participants were interviewed during the test to get the responses from them, while they were doing the given tasks. The tasks are think-aloud where the participant has to say what he is doing and thinking. They also have to do the assigned gestures for each task, pretending that the interface is on the laptop screen.

Procedure:

The participant is placed in front of the laptop and is explained the basic overview for the project and the test. The webcam is then turned on and put on full screen on the laptop. This is to simulate using the interface. They are then given a picture of the interface and is asked to imagine that the interface is on the screen. They are then shown the three gestures: Grab, Release and Volume. Without getting explained how to use the gestures, they are getting assigned on to their first task. The first task is to drag one of the objects from the left side of the interface into the grid in the middle. The participant is asked to do it again with another one, if the interviewer is uncertain whether or not the task is completed correctly. The next task they are given is to turn the volume up and down. If the task is performed incorrectly, the interviewer will hint towards how the volume gesture is performed correctly. When these tasks are done the participants is asked questions about the interface and the gestures.

Setup:

See figure 34. The participant was placed in front of the laptop with webcam, to simulate the software. They were given the two pictures in front of them to serve as help with the different tasks they were given. From the group there were two or three members during the interviews. One as the interviewer, one to write down, and a last one to film and take pictures of the test.

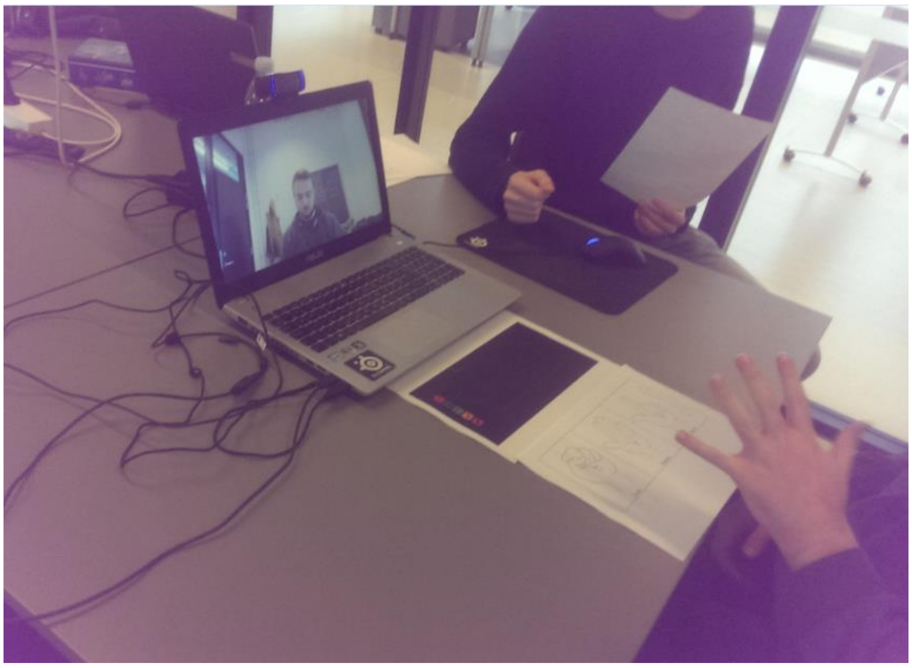


Figure 34 - shot from Lo-Fi test.

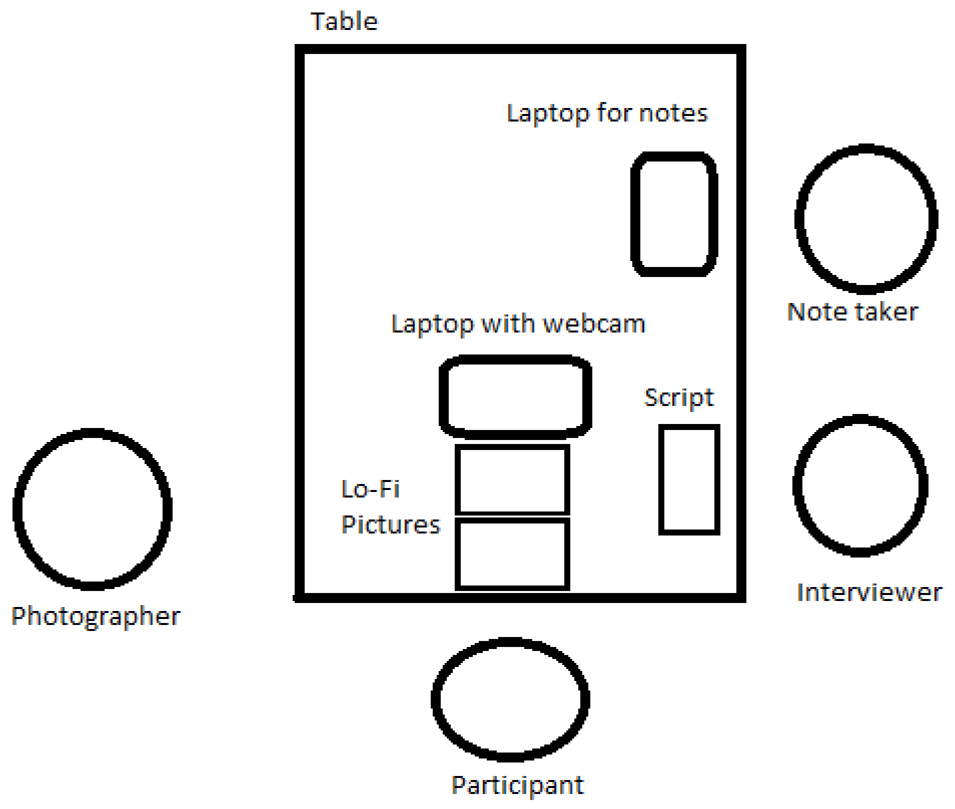


Figure 35 - drawing of the setup.

Participants

The participants for the lo-fi tests consisted of young adults in the ages 18-25, which were all Medialogy students from the 3rd semester. By choosing fellow students, the tests could quickly be done, without the group having to find and interview participants from outside of the university.

All participants had either very limited or no knowledge about music creation software, but they were aware of programs like Fruity Loops, and some of them mentioned during the lo-fi test that they knew *how* that type of software works. Although most participants had knowledge about the current solutions, they weren't confused about how the groups prototype worked, and even the participants that had never used other music software before, could figure out how to navigate the lo-fi prototype. Some had played musical instruments themselves, but none associated with digitally created music tracks. The results from the tests were positive and gave the group a lot of great feedback for the design phase, which will be further described in the results chapter.

Lo-fi results

The procedure of the test was not changed during the Lo-Fi testing and therefore the results should be able to contribute to each another. That said, four out of the ten participants already had experience with music. These four were actually also among the participants with the longest testing time and contributed with the most ideas and creative angles during the process.

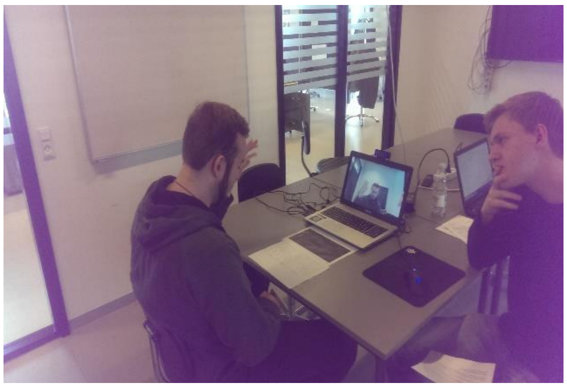
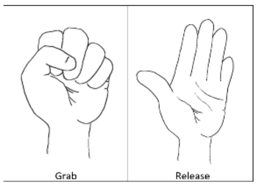
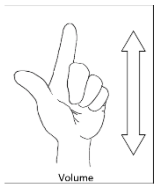


Figure 36 - pictures taken during the Lo-Fi testing on fellow students. It shows how the laptop in front of the participant was mirroring his movements so that it was like actually using a creative music board where you could see your movements. Also papers in front the participants showed the signs and the music grid with instruments. The test administrator was put beside the participant to introduce him, then give him tasks and ask questions.

Gestures



Results from the test showed that it felt very natural for the participants to open and close their hand to control the instruments on the grid. Only one of the participants created his own unique way of "grapping" with just his fingertips instead of closing his entire hand. The next task was to increase the volume, which actually resulted in many failures regarding using the right gesture. In other words, most of the users started using the right gesture with pointing one finger upwards with the thump to the side – but then just kept their hand in this position in the belief that this command increased the volume. When the participants then were asked to decrease the volume most of them got it right by moving this gesture downwards. After doing this, they were also able to figure out the right command for increasing the volume – by moving the gesture upwards. A few of the participants did not get the volume commands at all, and when asked to turn it down they simple reversed the gesture with the pointing finger downwards and the thump to the side. They believed that the opposite of the first gesture would be the correct command and did not get that they actually had to move the hand up and down. After clearing this confusion with the participants, they all seemingly agreed that having a volume bar appearing on the board would make it a lot more natural to move the gesture up and down.

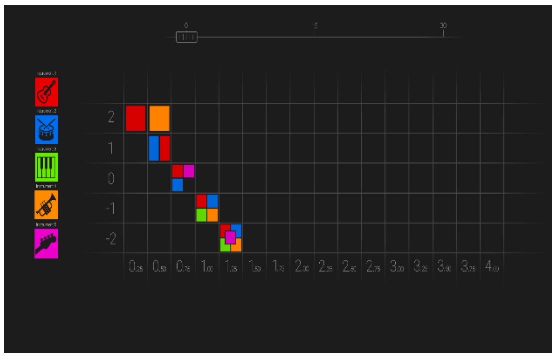


New proposals

A big part of the participants agreed on the fact that the volume gesture should be changed slightly by not having the thump resting straight to the one side. This would make the gesture more natural and relaxing for the hand. Some also believed it would be easier to have something like a button on the screen that you could turn to the sides for controlling the volume. One of the participants, with musical experience, came up with the idea to be able to 'hover' your finger or hand over an instrument to listen to it for a short amount of time.

Moving on was the 'play/go' and 'stop/pause' gesture, where the participants themselves had to come up with own ideas. Some of them passed on doing so, but many also made their own gestures. These were mainly the 'thumps up' gesture for playing the sequence of music and then turning it downwards to stop it. Another was to find new 'release/grab' gestures and then use the current ones for the 'play/stop' instead. A final one was to use the known 'rock'n'roll' gesture to start.

The board



None of the participants understood the 'pitch'-bars in the y-axis of the board without clear introductions first. After the instructions though, they all concluded that the bars themselves was a good idea, but that they needed clear signs or a quick tutorial for new users to get.

Also having the possibility of releasing more than four instruments on the same square could lead to problems removing them or grabbing the right one. The participants felt that the board itself was too dark in its current mockup.

New proposals

One of the participants, also with experience in music, came up with an idea to change the timeline into a sequence of beats instead. Another one stated that the idea of having five pitch-values would not be correct corresponding to all instruments like for example the piano option. Other ideas were features responding to the direct angle of your hand or being able to extend tones instead of grabbing/releasing every single one of them.

Conclusion

The first aspect was the fact that the early implementations of the project should simply focus on one instrument instead of five. After all, the music itself was not to be the main focus in this project, but instead the interface and navigation. Furthermore, the volume gesture should be made more natural and a volume bar should either appear when using the correct sign – or already be shown in the side of the screen automatically.

The board itself should also be altered. The visual design should be different, as well as the 'pitch'-bars and other visuals also needing to be more obvious for the user. Finally, instead of calling it a 'music composing program' it should rather be changed into a design focusing on producing a sequence of tones and melodies instead – this also utilizing the creative grid system better. As a possible gap, participants also made us aware of the fact that people with cognitive issues could have a hard time using some of the gestures.

5.4 Minimum implementation requirements

In order to properly evaluate a solution based on the success criteria stated in chapter 4, the group will have to implement a minimum of the following features:

A program that can recognize the hand(s) of a user, with a strongly colored glove. This program should be made in C++ using the OpenCV [24] library, which gives access to a variety of different tools for image processing. The hand recognition software should be able to count the fingers of the hand that is currently being tracked, so that different hand gestures can be recognized.

At least three gestures for controlling the music program. These should be integrated in the image processing software. These controls should at least be able to control the cursor, allowing to user to move freely within the interface.

A music program should be developed, for use along with the image processing software. This program should at least implement a way to play different tones with specific timing. At least five different tones should be implemented, to give enough variety to create small pieces of music.

The user should be able to toggle tones at all beats on and off, through the controls implemented in the image processing software. The user should also be able to start and stop playing of the tones through one or more hand gestures.

6. Implementation

6.1 Image Processing

Color space

The most common colorspace is known as RGB (red, green and blue) which controls each color of a pixel by mixing the three different colors value. Red, green and blue are known as primary colors and the mix of the adjacent primaries results in secondary hues such as magenta, yellow and cyan. These colors can be represented in something known as a color wheel as seen in figure 37. This wheel represents the hue where red is 0º which is the most top position of the wheel. Green is position at 120º and blue at 240º.

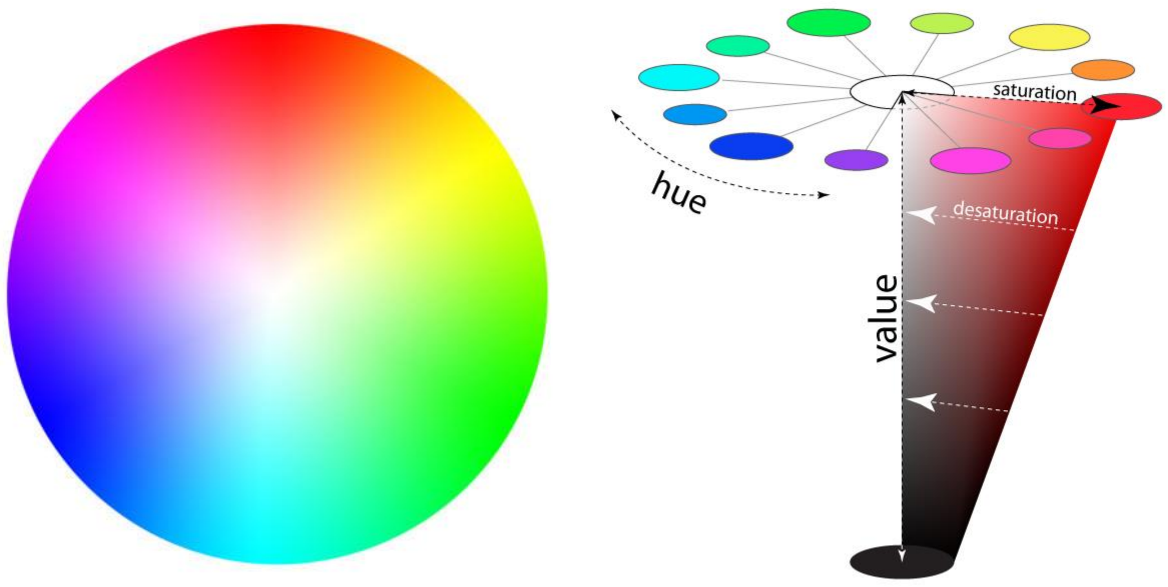


Figure 37 - hue color wheel[6] and the graphical representation of the HSV colorspace [25]

This wheel is used in the color space called HSV, which we use in the image processing part of the program. This format consist of three variables as seen in figure 37.

Hue

The colors of the wheel which is not only limited to be around the outer edges of the wheel. This only explains the color type, not how vibrant or light it is.

Saturation

This value goes from the center of the wheel and out towards the edges. This determines how vibrant and light the colors are.

Value

This value creates the cone shape of the colorspace. This goes from the bottom to the top of the cone, defining how bright the color is from black to white.

Morphology

This is a mathematical method to remove noise or fill in the holes of a blob. This is done by applying a kernel to each pixel. A kernel in morphology is a structuring elements, which contains one's and zero's. It is usually formed as a box or a sphere with a pattern of one's. A box shaped kernel usually preservers sharp edges while a sphere shaped kernel is round in the edges. The kernel is applied with either hit or fit operations. We perceive one as white (255). For each one in the structuring element that we apply to a pixel we note if the pixel in the same position also has the value of one. If this is true, the structuring element hits the image at the pixel position (center of the structuring element). The value stays one and if this is not the case the value is set to zero instead. If all the one's of the structuring element and the one's in the image is in the same position (from center of the structuring element) it is said to fit the image. This is therefore also set to one, otherwise it is set to zero. Remember we are comparing all the pixels in the image but only change one at a time taking the pixels last value into consideration. This is what is known as dilation and erosion [11].

Applying hit to the whole image is known as dilation. It is called dilation as it is increasing the size and filling the holes of the elements in the picture. How big the element gets depends on the size of the structuring element that is applied. It should be noted that increasing the size of the element in the image can also be done by applying a small structuring element twice. The disadvantage of dilation is that it also increases the size of other objects in the image and can create noise if applied immediately. Applying fit to the whole image is known as erosion. This is the opposite of dilation and is decreasing the size of the elements in the picture, removes small objects such as noise and larger objects splits into smaller ones. The effect of this also depends on the size of the structuring elements, the bigger it is the more it removes [11].

Combining erosion and dilation introduces two new terms known as closing and opening. Closing is where you apply dilation to fill in the holes, increase the size of the object, and then apply erosion to bring the object back to its approximate original size. It is important to note that the structuring element applied in both dilation and erosion should be the same. Opening is technically the same thing, you just apply erosion and then dilation to remove noise from the image but preserve the big objects. Applying closing followed by opening will remove the noise of the image, remove small objects and preserve the big objects and filling its holes [11].

Contour

Also known as boundary detection/border following, contour is used to get the edges of an object by looking into how steep the change between the two images pixel value is and change it accordingly. This is easily done on an image that is only two values, black (zero) and white (255). As soon as the image changes from zero to 255, keep the first value and change the values of the rest of the object to zero until it reaches another steep change. This way you can gather the position of the edges and draw them afterwards [13].

Convexity hull

Convexity hull is the minimum convex polygon in which the object can be contained. An example of a convexity hull can be seen in figure 38. This is used to get the outline of the object in total with the holes between the edges of the object. It is similar to placing a rubber band around an object. It will not sit tight around the object but stretched to each outer edge [16].

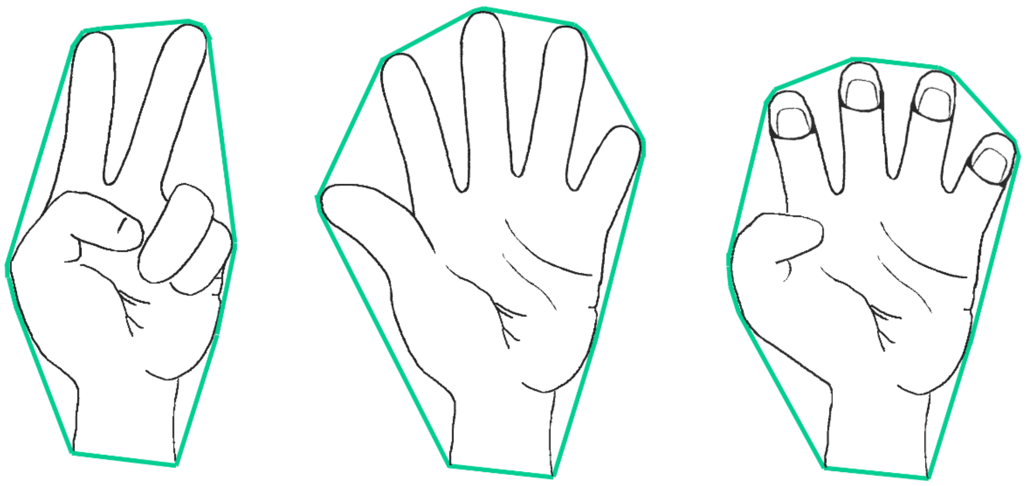


Figure 38- illustration of convexity hull.

Convexity defect

Convexity defect is by taking both the hull and the contour into consideration to figure out where there is space in between the object and its hull. This method is most commonly used to detect an amount of fingers on a hand. An example of what a defect is can be seen in figure 39. There will usually be more defects on a hand than fingers but to distinguish between them is to set a required length of the defects before they are recognised. This way we can count the amount of fingers on the hand reliably [4].

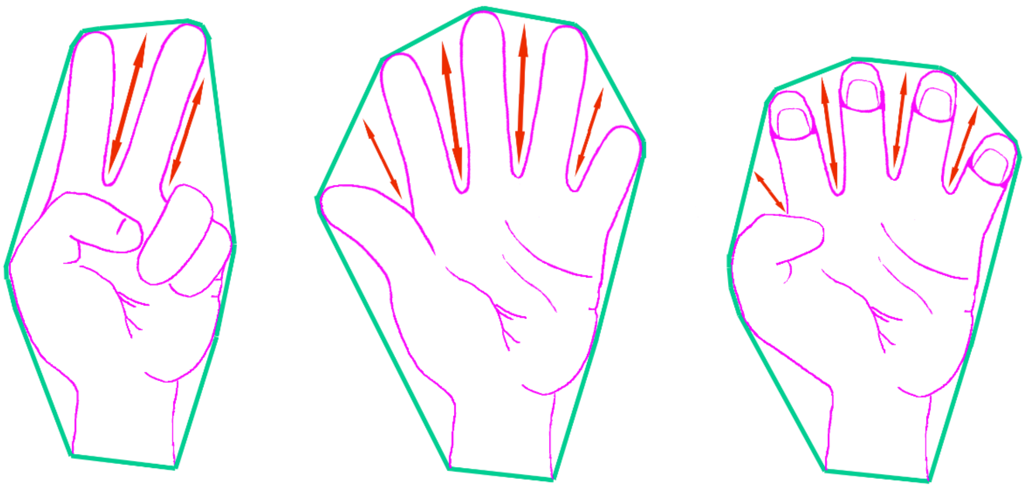


Figure 39- illustration of both contour and convexity hull with orange arrows representing convexity defect.

Tracking and input control

C++ is capable of managing inputs of the computer such as simulating key presses, cursor position and mouse clicking. This can be used in combination with the other tools explained in this chapter to control the standard controls of a windows computer. The area of the recognized hand as seen to the right of figure 40, is calculating its center position in a two-dimensional space returning the coordinates x and y. These coordinates can be applied to the cursor position and move the mouse according to the position of the hand on the camera. If the hand is placed in the top left most corner, the mouse will also be displayed in the top left corner of the computer screen. Depending on how many fingers are present using convexity defect, different key presses can be simulated to function as a hot key in another program or similar. An example of the finger counting is displayed on the left of figure 40. Currently if all five fingers are present, it acts as a mouse release. To simulate a left mouse click, move down your fingers as if you were to grab a handball. Again stretch out your fingers to release the left mouse click. Closing the hand will result in zero fingers and the program will simulate the key press P.

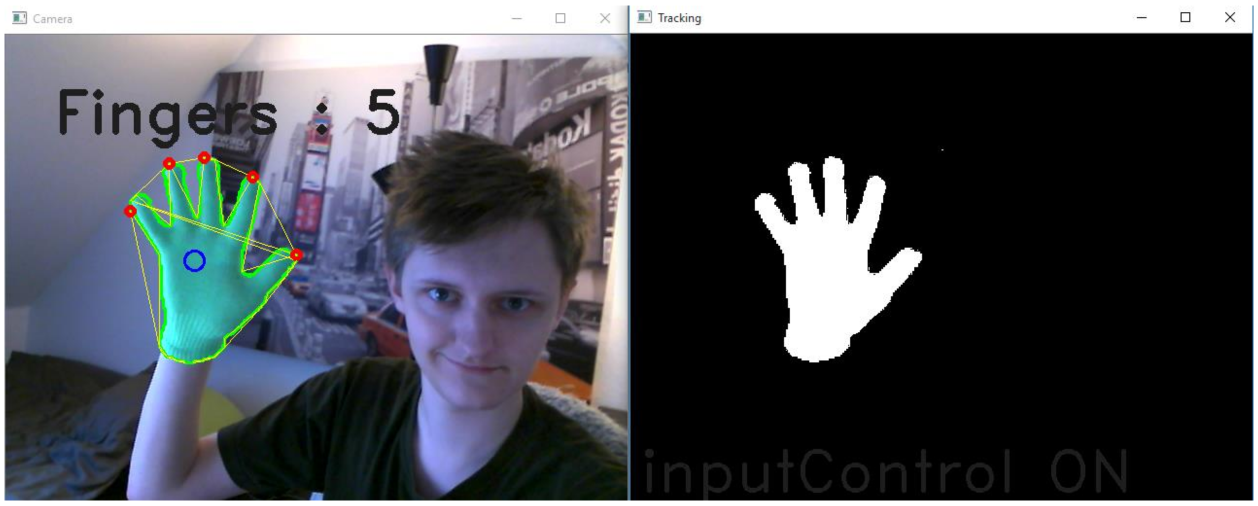


Figure 40 - screenshot of the two windows displaying the camera feed and the tracking window.

6.2 Image processing code

The code takes advantage of multiple functions calling each other, applying the algorithms and drawing as asked. This is controlled by using key bindings that loads functions or changes windows. The software applies threshold, contours, hulls, defects and simulates mouse and keyboard presses. How all of this is achieved will be explained further as the code is reviewed.

Figure 41, this is the start of the main() function which is the first to run. It starts off setting the different Mat variables we are working with, which is the camera inputs and the edits of it. frame is the camera, hsvFrame is the frame converted to the colour space HSV. rangeFrame is the camera input we apply the threshold to, to find the green glove. This is also used to calculate contours, hull and defects. trackFrame is used for positioning the mouse appropriately using the position of the hand on the camera. It starts the camera unless it cannot be found or refuse to start. If it does not start it will shut down after 5 seconds. Otherwise the function will continue.

```
int main(void) {
    Mat frame, hsvFrame, rangeFrame, trackFrame;
    int key;
    VideoCapture cap(0);
    if (!cap.isOpened()) { // Just for the fun of it.
        cout << " ERROR! No camera found at the selected port or the camera is not
functional!" << endl << endl;
        waitKey(500), cout << " The program will shut down in 5";
        waitKey(250), cout << ".", waitKey(250), cout << ".", waitKey(250), cout << ".",
waitKey(250), cout << "4";
        waitKey(250), cout << ".", waitKey(250), cout << ".", waitKey(250), cout << ".",
waitKey(250), cout << "3";
        waitKey(250), cout << ".", waitKey(250), cout << ".", waitKey(250), cout << ".",
waitKey(250), cout << "2";
        waitKey(250), cout << ".", waitKey(250), cout << ".", waitKey(250), cout << ".",
waitKey(250), cout << "1";
        waitKey(250), cout << " THE BUILD PEACED OUT!";
        return -1;
    }
```

Figure 41 – main() function start after initialising libraries and variables. Checks if camera works.

Figure 42, this part starts recording the video to frame, mirrors the camera, converts frame to HSV and applies threshold to the converted image and saves them in rangeFrame and trackFrame.

```
    trackbar();
    trackbarContent(0, 0);

    while ((key = waitKey(30)) != 27) {
        keyBindings(key);
        cap >> frame;
        flip(frame, frame, 180);
        cvtColor(frame, hsvFrame, COLOR_BGR2HSV);

        inRange(hsvFrame, Scalar(minHue, minSaturation, minValue), Scalar(maxhHue,
maxSaturation, maxValue), rangeFrame);
        inRange(hsvFrame, Scalar(minHue, minSaturation, minValue), Scalar(maxhHue,
maxSaturation, maxValue), trackFrame);
```

Figure 42 – main() function continued. Loads the trackbars and starts reading the camera.

Figure 43, this loads the functions that are used throughout the code to apply the different algorithms, contours and tracking. If a key is pressed with 'M' as an example, it will apply morphological opening and closing to the rangeFrame. If 'R' is pressed, it will switch the view between the camera feed and the thresholded video feed. This is the end of the main() function.

```
        if (controlSwitch) {
            inputControl(trackFrame);
        }

        if (morph) {
            morphologicalOpening(rangeFrame, trackFrame);
            morphologicalClosing(rangeFrame, trackFrame);
        }

        if (blurr) {
            blurTreshold(rangeFrame, trackFrame);
        }

        if (showContours) {
            showImgContours(rangeFrame, frame);
        }

        if (trackObject) {
            tracking(trackFrame);
            if (outDrawArea > 10000) {
                mouseMove(lastX, lastY);
            }
        }

        if (switchView) {
            imshow("Camera", frame);
            imshow("Tracking", trackFrame);
        }

        else {
            imshow("Camera", rangeFrame);
            imshow("Tracking", trackFrame);
        }
    }
}
```

Figure 43 – main() function continued. Starts the different functions and changes shown footage depending on key press.

Figure 44, this short function makes sure that ellipse used in erode, dilate and blur is not below 0.

```
void trackbarContent(int, void*) {
    if (sizeErode == 0) {
        sizeErode = 1;
    }

    if (sizeDilate == 0) {
        sizeDilate = 1;
    }

    if (sizeBlur == 0) {
        sizeBlur = 1;
    }
}
```

Figure 44 – trackbarContent() function that makes sure the ellipse used for morphological algorithms is not below 0.

Figure 45, this function starts up a new window called "Trackbars" and enables several track bars inside the window. All these are used to control the threshold of the image applied on rangeFrame and trackFrame. Then there is also the size of the two ellipses used to erode and dilate the image of rangeFrame. Dragging the bars from zero all the way up to the max defined in the start of the code. The size of blur and the intensity of the threshold.

```
void trackbar() {
    String trackbarWindowName = "Trackbars";
    namedWindow(trackbarWindowName, CV_WINDOW_AUTOSIZE);
    resizeWindow("Trackbars", 300, 450);
    createTrackbar("Hue min", trackbarWindowName, &minHue, maxhHue, trackbarContent);
    createTrackbar("Hue max", trackbarWindowName, &maxhHue, maxhHue, trackbarContent);
    createTrackbar("Saturation min", trackbarWindowName, &minSaturation, maxSaturation,
trackbarContent);
    createTrackbar("Saturation max", trackbarWindowName, &maxSaturation, maxSaturation,
trackbarContent);
    createTrackbar("Value min", trackbarWindowName, &minValue, maxValue, trackbarContent);
    createTrackbar("Value max", trackbarWindowName, &maxValue, maxValue, trackbarContent);
    createTrackbar("Erode", trackbarWindowName, &sizeErode, 31, trackbarContent);
    createTrackbar("Dilate", trackbarWindowName, &sizeDilate, 31, trackbarContent);
    createTrackbar("Blur", trackbarWindowName, &sizeBlur, 255, trackbarContent);
    createTrackbar("Threshold", trackbarWindowName, &thresholdValue, 255, trackbarContent);
}
```

Figure 45 – trackbar() function that loads up a window with track bars to control the threshold.

Figure 46, this is where the morphological opening and closing is applied using the size of the given ellipse from the trackbar() function. Blur can also be applied to the image using the normalized box filter and binary threshold.

```
void morphologicalOpening(Mat &source, Mat &track) {
    erode(source, source, getStructuringElement(MORPH_ELLIPSE, Size(sizeErode, sizeErode)));
    dilate(source, source, getStructuringElement(MORPH_ELLIPSE, Size(sizeDilate, sizeDilate)));
}

void morphologicalClosing(Mat &source, Mat &track) {
    dilate(source, source, getStructuringElement(MORPH_ELLIPSE, Size(sizeDilate, sizeDilate)));
    erode(source, source, getStructuringElement(MORPH_ELLIPSE, Size(sizeErode, sizeErode)));
}

void blurTreshold(Mat &source, Mat &track) {
    blur(source, source, Size(sizeBlur, sizeBlur), Point(-1, -1), BORDER_DEFAULT);
    threshold(source, source, thresholdValue, 255, THRESH_BINARY);
}
```

Figure 46 - These three functions applies opening, closing and blur with threshold.

Figure 47, the keyBindings() function sets different keys to invert the state of a boolean. This is used earlier in the main() function to control the different calls of functions and what video feed to be displayed in the windows. This is also used to show hull and defects or remove them.

```
void keyBindings(int key) {
    if (key == 'm' || key == 'M') {
        morph = !morph;
    }

    if (key == 'b' || key == 'B') {
        blurr = !blurr;
    }

    if (key == 'r' || key == 'R') {
        switchView = !switchView;
    }

    if (key == 'c' || key == 'C') {
        showContours = !showContours;
    }

    if (key == 'h' || key == 'H') {
        showHull = !showHull;
    }

    if (key == 'd' || key == 'D') {
        showCondefects = !showCondefects;
    }

    if (key == 't' || key == 'T') {
        trackObject = !trackObject;
    }

    if (key == 'i' || key == 'I') {
        controlSwitch = !controlSwitch;
    }
}
```

Figure 47 – keyBindings() function that applies different keystrokes to change the value of booleans.

Figure 48, this part of the code finds the contours of the hand using findContours(). During this function, a hierarchy is applied to the drawing of the contour. This hierarchy functions to only take the largest one found in the image, which in optimal cases is the hand. When the size of the found contours is above zero, which means when an object is present it will draw the contour of the largest visual object. Hull and defect will also be drawn independently if prompted to through the key presses ‘H’ or ‘D’. These are also limited to a maximum of one that is the largest. The defect prompt starts another function called convexDefects(), which handles the defects.

```
void showImgContours(Mat &threshImg, Mat &source) {
    vector<vector<Point> > contours;
    vector<Vec4i> hierarchy;
    int largestArea = 0;
    int largestContourIndex = 0;

    findContours(threshImg, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);

    vector<vector<Point> >hull(contours.size());
    vector<vector<int> >inthull(contours.size());
    vector<vector<Vec4i> >defects(contours.size());
    for (int i = 0; i < contours.size(); i++) {
        convexHull(Mat(contours[i]), hull[i], false);
        convexHull(Mat(contours[i]), inthull[i], false);
        if (inthull[i].size()>3) {
            convexityDefects(contours[i], inthull[i], defects[i]);
        }
    }

    for (int i = 0; i< contours.size(); i++) {
        double a = contourArea(contours[i], false);
        if (a>largestArea) {
            largestArea = a;
            largestContourIndex = i;
        }
    }

    if (contours.size() > 0) {
        drawContours(source, contours, largestContourIndex, CV_RGB(0, 255, 0), 2, 8, hierar-
chy);
        //drawContours(source, contours, -1, CV_RGB(0, 255, 0), 2, 8, hierarchy);
        if (showHull) {
            drawContours(source, hull, largestContourIndex, CV_RGB(0, 0, 255), 2, 8, hierarchy);
        }

        //drawContours(source, hull, -1, CV_RGB(0, 255, 0), 2, 8, hierarchy);
        if (showCondefects) {
            convexDefects(defects[largestContourIndex], contours[largestContourIndex], source);
        }
    }
}
```

Figure 48 – showImgContours() function which finds the contours and applies a hierarchy.

Figure 49, convexDefects() is called from showImgContours() to display and count the amount of defects inside the hull. It draws small red circles where a defect is present and draws a line from the start of it to the end. How long the defects are determines whether it is counted as a finger or not. If it is above the depth of ten, it will draw its start circle and add to the amount of fingers displayed. If it somehow detects more defects than fingers, it will be forced to stay at five or below. It displays the amount of fingers present through text on the camera window.

```
void convexDefects(vector<Vec4i> convexityDefectsSet, vector<Point> contour, Mat &source) {
    Point2f center;
    float radian;
    int fingers = 0;
    minEnclosingCircle(contour, center, radian);
    circle(source, center, 10, CV_RGB(0, 0, 255), 2, 8);

    for (int cDefIt = 0; cDefIt < convexityDefectsSet.size(); cDefIt++) {
        int startIndex = convexityDefectsSet[cDefIt].val[0]; Point ptStart(contour[startIn-
dex]);
        int endIndex = convexityDefectsSet[cDefIt].val[1]; Point ptEnd(contour[endIndex]);
        int farIndex = convexityDefectsSet[cDefIt].val[2]; Point ptFar(contour[farIndex]);
        double depth = static_cast<double>(convexityDefectsSet[cDefIt].val[3]) / 256;

        if (depth>10 && ptStart.y<center.y) {
            circle(source, ptStart, 4, CV_RGB(255, 0, 0), 4);
            line(source, ptStart, ptEnd, Scalar(0, 255, 255), 1);
            line(source, ptStart, ptFar, Scalar(0, 255, 255), 1);
            line(source, ptFar, ptEnd, Scalar(0, 255, 255), 1);
            fingers++;
        }

        if (fingers >= 5) {
            fingers = 5;
        }
    }
    putText(source, "Fingers : " + std::to_string(fingers), Point(50, 100), 2, 2, CV_RGB(30,
30, 30), 4, 8);
    lastFingerCount = fingers;
}
```

Figure 49 – convexDefects() function which finds, draws and counts the amount of defects.

Figure 50, tracking() function is used to get the position of the hand. This is done by calculating the hands area and getting its approximate centre. This is done by using something known as Greens theorem.

```
void tracking(Mat &track) {
    Moments moment = moments(track);
    double drawM01 = moment.m01;
    double drawM10 = moment.m10;
    double drawArea = moment.m00;

    if (drawArea > 10000) {
        int posX = (drawM10 / drawArea);
        int posY = (drawM01 / drawArea);
        lastX = (posX - 100)*4.2;
        lastY = (posY - 100)*3.3;
        outDrawArea = drawArea;
        circle(track, Point(posX, posY), 20, CV_RGB(255, 0, 0), 7);
        putText(track, "X: " + std::to_string(lastX) + " Y: " + std::to_string(lastY),
Point(10, 300), 2, 2, CV_RGB(255, 255, 255), 2, 1);
    }
}
```

Figure 50 – tracking() function is used to find the position of the hand in trackFrame.

Figure 51, these functions simulates mouse movement, mouse click, mouse release and key press. mouseMove() function sets the x and y axis of the mouse to the centre of the hand. leftClicked() simulates the down press of the left mouse button while leftReleased() simulates the release of the left mouse button. startClick() simulates a key press of a keyboard and in this case it is 0x50 or better known as ‘P’.

```
void mouseMove(int x, int y) {
    double fScreenWidth = ::GetSystemMetrics(SM_CXSCREEN) - 1;
    double fScreenHeight = ::GetSystemMetrics(SM_CYSCREEN) - 1;
    double fx = x*(65535.0f / fScreenWidth);
    double fy = y*(65535.0f / fScreenHeight);
    INPUT  Input = { 0 };
    Input.type = INPUT_MOUSE;
    Input.mi.dwFlags = MOUSEEVENTF_MOVE | MOUSEEVENTF_ABSOLUTE;
    Input.mi.dx = fx;
    Input.mi.dy = fy;
    ::SendInput(1, &Input, sizeof(INPUT));
}

void leftClicked() {
    INPUT Input = { 0 };
    Input.type = INPUT_MOUSE;
    Input.mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
    ::SendInput(1, &Input, sizeof(INPUT));
}

void leftReleased() {
    INPUT Input = { 0 };
    Input.type = INPUT_MOUSE;
    Input.mi.dwFlags = MOUSEEVENTF_LEFTUP;
    ::SendInput(1, &Input, sizeof(INPUT));
}

void startClick() {
    INPUT input;
    WORD vkey = 0x50;
    input.type = INPUT_KEYBOARD;
    input.ki.wScan = MapVirtualKey(vkey, MAPVK_VK_TO_VSC);
    input.ki.time = 0;
    input.ki.dwExtraInfo = 0;
    input.ki.wVk = vkey;
    input.ki.dwFlags = 0;
    SendInput(1, &input, sizeof(INPUT));

    input.ki.dwFlags = KEYEVENTF_KEYUP;
    SendInput(1, &input, sizeof(INPUT));
}
```

Figure 51 - four functions that simulate mouse movement, click, release and keyboard key press.

Figure 52, this part of the function starts each simulation according to the amount of fingers present. If you move your fingers slightly down from a completely stretched hand as you were to grab a handball, a left mouse click will be simulated. It then waits to release the button until the hand is stretched out completely once again. If the hand is closed it will simulate the press of a key ‘P’ and will wait for the hand to be stretched out once again before ‘P’ can be simulated again.

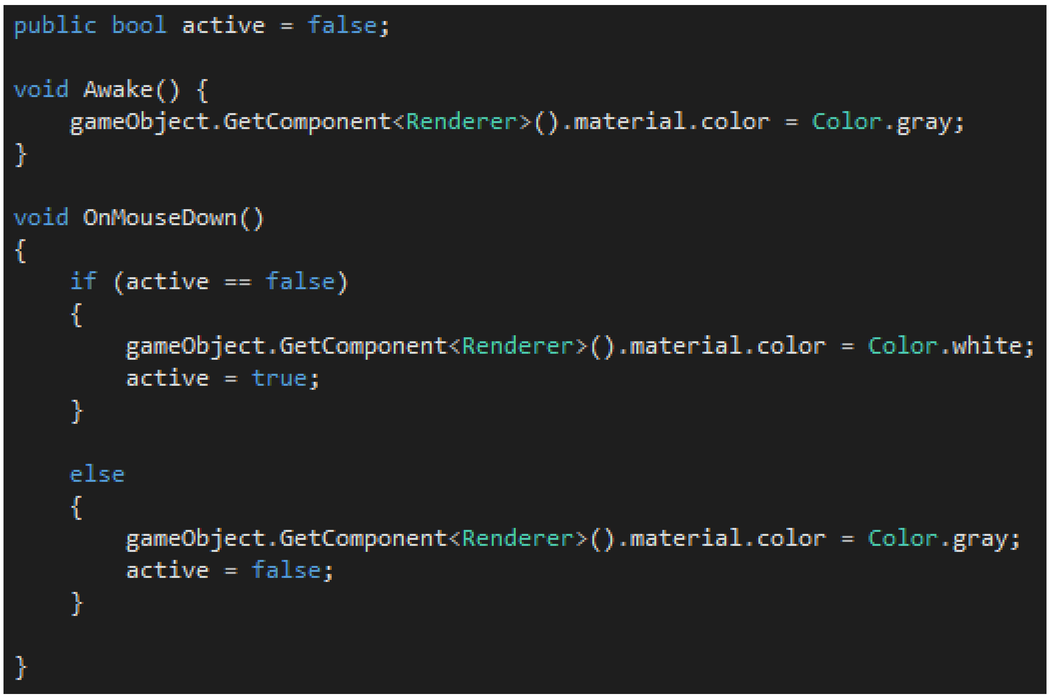
```
void inputControl(Mat &track) {
    putText(track, "inputControl ON", Point(10, 470), 2, 2, CV_RGB(120, 130, 30), 2, 2);

    if (lastFingerCount >= 1 && lastFingerCount <= 3 && leftHit == false) {
        leftClicked();
        cout << "LEFT CLICKED" << endl;
        leftHit = true;
    }

    if (lastFingerCount >= 4 && lastFingerCount <= 5 && leftHit == true) {
        leftReleased();
        cout << "LEFT RELEASED" << endl;
        leftHit = false;
    }

    if (lastFingerCount == 0 && startHit == false) {
        startClick();
        cout << "START CLICKED" << endl;
        startHit = true;
    }

    if (lastFingerCount >= 4 && lastFingerCount <= 5 && startHit == true) {
        cout << "START RELEASED" << endl;
        startHit = false;
    }
}
```

Figure 52 – inputControl() function which applies the simulations according to the amount of fingers displayed.

6.3 Unity code

In this Unity [26] project, we try to re-create a musical “sandbox” (playground if you will), inspired by ToneMatrix. As can be found in the conceptual model (chapter 5.1), we need at least a set of press-able buttons arranged in a grid. Each button in a column should play a different note, and each column should represent one beat. This means that each column should be identical, except for playing their tones at different beats along the timeline.

In order to create this, we arranged a grid of cubes with box colliders. Each of the buttons has a simple C# script applied to them called “Pressed” (figure 53), that is used for changing colors of the buttons and toggling them on/off.



```
public bool active = false;

void Awake() {
    gameObject.GetComponent<Renderer>().material.color = Color.gray;
}

void OnMouseDown()
{
    if (active == false)
    {
        gameObject.GetComponent<Renderer>().material.color = Color.white;
        active = true;
    }

    else
    {
        gameObject.GetComponent<Renderer>().material.color = Color.gray;
        active = false;
    }

}
```

Figure 53 - pressed script.

In the Pressed script, we start out by declaring a single Boolean variable “active”. This variable is going to track whether or not the button is currently active. This variable is set to false by default, as all buttons are inactive on program start.

After this we see the function Awake(). Awake() is called whenever the script is being set to active, in this case when the program starts. In Awake(), we set the color of the button to gray as it is being called. This will be our color symbolizing that a button is not active, meaning that all buttons will appear not active as the program starts.

After this, we see the OnMouseDown() function. This is another native function of Unity, that checks if the left mouse button is currently pressed down, while the mouse is within the collider of the object with the script attached. If it is, it runs the function. Inside the function we have a simple if/else statement. If the variable active is set to false, the button will change color to white (the active color), and set the active variable to true. Else, it will set the color of the button to gray, and the active variable to false.

Beyond the buttons that make up the grid, we also have a start button, that acts both as the start button, and as the actual music player. The start button object has a more complicated script attached to it, called “StartButton”. As with the Pressed script, we start this script out by declaring all of our variables. Beyond the variables we are going to use inside the script, here we also need references to each of the buttons in the grid, so that we may access their active variable.

In Unity, you can extract any component of another object in the program with the GetComponent<> method and we use this to access the 5 different audio clips, as well as the script for each of the buttons in our grid.

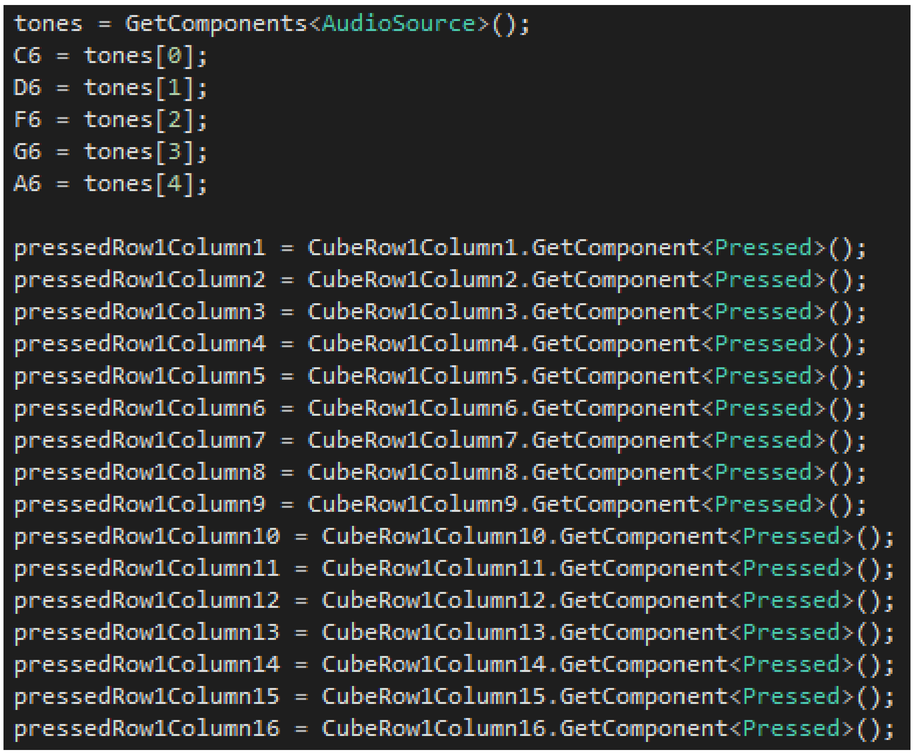

```
tones = GetComponents<AudioSource>();
C6 = tones[0];
D6 = tones[1];
F6 = tones[2];
G6 = tones[3];
A6 = tones[4];

pressedRow1Column1 = CubeRow1Column1.GetComponent<Pressed>();
pressedRow1Column2 = CubeRow1Column2.GetComponent<Pressed>();
pressedRow1Column3 = CubeRow1Column3.GetComponent<Pressed>();
pressedRow1Column4 = CubeRow1Column4.GetComponent<Pressed>();
pressedRow1Column5 = CubeRow1Column5.GetComponent<Pressed>();
pressedRow1Column6 = CubeRow1Column6.GetComponent<Pressed>();
pressedRow1Column7 = CubeRow1Column7.GetComponent<Pressed>();
pressedRow1Column8 = CubeRow1Column8.GetComponent<Pressed>();
pressedRow1Column9 = CubeRow1Column9.GetComponent<Pressed>();
pressedRow1Column10 = CubeRow1Column10.GetComponent<Pressed>();
pressedRow1Column11 = CubeRow1Column11.GetComponent<Pressed>();
pressedRow1Column12 = CubeRow1Column12.GetComponent<Pressed>();
pressedRow1Column13 = CubeRow1Column13.GetComponent<Pressed>();
pressedRow1Column14 = CubeRow1Column14.GetComponent<Pressed>();
pressedRow1Column15 = CubeRow1Column15.GetComponent<Pressed>();
pressedRow1Column16 = CubeRow1Column16.GetComponent<Pressed>();
```

Figure 54 - components.

We now have access the audio clips and we know whether or not each button is active. To emulate ToneMatrix's music player, we use the InvokeRepeating() method. InvokeRepeating() takes 3 parameters: which function it should set to repeat, the initial delay, and the repeat timer. We make an InvokeRepeating() call for each column, as we want the tones in each column to play simultaneously.

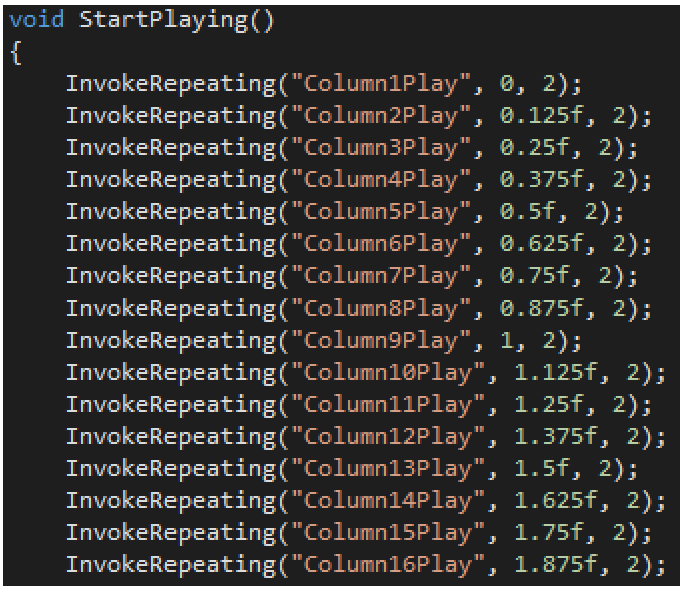
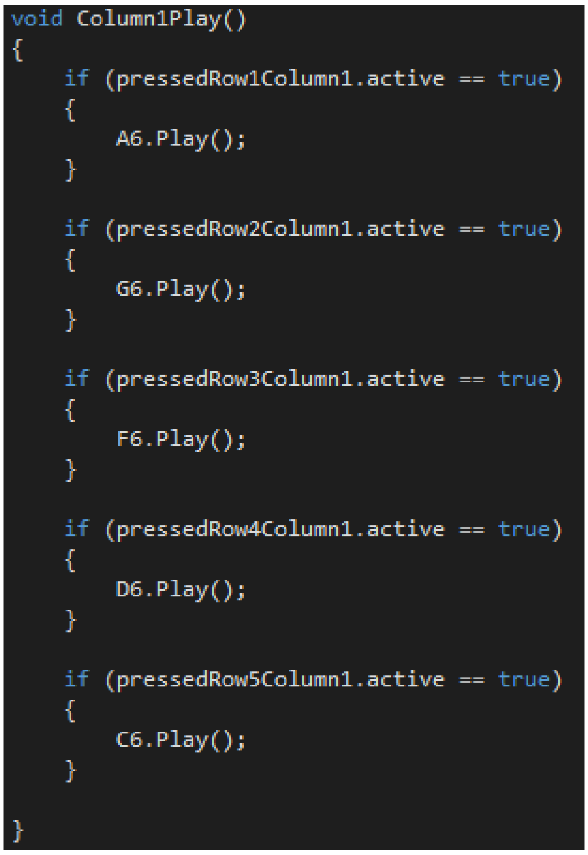

```
void StartPlaying()
{
    InvokeRepeating("Column1Play", 0, 2);
    InvokeRepeating("Column2Play", 0.125f, 2);
    InvokeRepeating("Column3Play", 0.25f, 2);
    InvokeRepeating("Column4Play", 0.375f, 2);
    InvokeRepeating("Column5Play", 0.5f, 2);
    InvokeRepeating("Column6Play", 0.625f, 2);
    InvokeRepeating("Column7Play", 0.75f, 2);
    InvokeRepeating("Column8Play", 0.875f, 2);
    InvokeRepeating("Column9Play", 1, 2);
    InvokeRepeating("Column10Play", 1.125f, 2);
    InvokeRepeating("Column11Play", 1.25f, 2);
    InvokeRepeating("Column12Play", 1.375f, 2);
    InvokeRepeating("Column13Play", 1.5f, 2);
    InvokeRepeating("Column14Play", 1.625f, 2);
    InvokeRepeating("Column15Play", 1.75f, 2);
    InvokeRepeating("Column16Play", 1.875f, 2);
```

Figure 55 - StartPlaying() function.

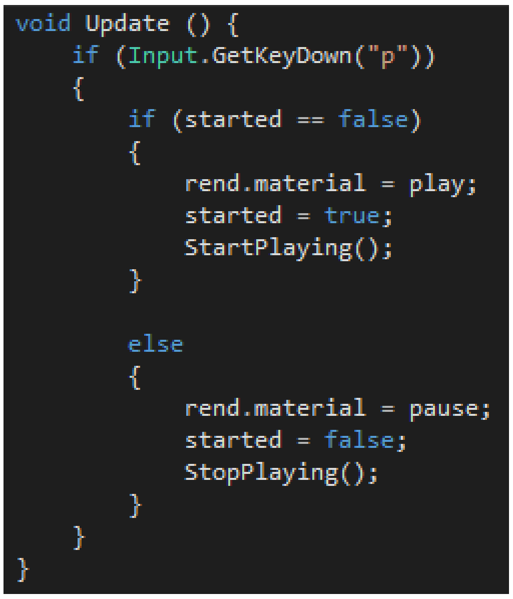
Since we want to play the tones with 8 beats per second, we set the initial delay of the first InvokeRepeating() call to 0 (plays on start), the second to 0.125 seconds, the third to 0.25 seconds, etc. We set the repeat time of each call to 2 seconds, as we have 16 beats with 8 beats per second, giving us a cycle time of 2 seconds.



```
void Column1Play()
{
    if (pressedRow1Column1.active == true)
    {
        A6.Play();
    }

    if (pressedRow2Column1.active == true)
    {
        G6.Play();
    }

    if (pressedRow3Column1.active == true)
    {
        F6.Play();
    }

    if (pressedRow4Column1.active == true)
    {
        D6.Play();
    }

    if (pressedRow5Column1.active == true)
    {
        C6.Play();
    }

}
```

Figure 56 - Column1Play() function.

In each of the ColumnXPlay() functions, we check whether or not each button in that column is active by accessing the script for those buttons, and checking the active variable. If the button is active, it will play its corresponding tone (C, D, F, G, A, bottom to top). This gives a very close approximation to ToneMatrix's experience.



```
void Update () {
    if (Input.GetKeyDown("p"))
    {
        if (started == false)
        {
            rend.material = play;
            started = true;
            StartPlaying();
        }

        else
        {
            rend.material = pause;
            started = false;
            StopPlaying();
        }
    }
}
```

Figure 57- Update() function.

In our Update() function that is being run every frame, we check if the user gives the input to start playing the music. In our image processing program, we have coded our play/pause gesture to simulate a keypress of "p". If p is pressed, we use the same if/else statement we used in our pressed script, with the only exception that it also calls either the StartPlaying() or the StopPlaying() functions respectively. Also, instead of changing colors of the button, it changes the material from a pause icon to a play icon, and vice versa.

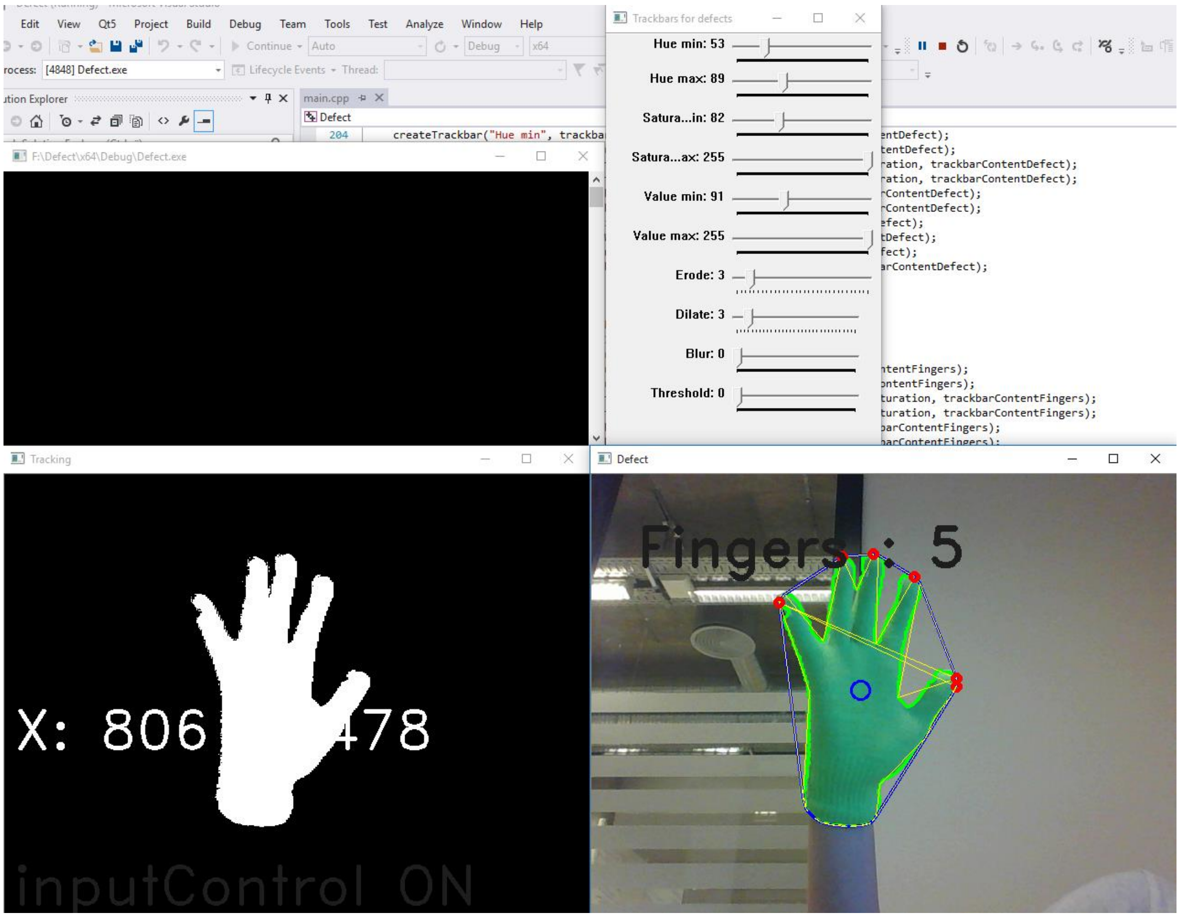
6.4 Pre-Evaluation Implementation

Consist of two programs: OpenCV hand recognition and the Unity music player

The build of the solution used for the evaluation consist of two separate programs. The first program is the hand recognition software written in C++ (Using OpenCV libraries), that can count fingers and use this to issue different commands. The second program is a simple music creation program implemented in Unity, heavily inspired by ToneMatrix [22].

Hand recognition software

In the hand recognition program, a threshold is first applied to the camera feed, in order to subtract the background of the image, leaving only the glove covering the users hand. Once the values for the threshold has been set (using the track bars as seen in the top right corner of figure 58, a silhouette of the gloved hand will be visible (bottom left corner of figure 58).


Trackbars for defects

Hue min: 53
Hue max: 89
Satura...in: 82
Satura...ax: 255
Value min: 91
Value max: 255
Erode: 3
Dilate: 3
Blur: 0
Threshold: 0

```
204    createTrackbar("Hue min", trackba
```

```
entDefect);
tentDefect);
ration, trackbarContentDefect);
ration, trackbarContentDefect);
rContentDefect);
rContentDefect);
efect);
tDefect);
fect);
arContentDefect);



ntentFingers);
ontentFingers);
turation, trackbarContentFingers);
turation, trackbarContentFingers);
barContentFingers);
barContentFingers);
```

Edit View Qt5 Project Build Debug Team Tools Test Analyze Window Help
Continue
Auto
Debug
x64
rocess: [4848] Defect.exe
Lifecycle Events
Thread:
tion Explorer
main.cpp
Defect
F:\Defect\x64\Debug\Defect.exe
Tracking
X: 806
478
inputControl ON
Defect
Fingers : 5

Figure 58 - screenshot of the input control system.

Once the silhouette has been extracted, the software will find the convex hull of the hand (see chapter 6.1), and use this to count the amount of fingers currently visible to the camera (bottom right corner of figure 58). Based on this, commands are issued in the form of mouse clicks and key presses (e.g. a mouse click when three to four fingers are visible). Every time a command is called, mouse clicks and key presses, this is printed to the debug window (top left corner in figure 58). The commands issued through the hand recognition software, is later used as input in the unity program. Furthermore, the movement of the hand controls the movement of the cursor on the screen, giving the user the ability to control the application only by moving their hands.

Music creation software

In the music creation program, implemented in Unity, the user can activate and deactivate buttons placed in a 16x5 grid by clicking on them. The command used for activating and deactivating buttons is called when the hand recognition software counts 3-4 fingers, and simulates a left click of the mouse. If the user is hovering over a button whilst issuing the left click command, that button will set its state to either active or inactive. Inactive buttons appear as a darker shade of grey than active buttons do, as can be seen in figure 59.

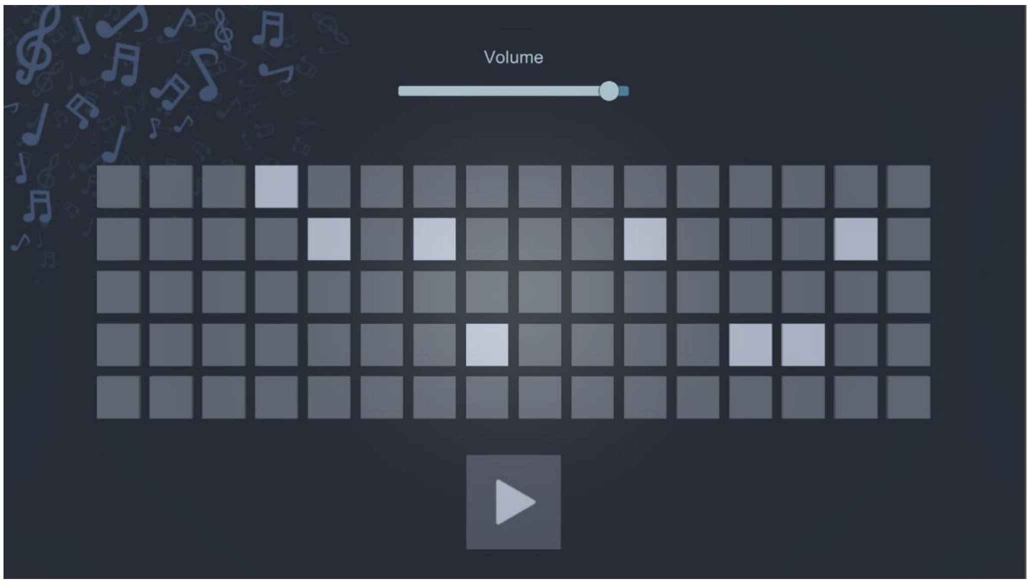


Figure 59 - screenshot of the music system.

Beyond the buttons, there is also a volume control slider bar that the user can manipulate. If the user issues the left click command while hovering over the control handle, and without "releasing" the command, they can adjust the volume by moving their hand left or right. They can release the handle again by making returning to five visible fingers.

The user can also set the music program to a "play" or "pause" state, by making a gesture that has no visible fingers. The same gesture is used for both play and pause. Each time the program is paused and then resumed, the player will start from the beginning of the grid again.

Using the hand recognition and music creation software together

Once both programs are running, the solution can be evaluated by having a webcam (connected to the computer) face the user wearing the green glove. Once everything in the hand recognition software has been calibrated, the solution should be ready for testing.

7. Evaluation

7.1 Materials/apparatus

For the evaluation the group used four group members to complete the testing. The Software tester used one laptop, that was connected to the TV-screen which showed the music software, and another laptop to use the software test excel sheet to see if the program was precise and where there were problems under the evaluation. The camera was connected to the first laptop that tracked the hand. The lead evaluator had his script in front of him and two assignments that he handed to the test person to see if they could complete them in the music creator. The group then had an interviewer who used a laptop to write down answers and notes from the testing. At last there was a group member that used a camera to capture the testing of the program and the interviews for transcripts, if needed.

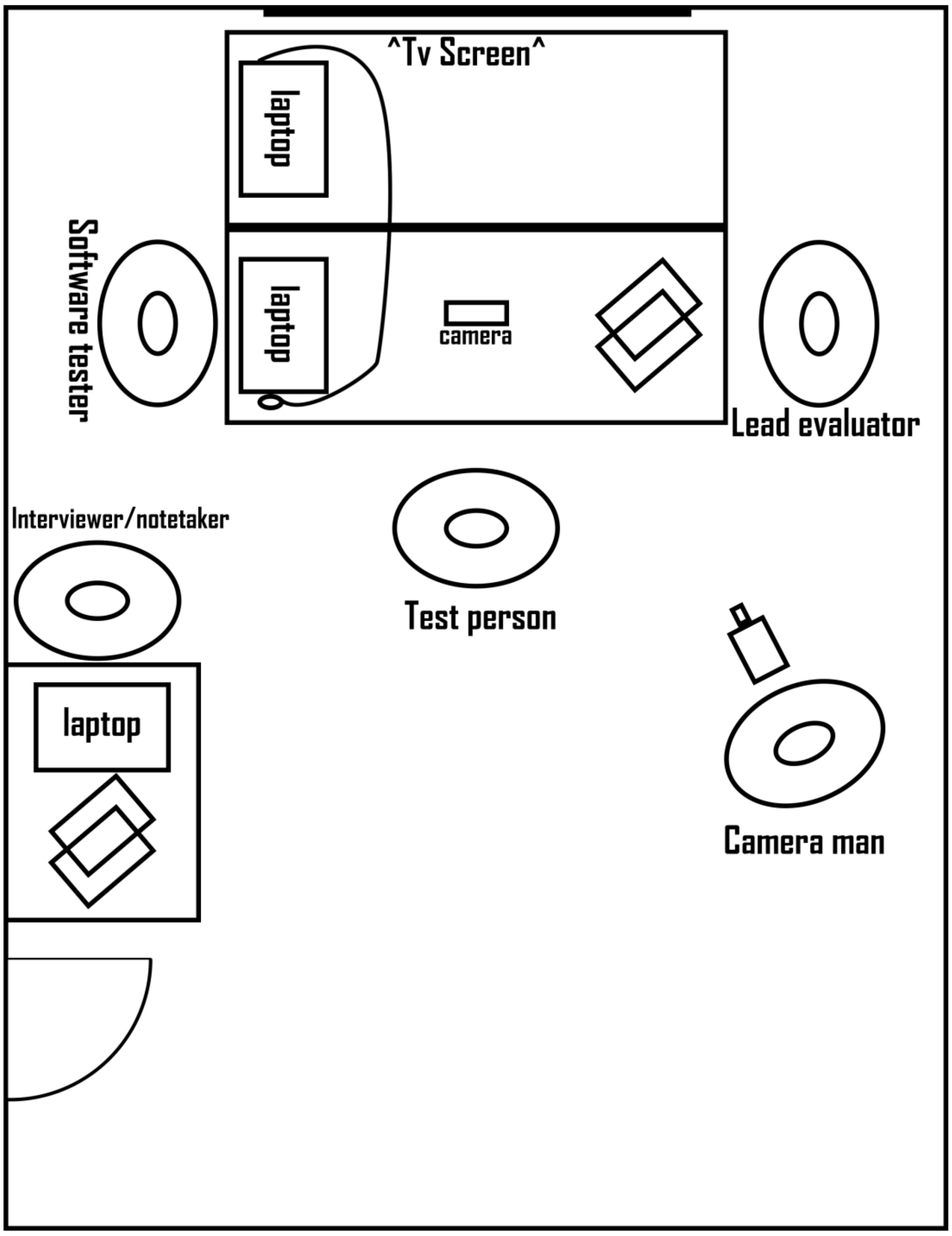


Figure 60 - sketch of the evaluation setup.

7.2 Procedure

This evaluation is tested on one participant at a time. When a participant arrives the first thing to do is to sign a consent form, to allow for the filmed data to be used. This is done at the table next to the entrance of the room. After that the participant stands in front of the camera, facing the screen. He is given a green glove and asked to show a different number of gestures to make sure the calibrations of the hand recognition software is correct. On the screen he is able to see himself, while doing the gestures to get a better view of where the webcam will be able to detect his hand. If the calibrations are in order, the participant is then introduced to the prototype and the three different gestures used in the evaluation. On the screen the webcam feed is then removed and the interface of the prototype is shown in full screen. They are allowed to click on a few buttons to get an idea of how the system works, and then they get their first assignment. Their first assignment is to reproduce a specified pattern on the grid.

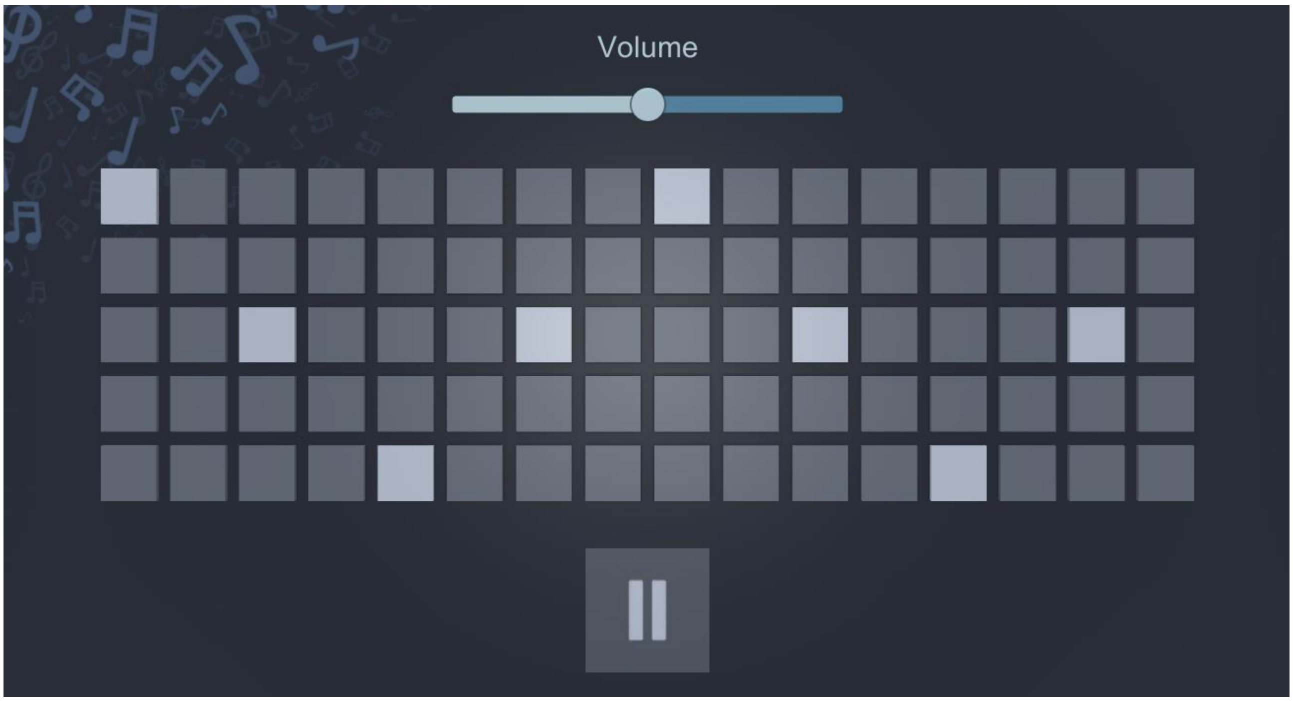


Figure 61 - the first assignment which is to reproduce the pattern on the picture.

This is done to find out how easy/difficult it is to click on specified buttons, and whether or not it is difficult to click on the thing you want to. If the melody is paused they are asked to press the play button to start it again. They are then asked to grab the volume bar, and turn the volume up and down, to find out how well the hand recognition software responds to grabbing. The participants then get another assignment, like the first one.

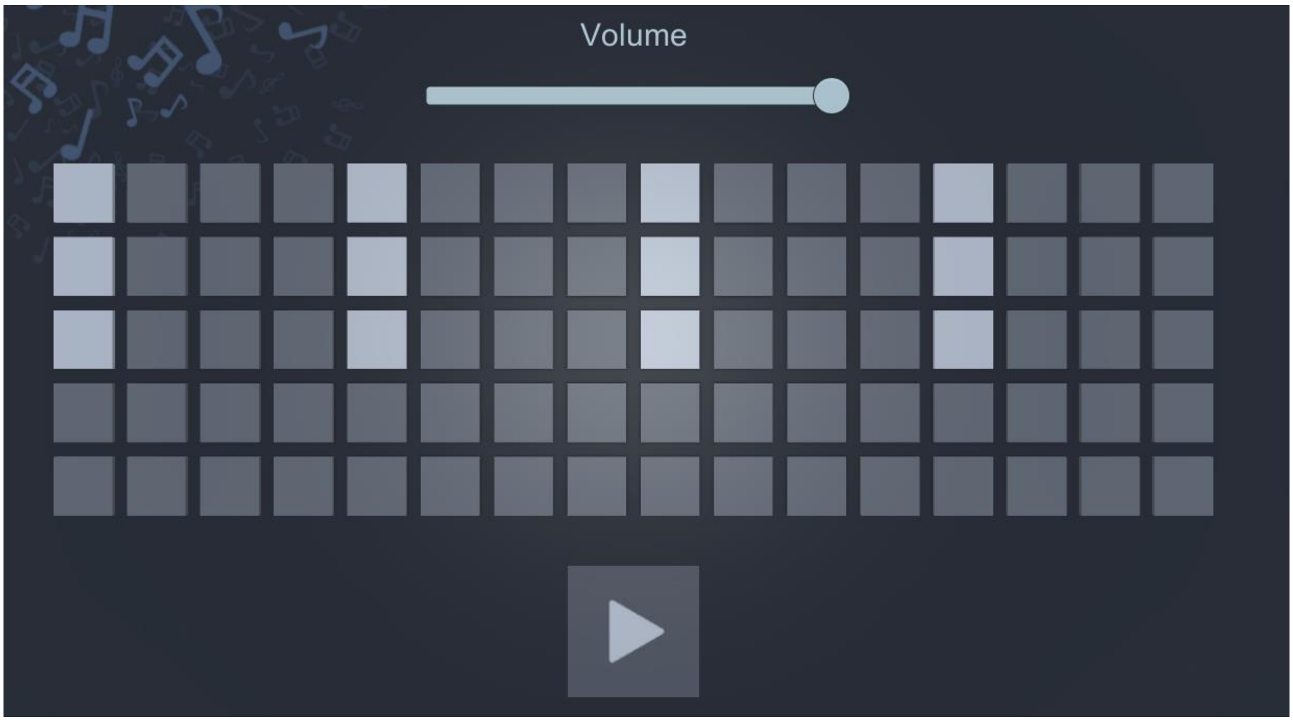


Figure 62 - the second pattern assignment, with the same principle as the first one.

The reason for this pattern is to find out how difficult it is to click on specified buttons, without accidentally turning of other buttons. The participant is then allowed to use the prototype for a minute to make anything he wants, to give him a final idea of how the prototype works. After that the test is over and the participant sits down at a table and are asked evaluation questions.

This evaluation is carried out by following the questions written down in the questionnaire form. The order of this script is kept when asking all participants, this mainly to keep the structure intact and to ensure every participant get through the same approximate process. The answers and comments of the participants are written down in key words and not in direct phrases – it is not the general idea that specific quotes will be needed, but if it happens, the evaluation is recorded to locate the precise phrasing. There are seven strict questions whereof two consist of grading between 1 and 5 for the participant to give. The seven questions are carefully chosen as tools to discuss and validate on the project's success criteria – and to gather as much relevant information from the user in the given time. If needed, the questionnaire can be followed up by supplementary questions regarding topics left unclear or hints given by the participants during the evaluation. These new questions will be written down during the questioning, or quickly afterwards, to keep track of the differences of each evaluation, also which of the participants that were asked what and why that specific supplementary question was important. After going through the questionnaire, the attendees would ask whether the participant had any questions regarding the project and then thank them for participating in the evaluation.

7.3 Participants

The participants of the evaluation were all other Medialogy students from third and fifth semester. The prototype was tested on ten participants. The evaluation was done on other Medialogy students because of their own knowledge and experience with image processing and the project guidelines. It was also convenient since they were in the same building. The distance between the participants and the camera depended on the participant's height, meaning that each participant was standing in a different place, and the image processing software was calibrated depending on their distance.

Eight of the participants were third semester Medialogy students. They provided with feedback based on their own experience with the tools used and because they are using them at the same time as the evaluation took place. The last two participants were fifth semester students, and they had more experience with the tools used, since they have been studying Medialogy and using the tools for a longer time, than the third semester students. The evaluation was done on one participant at a time. Three out of ten participants had previous experience with music or music composing software.

7.4 Data preparation

For data preparation, the group prepared the evaluation tests by making an error-sheet, an interview script and a usability-test, where the evaluation participants could test the current iteration the system and give the group qualitative feedback on the overall design, the functionality and the shortcomings of the prototype, if any were present during the tests.

Preparing the data collection, the group made semi-structured scripts for the interviews, which allowed the interviewee and the interviewer to have a more dynamic conversation about the prototype, rather than just a static experience for both, where the feedback could be lacking in quality.

For analysis of the results, the group recorded all testing sessions with a video camera, and during the interviews, a writer took notes of the replies the interviewee gave to the usability questions during the test. The group went through the interview replies to look for any recurring answers and feedback and compared them to the overall data on the prototype, where an error-sheet was made to check for any functionality errors, such as if the program misbehaved or the user had trouble with the software. The error-sheet had multiple checkboxes, where the group could distinguish between different categories of responsiveness. One for if the program behaved accordingly to the gestures presented, one if it did not, and one if it was inconsistent, where the answer could be ambiguous.

Before the discussion of the results, the group categorized the different responses as achieved or partially achieved in accordance to the success criteria of the project.

7.5 Results

Ten people were a part of the evaluation process during the two days, 8th and 9th of December. General system issues were noted during this process, but there occurred no overall failures or system breakdowns. Though, at one time during the 9th, the process had to take a break during an evaluation to close the program and adjust the image processing settings – this was due to a sudden increase in light and brightness from the windows that particular day. This increased level of light interfered the program and made its tracking of the hand gestures less precise, but – if needed – the evaluation could still had been continued even with the decreased accuracy. Furthermore, all participants completed the given assignments within the program and none seemed to have a difficult time doing it. The following part will be pinpointing the main results from the evaluation questionnaire, which will be listed under their respective questions.

Questionnaire

Did you encounter any problems with the program during your test period?

Eight people had direct problems with the controls, mainly because they felt that some buttons which they did not intend to press was accidently pressed when they moved their hand across the grid. This often resolved in the wrong button being pressed, but sometimes also deactivating a button which was intended to be activated instead. As a remark to this, few participants also experienced the program starting to play/stopped playing when it was not intended, as the cause of the recognition software misreading a gesture. Generally and especially later in the process every participant learned to be more precise with their gestures and were more careful with their hand's movements and location. Although, six of these people also felt it frustrating and/or fatiguing that they had to hold their hand in a specific angle while keeping – what was described as a 'strained' – outstretched hand gesture. While they had to grab and press buttons, most of the time were actually spend using the one of the three gestures that felt most stressful. In addition to this remark, the participants felt that the program required precise movements and that the corner areas of the screen could be difficult to reach. Two out of the ten participants disagreed, and stated that they thought the controls were just something you got used to and that there were no issues with it, except maybe needing a more 'precise' calibration. These two participants were also faster than the other participants in accomplishing the assignments, with one of them being the absolute fastest and with near zero unintended buttons being activated. During the evaluation he also open-mindedly stated that the program was surprisingly smooth and responsive to his movements.

What is your opinion on the program's interface?

It was a general opinion that the design itself was simple and that it was easy to navigate the program. Also the gestures were easy to make after having been briefed by one of the projects attendees. They also agreed on the fact that an easily navigated interface was an important element. The rest of the statements are quite individual, with one participant stating that she experienced difficulties when moving her right (controlling) hand to the left side of the program. Another one thought the design was good and that the color choice was optimal. Two out of the ten participants wanted more information on the different columns and rows. Not enough information was given to the participants from the screen itself. A last one thought that there should be more space between the buttons.

Did you find the program easy to use and understand?

This is a question where the participant answered with a number from 1 to 5, where 1 was the worst and 5 the best. This number could, but didn't have to, be accompanied with some key statements.

Grade	Times given
1	0
2	0
3	0
4	5
5	5

Figure 63 - Shows the different grades and the amount of times that specific grade was given by the participants. There were 10 participants.

General statements:

- It was difficult to see which key tones the different buttons would make
- The pause/play icon seemed confusing, some thought it was a button
- It was fun and easy to complete the assignments

Did you find the program intuitive?

This is a question where the participant answered with a number from 1 to 5, where 1 was the worst and 5 the best. This number could, but did not have to, be accompanied with some key statements.

Grade	Times given
1	0
2	1
3	2
4	3
5	4

Figure 64 - shows the different grades and the amount of times that specific grade was given by the participants. There were 10 participants.

General statements:

- It felt natural to click with your hand they way the program wanted
- There was a well-organized mindset behind the gestures and the way they worked
- It was not natural to move your right hand that much to the left side of the program
- It was interesting to use the gestures and movements, but the angle of the hand seemed unnatural at times

Would you prefer a mouse over the hand gestures?

Five of the participants stated that they would prefer to continue working with the hand gestures, whereof two of these added that they would prefer the mouse if they had to work for a large amount of time or if they had to create music more seriously. The reasons for using the hand gestures were that it was fun, more interactive and innovative. The general thoughts behind shifting to a mouse were that the hand were fatigued from the gestures over time, that the mouse were less 'demanding', less 'glitchy' and more responsive. It was therefore a choice made primarily on the fact that the mouse had a less amount of errors and were more professional.

When asked during a follow-up question, one participant felt that the mouse was the optimal choice, because he was used to it and it would be more effective.

Any suggestions for the program that could improve it?

Four out of ten participants wanted a dedicated button to clear the grid for all activated buttons, reset the grid system to a fresh start. However, more than four were frustrated by the fact they had to clear the grid themselves by clicking every button individually, but these participants did not bring it up later during the questionnaire. Five participants wanted more information on the grid, for example regarding the key tones that were played or a general bar showing the progress of the player. The participants with musical experience often suggested a bigger range of instruments, pitch levels and key tones. Also, a participant said that right now the design was fun and innovative, but that it would need a relevant reason to utilize 'utilize controlling' to move further.

What do you think about creating music in this way and do you think it could be used in a public environment?

All ten participants thought that the concept itself was good and interesting, and two added that it was also very 'childish' and fun and that the concept relied more on this than other programs. One said it could utilize the 'childlike soul' that dwells inside us all, and that people would 'lighten up' and have fun when they used the program. Nine participants seemed to agree that the program could be used in public spaces and environments, with the 10th person actually being a bit unsecure and stating that perhaps it could be done, but that he did not know directly how. The other nine had several of ideas on how it could be done, and concluded that it could be used in bigger social projects or public activities. The ideas were mainly pointing towards being a 'waiting screen' or 'creative board' for people to use in public spaces as airports, railway stations, stations in general, libraries or long passages where a lot of people are walking close by. There were also suggestions of utilizing it in other areas and environment, such as schools with focus on learning, educational purposes, rehabilitation or to increase motor skills. Some of the participants said the concepts were quite like the project Piano Stairs [12]. Another idea mentioned was to look at the physiological state and brain damaged individuals to perhaps use the beats or rhythm also in some sort of process. One participant felt that the sounds of the current program could be quite annoying and that, due to this, it would be better suited for children or bigger 'fun' and social projects on a bigger monitor.

Collected topics from the questionnaire

Some few main topics collected from the walkthrough of the questions and their results above.

- Generate a 'clear' or 'grid reset' function. It could both be a button or a gesture.
- Give more information on the grid and indicate which buttons play what music.
- Expand the musical range with full octave spectrum and the option for more advanced pitching.
- Hinder the users in getting a fatigued hand
- Give the users an advantage, rather than it just being more fun, when using gestures instead of mouse.
- Many participants accidently start/stop the program when they actually want to activate a button
- Many participants accidently activate button when they are moving past them with their hands
- The play/pause icon below the grid is confusing to many participants.
- Do not engage competition with more professional and traditional composing programs, instead use the more 'childlike' and interactive concepts unique to this program.

7.6 Discussion

The success criteria for this project are:

1. Provide an easy-to-use interface
2. Implement a responsive and precise navigation system using Image Processing
3. Recognizing different hand gestures by counting amount of fingers
4. Include several hand gestures
5. The controls of the music creation program should be intuitive and natural
6. Evaluating the prototype in a public area

Achieved Success Criteria

Provide an easy-to-use interface:

The results regarding the interface was quite clear. The participants found it easy and natural to use during the evaluation. The program in general also got just 4's and 5's when the participants had to grade it for how easy and understandable the system was – mainly being due to the simplicity of the grid system, activating buttons and few gestures. If you made the wrong gesture, and activated the wrong button, you could simply remove it again. Even though the participants felt it was an easy interface, they still had some improvements with for example more information and cues on the musical grid.

Partially Achieved Success Criteria

Implement a responsive and precise navigation system using image processing:

The system was responsive and precise as long as the participants stayed in front of the webcam with their palm in the correct angle towards it. As seen in system error-sheets (see table xx) the program worked with the hand gestures assigned to their specific functions and the precision was decent at least, with the gestures nearly always being recognized. However, sometimes the gestures did not respond as was intended, due to the software not counting the correct amount of fingers. One participant was even saying out lout that it was 'surprisingly responsive'. However, this is seen from a technical point of view, from a user-friendly angle the evaluation of the criteria change significantly. The system was precise, but not consistently precise enough, often leading to the image processing part varying in its count of fingers. For example, there was very little distinction between the 'open' and 'clicking' hand, often leading to participants clicking the wrong buttons if they were not careful or precise enough with their movements.

Recognizing different hand gestures by counting the amount of fingers:

The recognizing criteria is only partially achieved, because even though the system is recognizing three different gestures, it is not consistently precise enough to distinguish between 4-5 fingers or 0-1 fingers. It will get it right eventually, but often it will vary or jump suddenly, which could cause frustration for the user if there were five gestures with each one affected by a special amount of fingers. For example, with a closed hand it will still sometimes recognize one finger, and with five fingers it might recognize four, but then go back to five. Right now it works with three gestures, when you choose the right intervals, but it will be difficult to implement new gestures between the first three due to this inconsistency.

Include several hand gestures:

Even though there are three different gestures, they are not very different from each other and the user often 'clicks' when he/she is actually just dragging the open hand over the grid. This happens because the two gestures are not distinctive enough. Furthermore, with the current way the hand gestures are implemented, it will be difficult to implement more hand gestures, without expanding the hand recognition software. As these features are seen as important factors in evaluating, this success criteria cannot be fully passed.

The controls of the music creation program should be intuitive and natural:

The prototype actually gets high grades in its initial intuitive usage, but the majority of the participants ran into issues longer into the evaluation. This either being the many right-handed users having difficulty using the program on the left side – because they still need their palm to stay in the correct angle – or the 'open-hand' gesture being fatiguing to uphold over a longer duration. As long as such issues are common, the system cannot be evaluated as fully intuitive or natural for the users.

Neglected Success Criteria

Evaluate the prototype in a public area:

The system was not evaluated in a public area. Even though it was conducted in a room at the Create building, this cannot be accepted as 'public'. The original idea – and therefore the criteria – was clearly that such a system should be evaluated in full publicity as this was also one of the main ideas for future research. Could this concept evolve with public environments? One of the questions in the questionnaire hinted that actually nearly all the participants saw this possibility as well, but the criteria remains neglected as time and strict planning resulted in a more isolated evaluation.

Comments concerning Success Criteria

Both the achieved success criteria will be needing either further improvements or some better formulated criteria for future evaluation. Also, the main parts of the system might be set, but the partially achieved criteria clearly demands more specific enhancements with a more developed evaluation in a 'open' and social environment. With all this included, the main regards of the interface and general concept were highly liked by the participants. The evaluation, and its success criteria, can be used as an acknowledging stepping stone for further implementations, studies and development.

7.7 Analysis of method

The evaluation was done in two parts: a test of the system and an interview with questions about the systems and its shortcomings. The feedback from the evaluation was gathered with a qualitative approach with a semi-structured interview with each participant after the test. The success criteria were used as a baseline when making the interview questions, to ensure validity [9] by testing the solution in relation to the context.

The testing part of the evaluation followed a structured script, meaning that all participants followed the same procedure and went through the same tasks. This makes it possible for others to reproduce the evaluation and thereby increasing reliability [9] in the evaluation.

During the interview part the interviewer avoided asking leading questions to make the data more valid. The interviews were semi-structured with a main script and a few follow-up questions based on the answers given. This decreases the reliability, since it makes the interviews more difficult to reproduce, but most of the questions were structured and made beforehand which also increases the reliability. It also increases validity when there is a large amount of structured questions, since the interview is structured to get feedback about the areas we want to investigate.

During the test part of the evaluation one of the group members observed the test with a technical error sheet. The point of the error sheet was to check whether or not the software performed as intended, marking off how well it worked. Some of the things included were whether or not the program recognized the correct number of fingers shown. The error sheet also contained another criteria which checked if the program reacted to the gesture as it was intended.

7.8 Further implementation based on evaluation

According to the results of the evaluation, some users did not find the use of the gestures natural and with the performance result, gestures did sometimes not register properly. With this information, the software controlling the system inputs where further improved taking advantage of contours counting and a second color on the hand. In addition, more features were added to better control the morphological opening and closing applied as well with the normalized box blur and binary threshold. A third window was added to show the detection of the new blue spots on the fingertips of the green glove. The process of this window is similar to the method we use to find contours for the defect. Instead of limiting the amount of contours to the biggest one, it now counts them all. This makes sure that the program can find all five blue finger tips. There is currently no worry about noise interfering with this as the other methods applied to reduce the said noise works almost perfectly. Figure 65 shows the code used to find the contours and add them up to count how many fingers is present.

```
void fingerFrameContours(Mat &fingers, Mat &source) {
    vector<vector<Point> > contours;
    vector<Vec4i> hierarchy;
    int largestArea = 0;
    int largestContourIndex = 0;
    int contourCount = 0;
    findContours(fingers, contours, hierarchy, CV_RETR_LIST, CV_CHAIN_APPROX_NONE);
    vector<vector<Vec4i> >defects(contours.size());
    for (int i = 0; i < contours.size(); i++) {
        contourCount++;
    }

    if (contours.size() > 0) {
        drawContours(source, contours, -1, CV_RGB(0, 255, 0), 2, 8, hierarchy);
    }

    if (contourCount >= 5) {
        contourCount = 5;
    }

    if (contourCount < 1) {
    contourCount = 0;
    }
    contourFingerCount = contourCount;
    putText(source, "Contour fingers: " + std::to_string(contourCount), Point(10, 440),
FONT_HERSHEY_SIMPLEX, double(0.9), CV_RGB(255, 255, 255), 1, CV_AA);
}
```

Figure 65 - fingerFrameContours() function used to count the amount of BLOPS found from the blue finger tips.

The finger count it gets from this is then as mentioned used in combination with the defects. Similar if statements are made in the inputControl() function used earlier but changing how the different clicks are activated based on the critic from the evaluation. This is also more precise as it requires the two finger counters to be almost similar before it initialises the click. Figure 66 shows how the new gestures were implemented and what they each call.

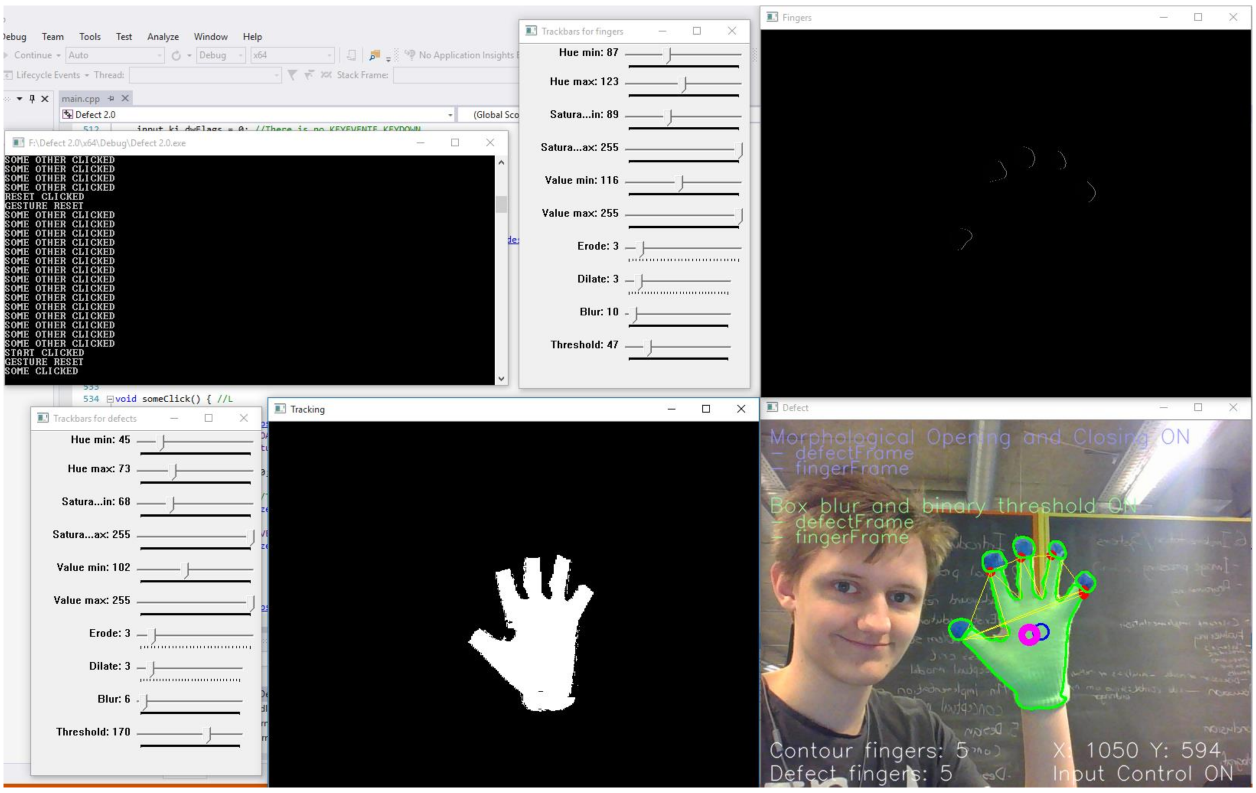
```
void inputControl(Mat &track, Mat &source) { //Letters already in use; I, M, B, D, R, C,
H and T
    putText(source, "Input Control ON", Point(380, 470), FONT_HERSHEY_SIMPLEX,
double(0.9), CV_RGB(255, 255, 255), 1, CV_AA);

    if ((defectFingerCount <= 2 && contourFingerCount <= 0) && (leftHit == true ||
startHit == true || resetHit == true)) {
        cout << "GESTURE RESET" << endl;
        leftHit = false;
        startHit = false;
        resetHit = false;
        someHit = false;
        someOtherHit = false;
        leftRelease();
    }

    if ((defectFingerCount >= 1 && defectFingerCount <= 2) && contourFingerCount == 1 &&
leftHit == false) {
        leftClick();
        cout << "LEFT CLICKED" << endl;
        leftHit = true;
    }

    if ((defectFingerCount >= 1 && contourFingerCount <= 3) && contourFingerCount == 2 &&
startHit == false) {
        startClick();
        cout << "START CLICKED" << endl;
        startHit = true;
    }

    if ((defectFingerCount >= 2 && defectFingerCount <= 4) && contourFingerCount == 3 &&
resetHit == false) {
        resetClick();
        cout << "RESET CLICKED" << endl;
        resetHit = true;
    }

    if ((defectFingerCount >= 3 && defectFingerCount <= 5) && contourFingerCount == 4 &&
someHit == false) {
        someClick();
        cout << "SOME CLICKED" << endl;
        someHit = true;
    }

    if ((defectFingerCount >= 4 && defectFingerCount <= 5) && contourFingerCount == 5 &&
someOtherHit == false) {        someOtherClick();
        cout << "SOME OTHER CLICKED" << endl;
        someOtherHit = true;
    }
}
```

Figure 66 - inputControl() function controls the different gestures used controlling music application or other windows functions.

With the more precise calculation of the amount of fingers, it is easier to implement even more gestures without worrying if they would overlap with each other. So as of right now there is up to six gestures used in the program. Currently only four of them are in use in the music software but there is space for further implementation. The current fully implemented gestures are;

0 finger(s) present:	Closed hand, this is the reset hand gesture and idle gesture.
1 finger(s) present:	Pointing, this is used to click with the left button on the mouse.
2 finger(s) present:	Peace sign, this is used to release the key 'P' to start/pause the music program.
3 finger(s) present:	Rock'n roll, this is used to reset the canvas on the music program with the key 'O'.
4 finger(s) present:	This is currently not used but releases the key 'L' for possible implementation.
5 finger(s) present:	This is not used either but releases they key 'K' for possible implementation.

In figure 67, there is a screenshot of the working program that controls the new inputs and counts the new blobs on the green glove.



```
input.ki.dwFlags = 0; //There is no KEYEVENTF_KEYDOWN
```

F:\Defect 2.0\x64\Debug\Defect 2.0.exe

```
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
RESET CLICKED
GESTURE RESET
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
SOME OTHER CLICKED
START CLICKED
GESTURE RESET
SOME CLICKED
```

```
void someClick() { //L
```

Trackbars for fingers: Hue min: 87; Hue max: 123; Satura...in: 89; Satura...ax: 255; Value min: 116; Value max: 255; Erode: 3; Dilate: 3; Blur: 10; Threshold: 47

Trackbars for defects: Hue min: 45; Hue max: 73; Satura...in: 68; Satura...ax: 255; Value min: 102; Value max: 255; Erode: 3; Dilate: 3; Blur: 6; Threshold: 170

Defect:
Morphological Opening and Closing ON
– defectFrame
– fingerFrame
Box blur and binary threshold ON
– defectFrame
– fingerFrame
Contour fingers: 5
Defect fingers: 5
X: 1050 Y: 594
Input Control ON

Figure 67 - screenshot of the new input control program in use.

8. Discussion

8.1 Quality of the solution

The solution was generally well received during the evaluation, with most of the participants finding it easy to understand and intuitive to use. However, there were some room for improvement in both the hand recognition software, and the music creation software.

Hand recognition software

The hand recognition software performs adequately most of the time. During the evaluation however, only convex/defects were used for counting the amount of fingers visible, at times giving varying results, both in the form of gestures not activating when they should, or activating when they shouldn't. The group has since tried to address this by implementing a secondary method for counting fingers, as discussed in chapter 7.8. This method is used for double checking the finger count when calling commands based off gestures, and has helped achieve a higher accuracy and consistency when detecting gestures. This could still be improved further, but is currently at an acceptable state, that does not negatively affect the experience of using the solution.

The system is not currently capable of detecting a big variety of different gestures, and this would be something that should be considered for future work on the solution. Whilst it does provide enough different gestures for testing purposes (3 was used for the evaluation), it is doubtful whether it would remain adequate for a fully devolved solution. In chapter 8.3, a possible approach to solve this problem is discussed.

Music creation software

The interface of the music creation software was well received by the participants during the evaluation. This would still have to be addressed in future work, along with some of the other suggestions for improving the interface given in chapter 8.3.

Even though it was possible to evaluate the prototype with only one type of instrument implemented, the original goal for this project was to develop a solution that could play several different sounds. If this solution is to be developed further, this is one of the key focus points for the music creation software, as it gives the users even more freedom in terms of possible combinations.

General comments

One major caveat of this project, is that the group did not have time to test the solution in a public area. One of the original motivations for making this project, was to create a solution that could possibly help support social activities. Whilst all of the participants said they thought this solution would function well in a public area, this cannot be used as definitive evidence. This aspect would require to be evaluated, and possibly also investigated further.

8.2 Wider context

From the results of the evaluation, several suggestions were given on how the solution might be applied in other areas than the originally intended ones. Furthermore, some general feedback was given concerning the usage of the solution.

This opens up for new frontiers that could be investigated, with the intention of creating new ways and contexts of use for the solution. Whilst these areas are not looked further into during this report, they are worth mentioning when looking at the wider context of the solution created within this project, and as an inspiration for further research.

One of the ideas that was brought up during the evaluation was the use of the solution in recreational activities. This could be when addressing motor-skills (such as precision and understanding of dimensions) or cognitive abilities, but each of these topics would require further investigations. Specifically, it would require studies that shows how hand recognition software or other image processing software could aid in these recreational activities. Furthermore, it might also be interesting to look into if the process of creating music can somehow aid in cognitive recreational activities but once again, this would require studies that support this claim.

Another idea that was mentioned during the evaluation was the use of the solution in an educational environment. This approach should focus on using the solution as an instrument for teaching children some basic concepts about tones and beats. Since the solution focuses on being easily accessible with intuitive controls, it should correspond well with the idea of using it for teaching basic principles. However, as with the recreational approach, this would require studies that show whether this would be beneficial or not. Nonetheless, it proposes an interesting idea for conducting further research.

Several of the participants from the evaluation mentioned, that the solution should not be trying to compete with professional music creation software. Rather, the focus of the solution should be on making it accessible for everyone in terms of difficulty, as well as make the process of using it engaging and entertaining.

Furthermore, all of the participants from the evaluation mentioned that they were interested in the idea of placing the solution in a public space. Several people suggested using in areas with children, as the simplistic approach of the solution might appeal well to them. This goes well with the previous suggestion of using the solution in educational activities, but would as mentioned require more investigation.

As a last thing, some of the participants also mentioned that it might be interesting to use this solution as a social experiment, with one of the participants drawing a direct reference to Piano Stairs (reference) This aligns well with the initial ideas set by the group, and is definitely something that would merit further investigations. Whilst the prototype was not tested in a public space, it is clear that there is a general interest testing how it would function under such circumstances. It seems there is potential for using the solution as a sort of community mirror that could help support social activities.

8.3 Further implementation

During the evaluation of our project, several suggestions were made to features that could be implemented to further enhance the quality of the solution. Here we are going to delve a bit deeper into the features that we believe would be interesting to have as further implementations, taken from the results of the evaluation, unrealized goals as well as other ideas we have had along the way. At first we are going to look at further implementations, we would like to add to the music player software, and afterwards we are going to look at what further implementations we would like to make to our hand recognition control software.

Music player

During the evaluation, some of the participants asked for a way to track the progress of the player. Several of them suggested implementing a progress bar, which would move along the X-axis to show what beat was currently being played. This would allow users of our solution to better keep track of what tones affected the music in what way, and could potentially be coupled with the play/pause-indicator to make that feature more intuitive as well. This idea is similar to the progress system that can be seen in the original ToneMatrix [22], where each active button emits a wave of light when it plays its tone, and thus giving you a sense of where the repeating function of the music player currently is.

Furthermore, most of the test participants asked for a feature to reset the active buttons. They felt it was too much trouble having to deactivate each active button individually before trying to make a new track. This would also fit well with the idea of placing the solution in a public area, as new participants might want to try and create something of their own, rather than continuing on the works of others.

Generally, it would also be beneficial to implement clearer indicators for the different elements on the screen. Some of the test participants asked for indications of the different tones (keys and octaves). This would allow users who are already familiar with some music theory to take more advantage of the system, by using some of the knowledge they already have. Potentially, it could also allow users without any knowledge of music theory to familiarize themselves better with some of the basic principles of notes and keys. However, this is something that would require further investigation, as we do not currently have any studies showing whether or not they would actually be able to improve their knowledge of music theory.

As mentioned in chapter 5.1, part of the original idea was to implement several different instruments that could be used simultaneously. Due to time constraints, this could not be implemented in the version which was used for evaluation, but would be a key feature to implement if going further with the project. Some of the test participants also uttered that they would like more options in terms of octaves of each tone, and/or controllable pitching. Implementing these steps would however require a greater amount of knowledge of music theory, as it focuses more on the musical part of the program, than the experience of using it. None the less, this would be an interesting feature to pursue.

Hand recognition controls

Whilst the hand recognition controls worked to a fairly decent degree during the evaluation, there was still things that could use improvement. In the build used at the evaluation, the program only used convex hull and defects for identifying fingers. This was then used to make different commands, based on how many fingers were visible to the camera. However, this did not always work as intended, as the program would sometimes find a finger at a point that was a not a finger, giving incorrect results at times.

This could be addressed by implementing a secondary method for recognizing fingers, and double check between the two methods before accepting the status as a finger. This is currently being implemented by assigning a secondary color to all of the fingertips of the green glove that is for recognizing the hand. Along with detecting fingertips based on defects, the program now also looks for blobs of the secondary color in a separate method using blob detection. This allows greater precision when determining how many fingers are currently visible, as there needs to be both a fingertip visible from the defect detection, and a blue blob visible from blob detection, before a finger is counted. Since the blue points on each finger won't be visible for the camera while the finger is bent, this should eliminate most of the errors that occur when counting fingers.

For further implementation regarding this feature, it would be interesting to give each fingertip a separate color, as this could allow for some considerably more complicated hand gestures. Having different colors on each finger would allow the program to recognize specific fingers individually, rather than just identifying them as a finger. Based on this, hand gestures could be programmed that requires one or several specific fingers to be visible. This feature could be implemented using the same method as described with the blob detection above, the difference being that it would have to run the blob detection 5 times instead of 1, to check for each of the five colors.

9. Conclusion

In this project, the group set out to develop an interactive music creation program that can be controlled through the use of image processing. The motivation for creating this system, was the possible flaw in current existing solutions that set a high entry level for new users. In order to address this problem, the group sought inspiration in existing solutions such as ToneMatrix. By creating a simplistic interface, which focuses more on combining different tones to create simple melodies, than the technical aspects of creating complex music. The solution was developed as a two part system: the hand recognition software was implemented in C++ using the OpenCV libraries, whilst the music creation software was implemented in Unity. During the evaluation, the solution was generally well received.

The solution provided an easy to use interface, with no complex features confusing users with little-to-no knowledge of digital music creation. However, several different ideas for new features were suggested during the evaluation that could help make the experience more enjoyable, and provide more combinations in terms of tones and instruments. Overall, the interface could use more information about what is happening where, but this did not disrupt the user experience.

The precision of the hand recognition software has reached an acceptable level, however the system still lacks possibilities for implementing a larger variety of unique hand gestures. Ideas for how to address the problem has been suggested for future work. With the given timeframe, it was not possible to test the solution in a public area. However, feedback from the evaluation suggested that the solution could function well in a public space, leaving this point open for further investigation.

Considering what has been achieved in this project, it should be considered if this answers the research question formulated in the problem statement:

How can we make an interactive way to create small music sequences by using intuitive controls implemented through image processing, and possibly use this to support social activities in a public space?

By combining an easily approachable interface with intuitive controls by using your hand for navigation and manipulation, it is possible to engage users in the creation of music, regardless of experience. The prototype was able to engage the participants in the creation of music, and received positive critique for its intuitive feeling. With the improvements made regarding precision and consistency in the hand recognition software, the solution is ready to be evaluated in a public space.

For future work, it would be interesting to investigate if the solution can be used to support social activities in a public space. Several good ideas for approaches to this were given during the evaluation, such as educational and recreational uses. It would also be interesting to develop the music creation software further, by adding additional instruments and functionalities, while still keeping approachability in mind.

In conclusion, it seems that it is possible to engage users in the creation of simple music, no matter their previous experience. With further investigations, this could possibly be used for supporting social activities in a public space, by making the tools easily accessible for people.

10. Bibliography

1. Apple. GarageBand. Retrieved from http://www.apple.com/dk/ios/garageband/
2. David Benyon. 2010. *Designing Interactive Systems: A Comprehensive Guide to HCI and Interaction Design*. Addison Wesley.
3. David B. Clark. 1973. The Concept of Community: A Re-Examination. *The Sociological Review* 21, 3: 397–416. http://doi.org/10.1111/j.1467-954X.1973.tb00230.x
4. Amiraj Dhawan and Vipul Honrao. 2013. Implementation of Hand Detection based Techniques for Human Computer Interaction. *arXiv:1312.7560 [cs]*. http://doi.org/10.5120/12632-9151
5. Paul Dourish and Victoria Bellotti. 1992. Awareness and Coordination in Shared Workspaces. *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work*, ACM, 107–114. http://doi.org/10.1145/143457.143468
6. Dabney Frake. 2014. The Color Wheel. *Apartment Therapy*. Retrieved December 10, 2015 from http://www.apartmenttherapy.com/the-color-wheel-your-guide-to-choosing-perfect-paint-schemes-202653
7. Jeff Nusz. 2012. V Motion Project – Part I: The Instrument. Retrieved from http://www.custom-logic.com/blog/v-motion-project-the-instrument/
8. Jeff Nusz. Custom Logic. Retrieved from http://www.custom-logic.com/about/
9. Jette Holdgaard. PV11 Theory of science - validity and reliability. Retrieved from https://www.moodle.aau.dk/pluginfile.php/394051/mod_resource/content/1/PV%20MEA%20TS%202014.pdf
10. Michael Koch. 2005. Supporting Community Awareness with Public Shared Displays. *BLED 2005 Proceedings*. Retrieved from http://aisel.aisnet.org/bled2005/45
11. Thomas B. Moeslund. 2012. *Introduction to Video and Image Processing*. Springer London, London. Retrieved December 16, 2015 from http://link.springer.com/10.1007/978-1-4471-2503-7
12. Rolighetsteorin. *Piano stairs - TheFunTheory.com - Rolighetsteorin.se*. Retrieved December 15, 2015 from https://www.youtube.com/watch?v=2lXh2n0aPyw
13. Satoshi Suzuki and KeiichiA be. 1985. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing* 30, 1: 32–46. http://doi.org/10.1016/0734-189X(85)90016-7
14. University of Lowa. Electronic Music Studios. Retrieved from http://theremin.music.uiowa.edu/MIS-Pitches-2012/MISxylophone2012.html
15. V Energy NZ. *The V Motion Project -- Can't Help Myself [Official Music Video]*. Retrieved December 15, 2015 from https://www.youtube.com/watch?v=YERtJ-5wlhM
16. M.M. Youssef, K.V. Asari, R.C. Tompkins, and J. Foytik. 2010. Hull convexity defects features for human activity recognition. *Applied Imagery Pattern Recognition Workshop (AIPR), 2010 IEEE 39th*, 1–7. http://doi.org/10.1109/AIPR.2010.5759709
17. 2015. Nudge theory. *Wikipedia, the free encyclopedia*. Retrieved December 15, 2015 from https://en.wikipedia.org/w/index.php?title=Nudge_theory&oldid=695338849
18. FL STUDIO 12. Retrieved December 15, 2015 from https://www.image-line.com/flstudio/
19. Technitone. Retrieved October 17, 2015 from http://audiocrawl.co/crawl/technitone-98fc, Site is no longer available
20. Digital Surgeons. Retrieved from http://www.digitalsurgeons.com/
21. Audiocrawl- Soundboard. Retrieved from http://audiocrawl.co/crawl/soundboard-e888

22. André Michelle Archive: ToneMatrix. *André Michelle Archive*. Retrieved December 15, 2015 from http://lab.andre-michelle.com
23. gskinner. Retrieved December 15, 2015 from http://www.gskinner.com/
24. OpenCV. Retrieved December 16, 2015 from http://opencv.org
25. Hue, Value, Saturation | learn. Retrieved December 10, 2015 from http://learn.leighcotnoir.com/artspeak/elements-color/hue-value-saturation/
26. Unity Documentation. Retrieved from http://docs.unity3d.com/Manual/index.html

11. Appendix

A. Lo-Fi script

The test person is given a sheet, showing the different gestures.

The webcam is on and is filming for data collection.

They are the given the main create page where the general program is shown.

They are given an objective to drag a sound over to the grid, while they have to do the gesture for it.

They are then asked to play the melody with the "play" gesture.

They are asked to turn the volume up and down with the gestures for that.

They are then asked to stop the melody with the "stop" gesture.

A.1 Lo-Fi notes

Hovering finger - You could hold a finger "hover it" over the for example guitar and then listen to the guitar sound as an appetizer for what the sound could bring to the mix.

Volume control gesture – after recognizing the gesture it will display a volume bar for the user to better be able to follow the volume progress. Because or else many people believe that just holding the gesture still will increase. It mostly takes them 2-3 guess to get the right motions. One with a lot of musical experience thought that the gestures was natural and easy-to-do. Make different gestures for volume up and down. One of the users got the idea of how to control the volume how it was intended. There should be a better way to control the volume (rotating a few fingers to each side to control the volume).

Volume gesture without thumb (the hand gesture) – remove the thumb in the gesture, because it is frustrating and harder to do. Many thought this would be easier. Some of the testers did not care if it was with or without the thumb.

Release – some users have some problems figuring out what to do after having released, but this could probably be due to not seeing the interactive mouse/hand on the screen **(most after having released will not keep their released hand on the screen). One of the users hands were not flat against the screen. Grab and release gestures were fine.** All of the testers know how to use the grab and release gestures.

"Play gesture" – 'thumps up' was an idea by one of the testers, Use the "grab/release" as stop/start was another one (two users thought it). Another one was unsure of which specific gesture it should be, but that it should be one that differentiates from the others a lot (so that you did not activate play/stop accidently). Another gesture was metal/rock'n'roll gesture. Ok sign to stop the play. Idea to start and stop is be able to grab the timeline. Point to the sides to start and stop the play.

Some one did not close their hands entirely, they merely grasped with the element with their fingers. But some time after they closed their hands entirely.

Volume up – mostly just held their hand in the air, did not move it upwards

Volume down – someone turned their hands downside to do it that way

Volume bar - appearing would help understanding how to turn volume up and down…. ! Many people could need it to appear. ANOTHER one wanted the volume bar to be stationary and then it will appear all the time with a default setting and then the volume gesture would simply adjust the stationary bar.

Piano – He knows music and said that pianos need a lot more than 5 pitches

Volume gesture on a button – thought that the gesture in itself was confusing and would like to use a more natural gesture like turning on a volume button or something like that (2 out of 5) did the movement like it was more a button.

The music progress bar – would like to divide the sequence into "beats"

4 out of 10 that has something to do with music in their sparetime

Grab – de fleste forstår grab med det samme. Everybody. One thought grabbing with fingers would be better than grabbing with the whole hand.

Song sequency –

Pitch bars – Most of the users didn't get that at all without help. It is not obvious enough. Also no one understood the pitch-tones bar without getting help. BUT after having a tutorial in the pitch bars, they thought it was a good implementation but that it should be easier to understand.

Different sound object on same place on the grid – If you have 5 on the same grid, it could be difficult to remove the ones behind the others.

Placing sound objects on the same place on the grid – Could be difficult to control.

(garageband somebody used)

Grid system – creative music board then the grid system will fit better in, because a music composing software will demand more complexity.

Design – kind of dark right now. Not nice to look at, you need to focus too much. 2 out of 6 of the test users thought that the design was so dark it frustrated them.

General random thoughts – that it wasn't complex enough for a music composing program, the grid-system seemed unfamiliar if you were to compose songs. (Mostly for those who already knew about music software)

Issues with gestures - People with cognition problems or "handicap" could be a problem to use the gestures,

The numbers on the interface were a bit confusing and did not know what they meant. Adding some add-ons to the instruments to distort sound etc.

Adding some features with the angle your hand is. The interface was pretty simple and a new person would get the idea. The Y-axis is a bit confusing the –and+ do not know what it is for.

One of the testers got the idea of x-axis and y-axis. An options to be able to extend the tones for multiple seconds instead of just the tone.

Might be hard to edit when there is 5 instruments in one box.

Look tutorials – Traktor og Ableton Live (for opbygning)

M4Sonic – Weapon Live Mashup

B. Evaluation script

First sign the consent form, while getting introduced to the prototype.

Introduce them to the different gestures.

Let them try out the gestures while watching themselves on webcam.

Switch to the interface and let them try to make a little melody.

Allow them to test the program for about 1-2 minutes.

Ask them to start and stop the melody.

Ask them to turn the volume up and down.

Give them the assignments

This is to test how difficult it is to select the right notes that you want to play.

Switch over to the questions

C. Evaluation questionnaire

Please follow the script below so the structure remains intact.

Did you encounter any problems with the program during your test period?

What is your opinion on the program's interface?

Did you find the program easy to use and understand? (1-5)

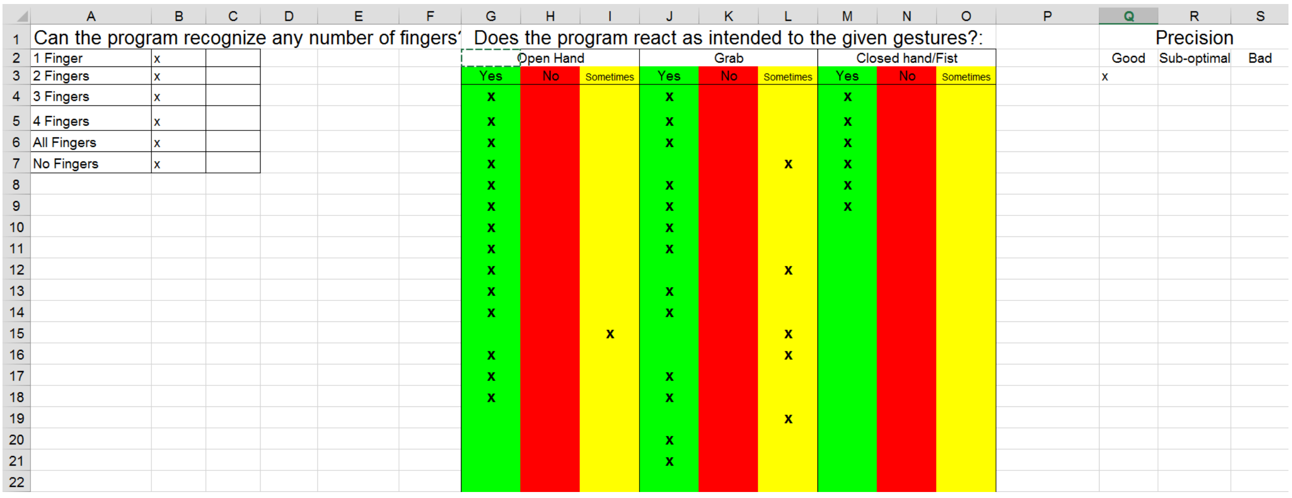
Did you find the program intuitive? (1-5)

Would you prefer a mouse over the hand gestures?

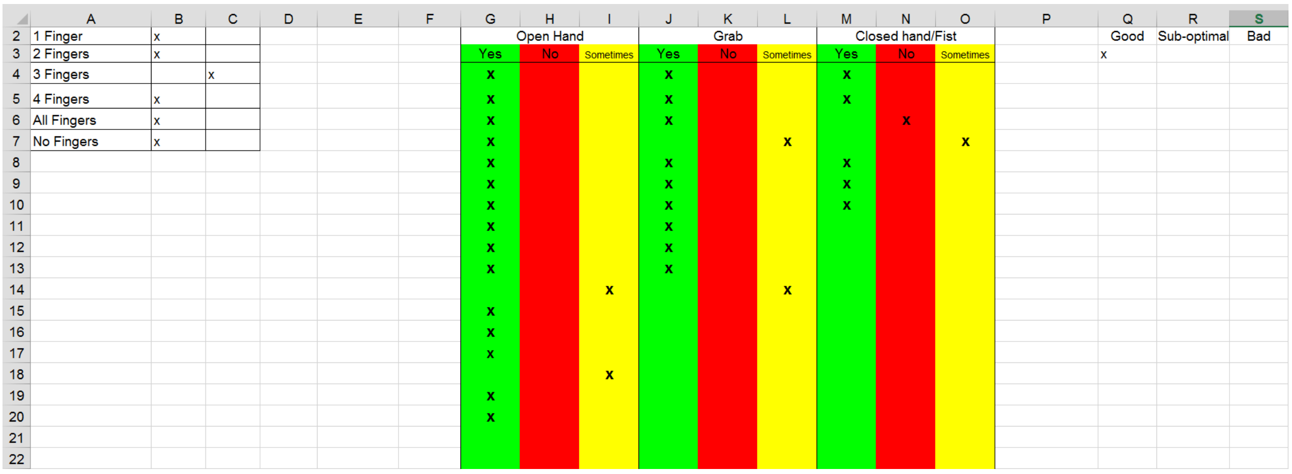
Any suggestions for the program that could improve it?

What do you think about creating music in this way and do you think it could be used in a public environment?

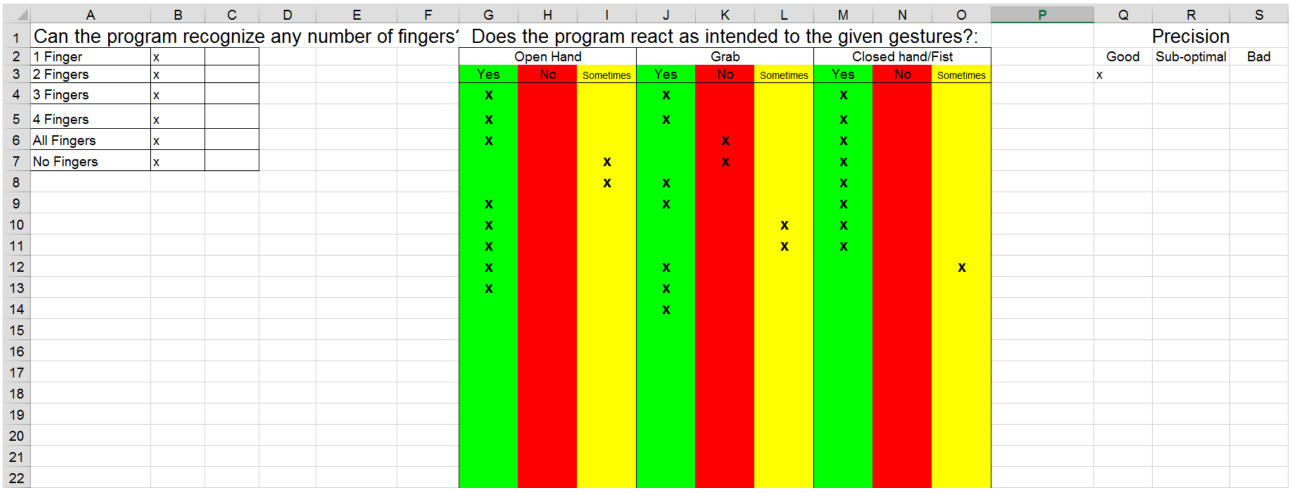
D. Evaluation performance sheet



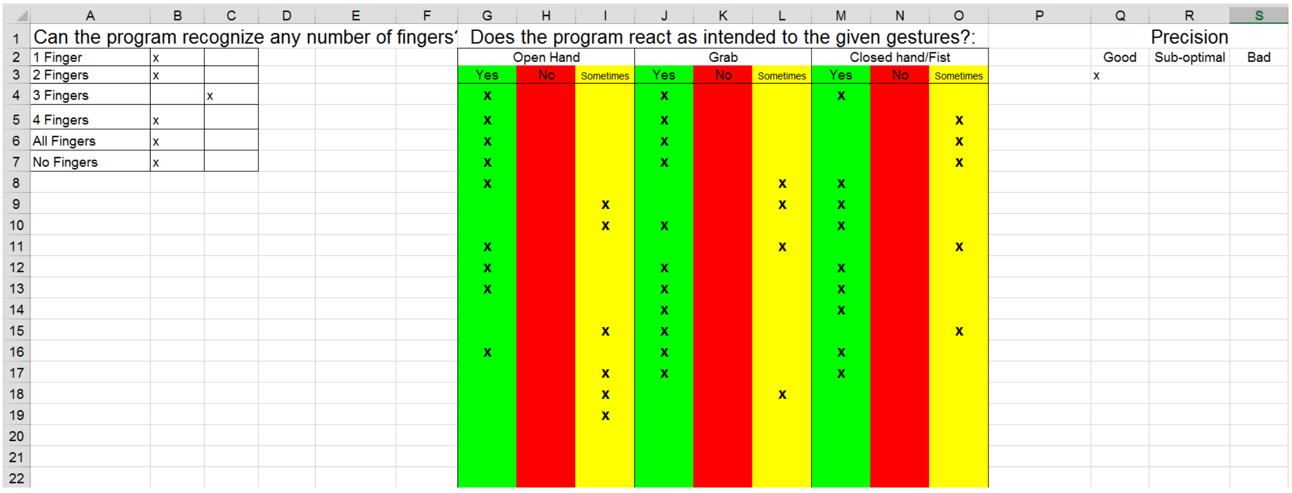
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Can the program recognize any number of fingers'						Does the program react as intended to the given gestures?:										Precision		
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers	x					x			x			x						
5	4 Fingers	x					x			x			x						
6	All Fingers	x					x			x			x						
7	No Fingers	x					x					x	x						
8							x			x			x						
9							x			x			x						
10							x			x									
11							x			x									
12							x					x							
13							x			x									
14							x			x									
15									x			x							
16							x					x							
17							x			x									
18							x			x									
19												x							
20										x									
21										x									
22																			



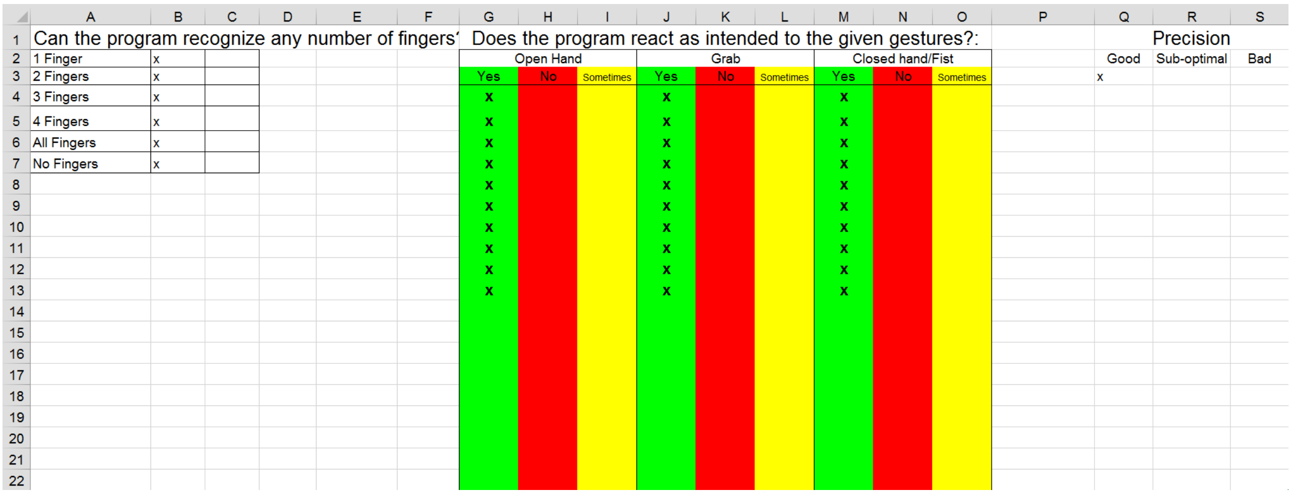
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers		x				x			x			x						
5	4 Fingers	x					x			x			x						
6	All Fingers	x					x			x				x					
7	No Fingers	x					x					x			x				
8							x			x			x						
9							x			x			x						
10							x			x			x						
11							x			x									
12							x			x									
13							x			x									
14									x			x							
15							x												
16							x												
17							x												
18									x										
19							x												
20							x												
21																			
22																			



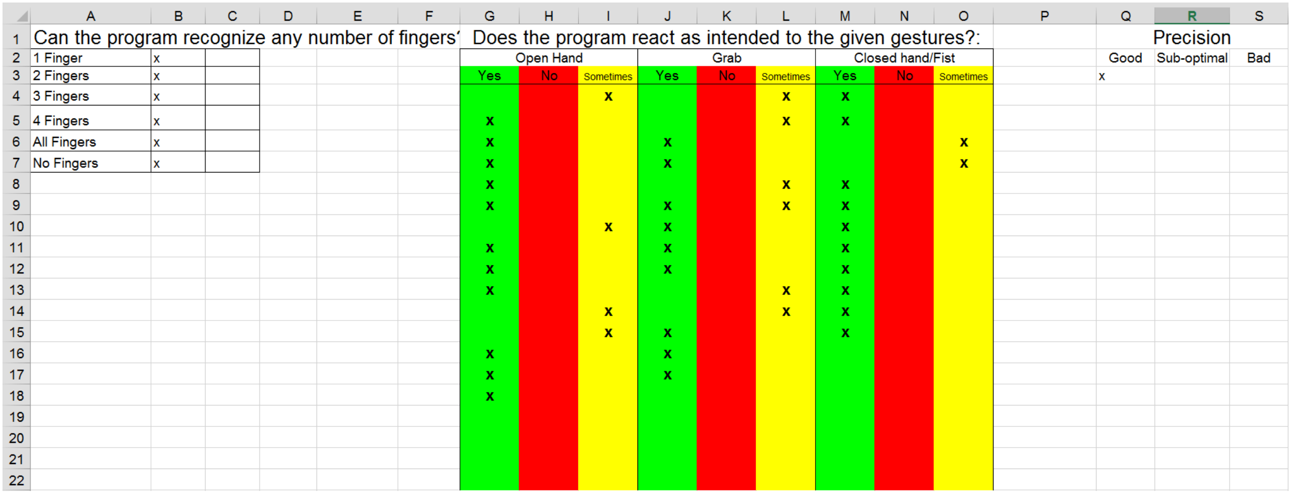
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Can the program recognize any number of fingers‘						Does the program react as intended to the given gestures?:											Precision	
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers	x					x			x			x						
5	4 Fingers	x					x			x			x						
6	All Fingers	x					x				x		x						
7	No Fingers	x							x		x		x						
8									x	x			x						
9							x			x			x						
10							x					x	x						
11							x					x	x						
12							x			x					x				
13							x			x									
14										x									
15																			
16																			
17																			
18																			
19																			
20																			
21																			
22																			



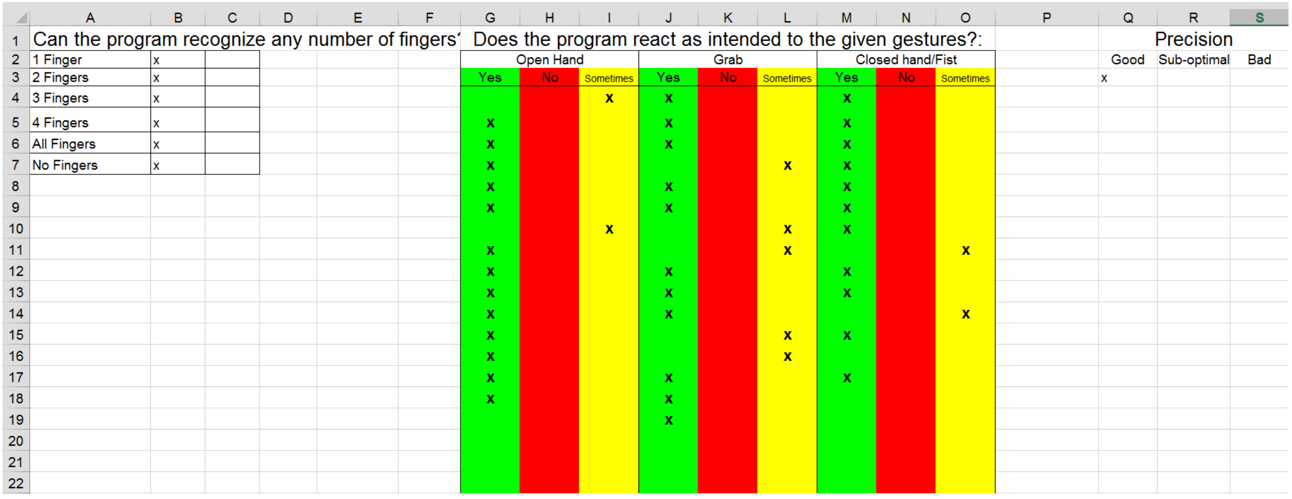
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Can the program recognize any number of fingers‘						Does the program react as intended to the given gestures?:											Precision	
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers		x				x			x			x						
5	4 Fingers	x					x			x					x				
6	All Fingers	x					x			x					x				
7	No Fingers	x					x			x					x				
8							x					x	x						
9									x			x	x						
10									x	x			x						
11							x					x			x				
12							x			x			x						
13							x			x			x						
14										x			x						
15									x	x					x				
16							x			x			x						
17									x	x			x						
18									x			x							
19									x										
20																			
21																			
22																			



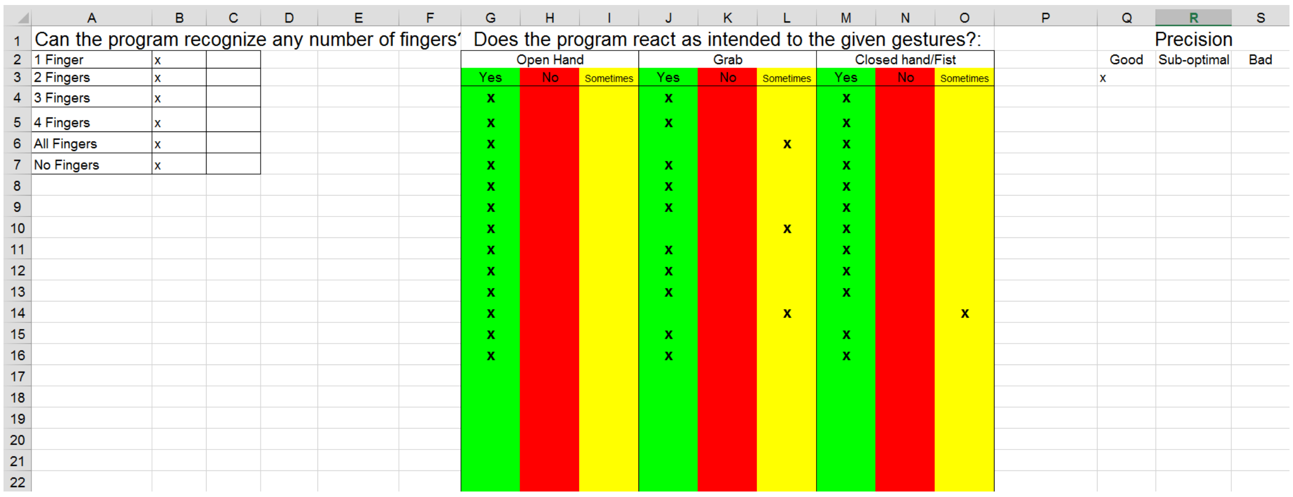
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Can the program recognize any number of fingers'						Does the program react as intended to the given gestures?:											Precision	
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers	x					x			x			x						
5	4 Fingers	x					x			x			x						
6	All Fingers	x					x			x			x						
7	No Fingers	x					x			x			x						
8							x			x			x						
9							x			x			x						
10							x			x			x						
11							x			x			x						
12							x			x			x						
13							x			x			x						
14																			
15																			
16																			
17																			
18																			
19																			
20																			
21																			
22																			



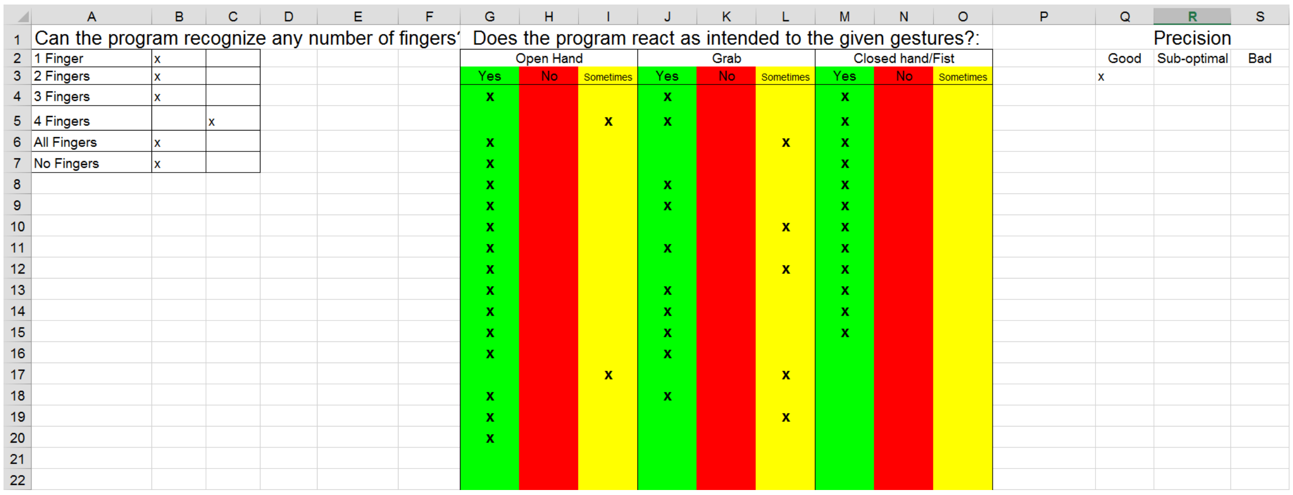
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Can the program recognize any number of fingers'						Does the program react as intended to the given gestures?:											Precision	
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers	x							x			x	x						
5	4 Fingers	x					x					x	x						
6	All Fingers	x					x			x					x				
7	No Fingers	x					x			x					x				
8							x					x	x						
9							x			x		x	x						
10									x	x			x						
11							x			x			x						
12							x			x			x						
13							x					x	x						
14									x			x	x						
15									x	x			x						
16							x			x									
17							x			x									
18							x												
19																			
20																			
21																			
22																			



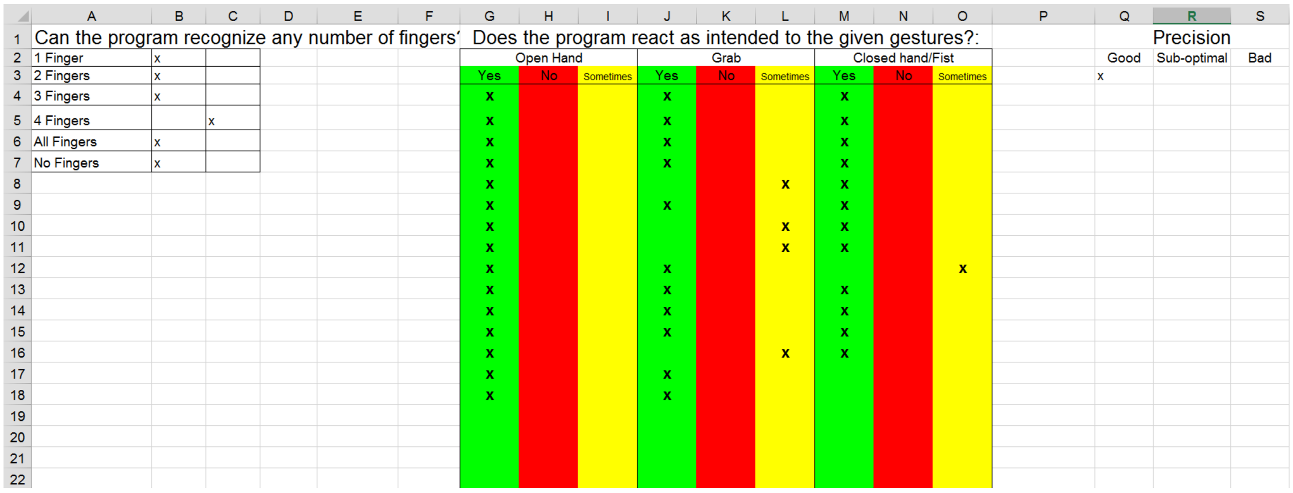
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Can the program recognize any number of fingers'						Does the program react as intended to the given gestures?:											Precision	
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers	x							x	x			x						
5	4 Fingers	x					x			x			x						
6	All Fingers	x					x			x			x						
7	No Fingers	x					x					x	x						
8							x			x			x						
9							x			x			x						
10									x			x	x						
11							x					x			x				
12							x			x			x						
13							x			x			x						
14							x			x					x				
15							x					x	x						
16							x					x							
17							x			x			x						
18							x			x									
19										x									
20																			
21																			
22																			



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Can the program recognize any number of fingers'						Does the program react as intended to the given gestures?:											Precision	
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers	x					x			x			x						
5	4 Fingers	x					x			x			x						
6	All Fingers	x					x					x	x						
7	No Fingers	x					x			x			x						
8							x			x			x						
9							x			x			x						
10							x					x	x						
11							x			x			x						
12							x			x			x						
13							x			x			x						
14							x					x			x				
15							x			x			x						
16							x			x			x						
17																			
18																			
19																			
20																			
21																			
22																			



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Can the program recognize any number of fingers'						Does the program react as intended to the given gestures?:											Precision	
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers	x					x			x			x						
5	4 Fingers		x						x	x			x						
6	All Fingers	x					x					x	x						
7	No Fingers	x					x						x						
8							x			x			x						
9							x			x			x						
10							x					x	x						
11							x			x			x						
12							x					x	x						
13							x			x			x						
14							x			x			x						
15							x			x			x						
16							x			x									
17									x			x							
18							x			x									
19							x					x							
20							x												
21																			
22																			



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Can the program recognize any number of fingers'						Does the program react as intended to the given gestures?:											Precision	
2	1 Finger	x					Open Hand			Grab			Closed hand/Fist				Good	Sub-optimal	Bad
3	2 Fingers	x					Yes	No	Sometimes	Yes	No	Sometimes	Yes	No	Sometimes		x		
4	3 Fingers	x					x			x			x						
5	4 Fingers		x				x			x			x						
6	All Fingers	x					x			x			x						
7	No Fingers	x					x			x			x						
8							x					x	x						
9							x			x			x						
10							x					x	x						
11							x					x	x						
12							x			x					x				
13							x			x			x						
14							x			x			x						
15							x			x			x						
16							x					x	x						
17							x			x									
18							x			x									
19																			
20																			
21																			
22																			

E. Consent forms

Lo-Fi Information Sheet for Participants

We are Medialogy students from Aalborg university.
We are interested in hearing your feedback on our project. We will ask you to try out our prototype and ask you some questions.
This will roughly take 10 minutes and you are free to

- not answer any questions or perform tasks that makes you feel uncomfortable,
- request to stop and delete the recordings
- end the session at any time.

While talking to you we would like to (video) record our conversation and how you’re interacting with the prototype. We will transfer and store the recordings for further analysis. If you give consent, images may be used for scientific publication purposes and our project report and its presentations only.

Participant Consent Form

I ________________________________ hereby agree to voluntarily participate in the research being undertaken. I have been informed that the consent to participate can be revoked at any time, or that I can have certain identifying data withheld from the research publications.

I give permission for recorded images and/or videos to be:

- taken of me during the interview and trial Yes ☐ No ☐, and
- re-produced in scientific publications, student reports and presentations: Yes ☐ No ☐

Name:

Signature: Date: / /

Evaluation information Sheet for Participants

We are Medialogy students from Aalborg university.
We are interested in hearing your feedback on our project. We will ask you to try out our prototype and ask you some questions.
This will roughly take 10 minutes and you are free to

- not answer any questions or perform tasks that makes you feel uncomfortable,
- request to stop and delete the recordings
- end the session at any time.

While talking to you we would like to (video) record our conversation and how you're interacting with the prototype. We will transfer and store the recordings for further analysis. If you give consent, images may be used for scientific publication purposes and our project report and its presentations only.

Participant Consent Form

I ________________________________ hereby agree to voluntarily participate in the research being undertaken. I have been informed that the consent to participate can be revoked at any time, or that I can have certain identifying data withheld from the research publications.

I give permission for recorded images and/or videos to be:

- taken of me during the interview and trial Yes ☐ No ☐, and
- re-produced in scientific publications, student reports and presentations: Yes ☐ No ☐

Name:

Signature: Date: / /