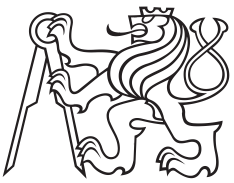
Bachelor's thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science and Engineering

EMG-controlled Tool for Image Editing

Tomáš Flek

Study Program: Open Informatics
Study Field: Software Systems

May 2014
Supervisor: Ing. Adam Sporka, Ph.D.

Acknowledgement / Declaration

I would like to thank to my advisor Adam Sporka for guidance, inspiration and valuable advices. I would also like to thank to my friends and family for support and motivation.

I declare that I worked out the presented thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis.

.....
Tomáš Flek

In Prague, 18 May 2014

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

.....
Tomáš Flek

V Praze, 18. května 2014

Abstrakt / Abstract

Tato práce je zaměřena na design, implementaci a testování jednoduchého grafického nástroje ovládaného elektromyografickými senzory. Většina lidí používá různé grafické editory, které ale nejsou obvykle přístupné pro lidi s těžkým pohybovým postižením. Lidé s tímto postižením interagují s počítačem skrze limitovanou množinu vstupních „signálů“, jako je stisk tlačítka, zatnutí svalu, atd. Elektromyografické senzory poskytují prostředky pro fyzické realizace těchto vstupních kanálů. Cílem této práce bylo použít sadu takových senzorů, které umožňují jednoduchou editaci GIF obrázků.

Klíčová slova: asistivní technologie, pohybová postižení, elektromyografie, animace, editace obrázků

Překlad titulu: Jednoduchý grafický nástroj ovládaný elektromyografickými senzory

This work focuses on design, implementation, and testing of an EMG-controlled tool for image editing. Most people use various graphic editors that are usually not accessible by people with severe motor impairments. People at this level of disability interact with computer via limited set of input “signals”, e.g. pressing a button, clenching muscles, etc. Electromyographic sensors provide means of physical realization of these input channels. The goal of this work was to use a set of such sensors that enable a simple GIF image editing.

Keywords: assistive technology, motor impairments, electromyography, animation, image editing

Contents /

1 Introduction 1
1.1 Quality of life 1
1.2 Digital content creation and accessibility 2
1.3 Support of creativity vs. complexity 2
1.4 Increasing participation in society 3
1.5 Summary and motivation 3
2 Assistive technology 4
2.1 Accessibility 4
2.2 AT supporting art 4
2.2.1 Speech/voice recognition ..5
2.2.2 Eye-tracking 6
2.3 EMG signal detection potential 7
2.4 Summary of relevant findings ...8
3 Graphic design workflow 9
3.1 Animation methods 9
3.2 Pixel art 10
3.2.1 Animation creation process 11
3.3 Summary of relevant findings . 11
4 Application design 12
4.1 General use case overview 12
4.2 Choice of relevant functions ... 13
5 Application interaction design .. 16
5.1 Input signal adaptation 16
5.2 Application environment 17
5.3 Workflow that we support 19
5.4 Overview of functions 20
5.5 Mapping low-channel input ... 20
5.5.1 Mapping in the GifPix Maker 21
6 Implementation 22
6.1 Requirements 22
6.2 Used technologies 22
6.3 Used design patterns 23
6.4 Raw data implementation 23
6.5 GUI implementation 24
6.6 Input signal adaptation 25
6.7 Input signal interpretation 26
6.7.1 2-signal input 27
7 Testing 29
7.1 Setup 29
7.2 Participants 30
7.3 Choice of use case 30
7.4 Testing process 31
7.5 Overview of findings 32
8 Conclusion 34
8.1 Future work 34
References 35
A Abbreviations 37
B CD Content 38
B.1 Software 38
B.2 Text of this thesis 38
B.3 Other files 38

Tables / Figures

7.1. Overview of findings revealed by testing 33

2.1. Screenshot of the VoiceDraw application 5
2.2. Example of a drawing created by using the VoicePen 6
2.3. Screenshot of the EyeDraw Version 2 7
3.1. Example of pixel art poster 10
3.2. Example of pixel art in non-digital form 10
4.1. Initial and Orientation menus of the GifPix Maker 14
4.2. Tools and Colors menus of the GifPix Maker 15
5.1. Start up dialog of the GifPix Maker 16
5.2. Diagram of the scanning method in the GifPix Maker ... 17
5.3. Information dialog of signals of the GifPix Maker 17
5.4. Screenshot of the GifPix Maker's environment 18
5.5. Diagram of an interaction in the GifPix Maker 19
5.6. Placement of the electrodes on the forearm 20
6.1. Diagram of the View package . 24
7.1. Photo taken during testing showing setup in the lab 29
7.2. Decomposed original animation used as a model for testing 30
7.3. Images created by participants during testing 32

Chapter 1
Introduction

Creating and editing graphic content is one of the main creation processes at all. Dozens of programs for simple graphic creation exist, but their usability for motor-impaired people is very bad, sometimes none. In this work will be presented a development and implementation of a program for simple image editing controlled by electromyographic signals. A detection of these signals allows simple control even for severely motor-impaired people.

1.1 Quality of life

Motor-impaired people are generally excluded from the society, which is based on physical health. To improve the quality of life, we have to improve basic aspects such as social belonging and everyday emotional experiences. A tool presented in this work should bring at least a distraction from their everyday problems. In the best case of scenario, it should bring them joy of creating something original without the help of others.

One of the most general evaluations of an individual well-being is the quality of life. This assessment of an existence includes various factors, unlike standard of living, which is based mainly on the income[1]). The main factors affecting the quality of life are:

- physical health
- mental health
- education
- leisure time
- social belonging

The quality of life can be measured in different ways. It depends on the field of study and on the context. The most frequently used approach is a quantitative measurement. It is based on an assignment of examined fact to a measurable context. The measurable context is for example a number or different scale.

It has been recently distinguished into two aspects of a personal well-being. The first, *life evaluation*, in which respondents are asked to think about their life generally and evaluate it against the scale. The second, and for our purpose interesting aspect, is *an emotional well-being*. Unlike life evaluation, respondents are asked about their every day emotional experiences. The quality of these experiences is measured as their frequency and intensity. The main measured emotions are:

- joy
- stress
- anger
- sadness
- affection

[1]) `http://en.wikipedia.org/wiki/Quality_of_life`

According to Robert M Kaplan [1], a health-related quality of life consist of two main parts. The first part is a functioning – *what person can do*. This includes self-care, role in society and whether person is socially active. The second part is a well-being – *how the person feels*. This includes emotional well-being, pain and energy.

Measuring a health-related quality of life is similar as it was described in the previous paragraph. Respondents are asked about routine procedures they do such as if they are able to dress and bath themselves. A survey also contains questions about everyday emotions such as if they are nervous more than they used to be.

1.2 Digital content creation and accessibility

One of the most important and basic human need is a self-expression [2]. Expressing our thoughts and feelings is an everyday routine. The most common is creating digital content. A digital content is specific that it always exists in digital data. This content can be stored in various formats – from text, image, audio to video. This gives user tremendous freedom, so he or she can create content composed of several formats at once.

To create a digital content, it is necessary to interact with a digital device – most frequently a computer. Many computer programs and applications for different platforms and different format creation exist but most of them are not accessible. They can not be used for severely motor-impaired people at all.

Imagine that you want to create an animation. What program would you use? How much time it would take you to create simple image? And how much time it would take if you can use just a few muscles on your body? An interaction with these programs requires high accuracy or it just can not be controlled with only a few input signals. Even if the program enables control by only a few signals, it is sometimes very hard to use or it takes very long time to create a simple thing. It is important to enable a digital content creation for such people, so they can also express themselves.

Other important aspect of accessibility is a possibility of adjustment. Motor-impaired people can interact with a computer in many different ways. This is closely tight to their type of disability. Therefore, program should be easily adaptable to different forms of input. Some people can simply use a few keys on a computer, other can use only their voice and some can use just a few muscles on their body. The program presented in this work supports adaptation to different input forms.

1.3 Support of creativity vs. complexity

A process where something new is created is called creativity. A thing that was created during this process can be a picture, a video, a text, a sound, even just an idea. In today's world, the easiest and the fastest way to create something new is by using a computer. There are many computer programs for content creation as it was mentioned in the previous Chapter. Available programs that does not restrict creativity are complex.

Programs for motor-impaired people should not be complex, on the contrary, they should be accessible and usable even for severely motor-impaired people. To meet this criterion it is not possible to create a program enabling complex creation. Solution is to create several specialized programs, which are focused on specific content creation, rather than one complex program. By minimizing this area of activity, program could be smaller, more minimalist and easier to control.

Recently, programmers are more interested in creating different tools for content creation to support creativity process. Thanks to wider education and greater awareness of people with disabilities, number of specialized programs has increased. In this work is presented the program specialized on a specific content creation – low resolution GIF animation.

1.4 Increasing participation in society

As it was mentioned earlier, social belonging is one of the most important factors of the quality of life. This implicates that improving participation in society should also improve the quality of life. According to Claire Wallace and Florian Pichler [3], participation in society can bring variety of indirect social rewards as well as direct personal rewards. One of the main rewards are access to friends, networks and jobs.

Motor-impaired people are usually partially excluded from the society. Their disabilities can affect normal functioning just a little. But especially for severely motor-impaired people it has a huge impact on their lives. Basic need – which is the most commonly affected – is the ability to move. In order to offset this deficiency, these people has someone who helps them or some specialized hardware or software tools. In most cases a combination of these tools is needed.

If we take a look on severely motor-impaired people we will see that they have very specific needs. In most cases they need specialized tools to control a wheelchair or computer. These tools are often very uncomfortable so they can not use them for extended period of time. These tools often try to simulate a computer mouse. It is important to realize that such users can not reach the same accuracy as a healthy person.

There are many projects focused on alternative text input methods for people with disabilities. It is important to realize that not everything can be expressed as a text. Emotions and generally a state of mind can be expressed properly only in art. Creating art leads to increased awareness in a society and therefore to participation in a society. Definitely, developing programs enabling image content creation for people with disabilities supports participation in the society.

1.5 Summary and motivation

The main goal is to provide **a tool for graphic content creation for motor-impaired people**. This tool should be:

- accessible
- easy to use
- minimalist
- adaptable

With this motivation was developed a computer program called the *GifPix Maker* which enables a simple creation and modification of GIF images in low resolution. This program can be controlled by only 2 (or more) input signals. This allows even severely motor-impaired people to control this program and easy operate with it. This program can be also easily adapted to different input forms by implementing its interface.

This program will hopefully allow people with disabilities to express themselves by creating original piece of art, which would improve their well-being and awareness in the society.

Chapter 2
Assistive technology

The assistive technology is an umbrella term[1]), which includes assistive and adaptive devices and specialized software and hardware that help people with disabilities. This chapter is focused on the assistive technology for motor-impaired people and creating art. The goal of this chapter is to introduce different approaches in the assistive technology and their pros and cons.

2.1 Accessibility

One of the main reasons why to talk about the accessibility is that modern graphical user interfaces assume users have the ability to move a mouse cursor, especially in painting and drawing applications. In fact, many users with motor impairments have difficulty moving a mouse cursor or performing a variety of other continuous control tasks [4].

Generally, an accessibility is a degree to witch an environment is accessible to as many people as possible. That is something that we should think about every time new service for people is created. Due to wide scale of disabilities, it is not possible to create something that is accessible and comfortable at the same time to all people [5].

In human-computer interaction, an accessibility means to enable an easy interaction with computer program to as many people as possible regardless of severity or disability of an impairment. As it was mentioned in the previous paragraph, it is not possible to create accessible and comfortable computer program for all people. For example different text editors and internet browsers enables to enlarge content and sometimes even user interface which helps people with visual impairment.

A more difficult situation arises when user is physically handicapped. There are two main parts that has to be handled.

- The first is how the user will interact with a computer due to its disability.
- The second part is if the program is usable with given type of an interaction.

Severely motor-impaired users needs a specialized interaction tool that will allow them to send input signals to computer. Also, they need a specialized program which will be ready to adapt to those signals. Different approaches of an assistive technology according to specific impairments were described widely by Andrew Sears and Mark Young [6].

2.2 AT supporting art

The main goal in an assistive technology for art is to eliminate control by hand, which is represented as a computer mouse. The voice, eye movement or head movement as a substitude for mouse are often used. Another approach which will be mentioned and used for the *GifPix Maker* is a detection of surface EMG signals.

Examples of existing tools supporting art with different software approaches follows.

[1]) `http://en.wikipedia.org/wiki/Umbrella_term`

2.2.1 Speech/voice recognition

This approach is used very often. It is translation of spoken words (or generally voice) into text. The text is further evaluated and tight to determined action. This approach is suitable even for severely motor-impaired people because in the most cases the voice is intact. But if the voice recognition is based on holding the same tone for the long time, it could by very uncomfortable.

Existing tools using speech or voice recognition are for example:

- VoiceDraw[7]

It is a hands-free voice-driven drawing application for people with motor impairments that provides a way to generate free-form drawings without needing manual interaction. It uses human voice to control application environment and to control brush strokes. Application was designed in two versions. One is fully controlled by speech recognition which means that commands like “start”, “stop”, “upper right” and “faster” are used for interaction. The second version uses recognition of vocal sounds that do not correspond to any words or phrases in a language. For example, pitch, volume, and vowel quality are all continuous features of voice that may be manipulated by users in control tasks.

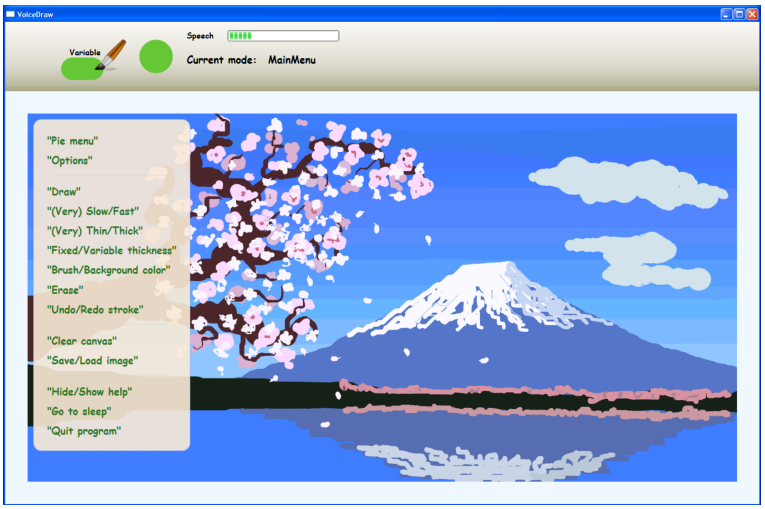


Figure 2.1. Screenshot of the VoiceDraw application. The first author created this painting using only his voice in about 2.5 hours [7].

Pros of this application are that no limb movement is needed for interaction and drawing is significantly faster when the user can use speech without limitations. Also using voice reducing fatigue. The main cons are it takes time to memorize mappings of the vowels to the directions. Also in case of low lung capacity it is not possible to make long strokes and smooth curves.

- VoicePen[8]

It is a drawing application which uses voice input as a secondary modality. Non-linguistic vocalization is used to augment digital pen input on a tablet computer. Motivation was that in many scenarios the user can benefit from the ability to continuously vary additional parameters. The application implements various interaction techniques to explore potential of combination of two modalities. One of the main interaction techniques is brush stroke creation. Various parameters of a brush stroke may be manipulated while the stroke is being drawn, such as opacity, color, hardness, shape or

thickness. Non-linguistic vocalization can be also used for other interaction techniques such as object manipulation (rotation, scaling) or workspace navigation.

Figure 2.2. Example of a drawing created by using the VoicePen, where the digital stylus was used to specify the stroke thickness and non-speech vocalization was used to specify the opacity [8].

Pros of this application are that user can adjust different variables with combination of digital pen. This speeds up drawing process and enables to create more complex brush strokes which would be normally difficult to create. Main cons are that users have problem with memorizing a pronouncing correct vowels. According to participants, it was sometimes hard to think about several variables such as opacity and thickness at the same time.

2.2.2 Eye-tracking

This method has a quite long history in HCI because of its potential [9]. Earlier, eye-tracking procedure was invasive and definitely not comfortable. But today, we can map eye movement just with camera and suitable recognition software. Eye movements has been used in real time as an input to the user–computer dialogue many times but it carries with it several problems [10–11]. The main problem is an accuracy. Eye-tracking can not be used as a substitution of a computer mouse for several reasons. Accuracy of an eye is not as good as might be expected – many of eye movements are involuntary which might lead to unwanted actions. Also eye movement does not support button clicks which can be emulated by blinking but it is neither reliable nor comfortable.

Eye-tracking can be generally used in most cases because eyes are generally intact even in case of severely motor-impaired people. The main drawback is the need for high-end equipment and high cost, configuration and maintenance. Previous works showed that these factors can significantly affect adoption process of an assistive technology [12].

Existing tools using eye-tracking are for example:

- EyeDraw[13–14]

It is a drawing application for people who can not move their limbs or speak, such as people with partial paralysis resulting from brain injury. The application design is centered for children. Whole interaction with application is provided by an eye

movement. System detects gazepoints, saccades and long fixations called dwells of an eye. These signals are evaluated and transferred to appropriate actions. Dwells detection serves as "click" commands and also prevents from unwanted actions caused by involuntary eye movement. The application enables drawing of simple shapes as line, rectangle and polygon shapes, color and predefined stamps which can be placed into image.

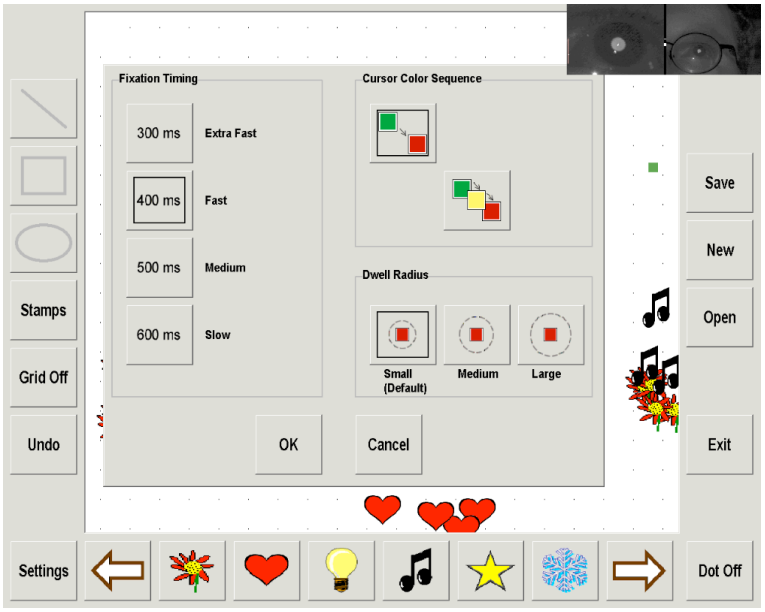


Figure 2.3. Screenshot of the EyeDraw Version 2 with the eye-controlled Settings dialog box opened [13].

Pros of this application is that it can be used even for severely motor-impaired people even if they can not speak. Even children are enable to draw pictures by only moving their eyes. This application also support the natural human developmental pattern of learning to draw. General cons of this application result from using the eye-tracking method which is not very comfortable and affordable.

2.3 EMG signal detection potential

A detection of EMG signals is used mainly for medical purposes. However, these signals can be used in assistive technology as an input modality [6]. People with severe disabilities are generally still able to perform muscle contractions. These contractions can be detected by surface electrodes and by using recognition software transferred into input signals. This enables almost every person to interact with an interface.

The detection of these signals were used for example in system controlling wheelchair [15]. It has shown that system is very transparent to the user and it is extremely fault-tolerant and easy to use. It took only several minutes to get familiar with the system and to join action with reaction.

The detection of EMG signals were also used in order to create system for text entry [16]. System implements two text input methods which enables to type at speed up to 4.5 words per minute. It took also several minutes to get familiar with system. With combination of word prediction system provides effective tool for text entry.

Muscle contractions generally require extremely little effort, which makes the system suitable even for people suffering from very severe physical disabilities. This enables

to use some device or interact with interface for a extended period of time without the exhaustion and pain.

Sadly, no program supporting art using the EMG signal detection exists. This is a motivation to create program which could use these signals as an input modality and enables to create graphic content.

2.4 Summary of relevant findings

The assistive technology provides many tools for human-computer interaction. Unfortunately, a huge amount of programs for art creation are still not accessible for all users and definitely not accessible for severely motor-impaired users. That is the reason why specialized program are developed using different assistive technologies.

Existing programs for art creation now uses only two approaches:

- speech/voice recognition
- eye-tracking

Both approaches has its limitation in case of severely motor-impaired people or can not be used in some circumstances. The third considered but not yet used approach for an application for art creation is

- **EMG signals detection**

Because muscle contraction requires extremely little effort, which makes the system suitable even for people suffering from very severe physical disabilities, it was decided that the *GifPix Maker* developed in this work will use EMG signals as an input modality. That will enable to use this application to as many people as possible independently on a severity and type of their impairment.

Chapter 3
Graphic design workflow

The main goal of this project is to support one of the graphic design workflows – animation creation. To do so, we have to know what the graphic design is and how it is created. The graphic design is a category of art used for visual communication. Disciplines of the graphic design are for example logos, logotypes, advertisement, posters, typogpraphy of books and magazines. Designs are made on order for a specific purpose and are intended for industrial processing – printing, web design etc.

This chapter is focused on one part of graphic design – animation. Creating a new animation is under the influence of general graphic design workflow. It comprises all the necessary steps that have to happen for a particular job to be completed. But it is important to realize that every workflow is different because every project has different goals [17].

3.1 Animation methods

Animation as the process of creating a continuous motion and shape change can be created by different approaches. Animation creation methods include the traditional animation creation method and those involving stop motion animation of two and three-dimensional objects. Images are displayed in a rapid succession, usually 25 or 30 frames per second which creates visual feeling of fluent move of an object in the picture.

There is overview of existing 2D animation techniques[1]):

- Traditional animation

  It is a traditional technique where frames are drawn by hand. Until the advent of computer animation it was the dominant form of an animation in cinema. The illusion of movement is created by slightly differs in frame drawings.
  - Full animation – animation is produced by use of detailed drawings and plausible movement.
  - Limited animation – despite full animation, limited animation reuses common parts between frames.

- Stop motion animation

  This technique of animation creation uses manipulation with real-world objects. After every manipulation with an object the picture is taken. Each picture represents one frame of animation and this gives feel of object movement.
  - Puppet animation – involves puppet figures interacting in constructed environment.
  - Clay animation – uses figures made of clay or a similar malleable material.
  - Cutout animation – produced by moving two-dimensional pieces of material such as paper or cloth.

[1]) `http://en.wikipedia.org/wiki/Animation`

- Computer animation

  Enables variety of animation types creation. Main approaches are the vector and raster graphics. The raster graphics is based on dot-matrix data structure representing points of colors. The vector graphics is based on geometrical primitives such as points, lines, curves and polygons. But in both cases animation creation uses automated computerized versions of traditional animation techniques.

3.2 Pixel art

The pixel art is a part of a computer graphics where an image is created by coloring each pixel by using raster graphics software. Generally, creating realistic image requires large amount of time. Pixel art is really simple to understand and use for even people who are not graphic designers. That is why it is used in the *GifPix Maker*. People with severe disabilities would not use software where it takes hours to create simple image or which would be unnecessarily complex.

Figure 3.1. Pixel art poster created by Eboy[1]).

However, pixel art can be found everywhere, not just as computer graphics. As you can see in Figure 3.2 pixel art of French urban artist called Invader whose work is modelled on the crude pixellation of 1970s 8-bit video games.

Figure 3.2. Pixel art outside the virtual domain using mosaic tiles by Invader[2]).

[1]) http://blog.templatemonster.com/2014/03/26/pixel-art-echos-web-design/
[2]) http://en.wikipedia.org/wiki/Pixel_artist

3.2.1 Animation creation process

Creating a pixel art animation composes of a few basic steps.

- **Determining animation number of frames and dimensions**
  This is specified mainly by the use of an animation. Web icon would be small but for example a computer game simple animation would be bigger and more fluent, so its necessary to create more frames with minimum change of image.
- **Edition of individual frames**
  Generally, author will not paint frame by frame, color pixel by pixel, in ascending order from the first to the last. Firstly, author creates several frames in some distance to draw-up main animation phases. Then, when these phases are painted, user fills frames between these significant phases and completes the whole animation. During editing process user has to have ability to move among animation frames. Also because of slight changes in each image user needs copy tool so he would do not have to draw each image again from scratch.
- **Composition of frames into one single image**
  Once all frames are created, author joins them into one GIF image. Before composition it is necessary to set image properties like frames per second speed. This frequency determines animation speed and fluency. GIF animations can be also repeatable or non-repeatable. This determines whether a sequence of frames would be shown only once or several time several times or displayed in loop.

3.3 Summary of relevant findings

One of the simplest ways to create art is to create the pixel art. No specialized software is necessary to create a pixel image. Any raster graphic editor with zoom feature will serve just fine. This is why from all of the possibilities of creating art for motor-impaired people was chosen the pixel art. Its easy to create something even for people who are not graphic designers. Another reason why the pixel art was chosen for the *GifPix Maker* was that existing painting tools for motor-impaired people presented in Chapter 2.2 support mainly brush painting. None of existing specialized programs for motor-impaired people support an easy creation of an animation.

To provide graphic tool for image editing for motor-impaired people that brings something new and is easy to control and understand was decided that the *GifPix Maker* will:

- **support pixel art**
- **support animation creation**

Chapter 4
Application design

In this chapter is presented how the design of the *GifPix Maker* was developed. The application design is based on use cases that are expected in use of an application. Use cases are technique for expressing functional requirements and for describing the behavior of a system or application [18].

When the user wants to use an application, he or she expects some functionality supported by the application. These functions are decomposed into smaller actions, described as user goals and then implemented in the application. This implementation should provide user native, transparent and quick way to reach the desired goal.

4.1 General use case overview

An application supporting editing and creation of a pixel art animation has some specific functional requirements. The best way to find out what these requirements are is to simulate workflow of a regular pixel art animation creation.

At the beginning, user has to have opportunity to open an existing animation file or create a new one. If user wants to create a new one, he or she will have to think of dimensions of an animation and set these dimension to a program. After that, user has to think of how many frames the animation will have. Larger number of frames equals to longer, more fluent animation but also to more work to do. User has to have opportunity to add or delete frames during an editing process. User might also want to change an order of frames. Another important feature is an ability to copy and paste a part or whole frame to another frame. That rapidly decreases the creation time.

During the editing process, user has to color each frame or use already mentioned copy/paste feature. Generally user chooses one frame to edit and colors one or group of pixels. User has to pick a color or mix his own. User might also want to create a color palette or use automatically generated palette consisting of recent or frequent used colors to speed up the coloring process. Colors consists of four channels - red, green, blue and alpha channel[1]) which determines opacity of a color. User might want to change the thickness of a brush so he can color a group of pixels at the same time or draw a geometric shape. Once user chooses a color and painting tool, he or she uses a computer mouse pointer or another input device such as graphic tablet to apply color. During this coloring phase, user sometimes do unwanted actions so undo and redo features become clearly necessary. Another, maybe sometimes handy, feature could by possibility of frame rotation.

Once all frames of the animation are painted and ready to be saved, user wants to set animation properties. These properties might be frames per second speed, which determines how many frames will be displayed in one second, or generally speed delay between frames. User might also want to set how many times would the animation repeat. It could be anything from one repetition to infinity loop. At the end, user could select format of an output file supporting animation.

[1]) `http://en.wikipedia.org/wiki/Alpha_compositing`

Here is the main use case "Create/edit GIF image" decomposed into smaller actions:

- Project creation
  - open existing animation
  - set new project dimensions
  - set new project number of frames
  - add new frame
  - delete frame
  - move frame
  - copy selected part of frame
  - paste selected part of frame
  - copy whole frame
  - paste whole frame

- Painting process
  - select color from pre-prepared colors
  - create own color based on setting ARGB channels
  - add color to a color palette/select color from recent used colors
  - change painting tool
  - draw geometric shape
  - change brush thickness
  - undo/redo painting actions
  - rotate chosen frame

- Animation properties
  - set frames per second rate
  - set delay between frames
  - set how many times should the set of frames repeat
  - set output file format

4.2 Choice of relevant functions

Choosing relevant functions for the *GifPix Maker* is not easy because motor-impaired people do not want to use a complex program but on the other side we want to provide them as many features as they might need. The choice of functions is based on the fact that as an input will be only a few signals (2-4). This should handle interaction design described in Chapter 5 but it generally limits implementation of some functions. The choice is also influenced by determining which functions are essential and which are not so important.

In the first stage – project creation – user chooses to open an existing project or to create a new one. This is essential and must be implemented in the application. The next step is setting new project dimensions. Because user can operate with only few signals and generally with low frequency of signal repetition, which significantly slows down movement in the picture, project dimensions must be limited to maximum of 64 x 64 pixels. Number of frames can be theoretically unlimited but because of the same reason creating more than 50 frames is not supported.

A new frame addition or deleting existing frame are also essential functions which must be supported in any program for animation creation. In order to decrease number of buttons and commands in the *GifPix Maker*, changing frames position is not directly

supported but can be simulated by creating a new frame on the target position and copying the source frame content to the new one. Then, the original frame can be deleted. As it was already mentioned, the application supports copy/paste feature but again to simplify whole program supports only copying of a whole frame.

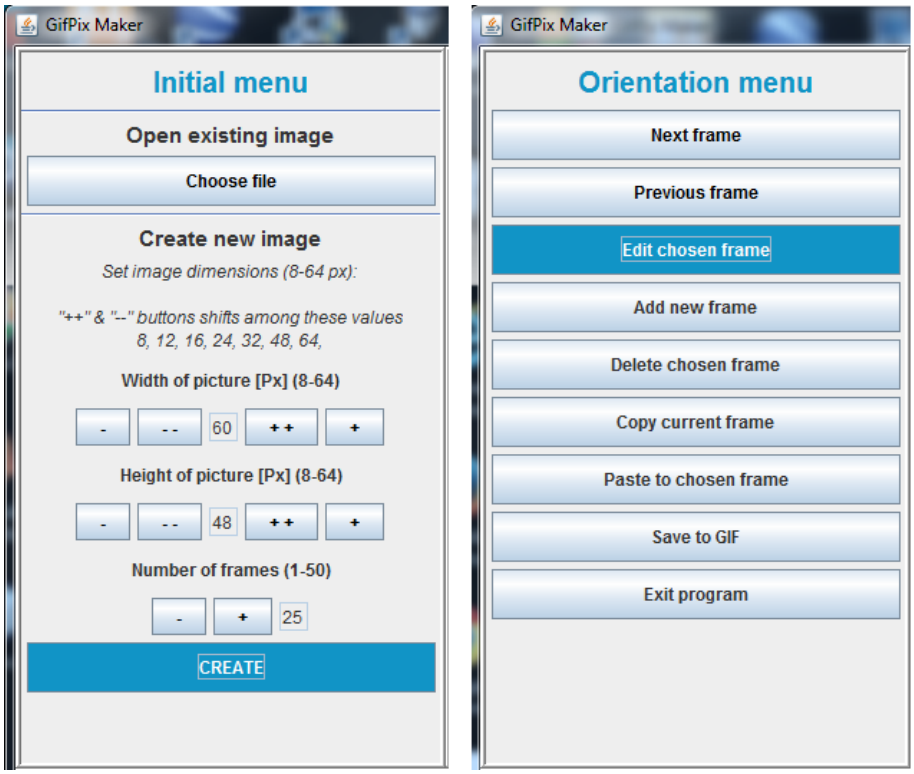


Figure 4.1. Screenshot of the Initial and Orientation menus of the GifPix Maker where are set new project properties and functions for frame management.

The painting process has to be rapidly simplified. Because user can not use their hands, use of a computer mouse or another pointing device has to be handled in a different way. The brush movement has to be implemented as a movement of a square on X and Y axes separately. How the movement of a brush works is closely described in Chapter 5.2. Because the image frame is relatively small, another tools like drawing geometric shapes is unnecessary and not supported. The color management has to be simplified too. Because of the nature of an multi-layer application environment (changing between the Drawing and Default control mode), enabling color palette feature – also called the history of used colors – was almost essential. Simplifying color management consist in the limitation of pre-prepared colors and impossibility to manage the alpha channel (pixel opacity). The set of pre-prepared colors was based on the set of colors considered as a default in the Microsoft Paint application[1]). The last feature discussed in the previous Chapter in the painting process stage is the undo/redo feature. This is more than essential in the application controlled by EMG signals because input signals are sometimes unwanted and not as reliable as in case of use for example a computer mouse.

[1]) `http://windows.microsoft.com/en-us/windows7/products/features/paint`

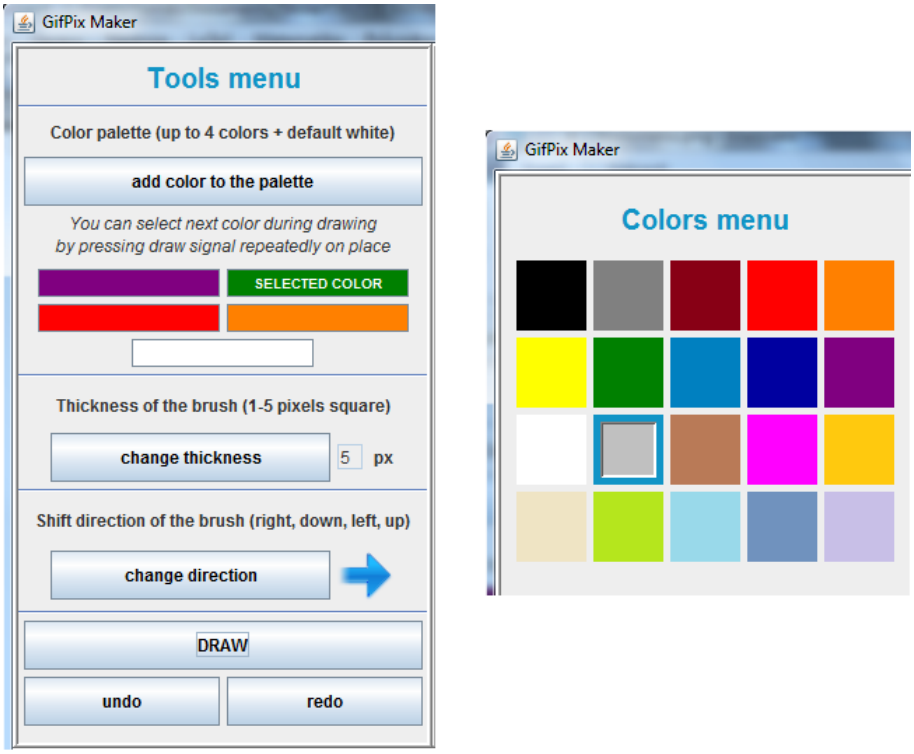


Figure 4.2. Screenshot of Tools and Colors menus of the GifPix Maker where are set painting properties.

Because the *GifPix Maker* enables only a simple pixel art animation creation, none of features enabling further animation properties setting are supported. This setting would require another dialog with lots of number settings which is really difficult to control with only a few input signals. Animation properties can be theoretically changed in the program but for our purpose are set to default values. The delay between frames is set to 0.2 seconds and the set of frames repetition is set to infinity.

Here is a summary of the general use case "Create/edit GIF image" decomposed into smaller actions adjusted for the *GifPix Maker*:

- Project creation
  - open existing animation
  - set new project dimensions (8-64 pixels)
  - set new project number of frames (1-50)
  - add new frame
  - delete frame
  - copy whole frame
  - paste whole frame

- Painting process
  - select color from pre-prepared colors
  - add color to a color palette/select color from recent used colors
  - change brush thickness
  - change brush movements direction
  - undo/redo painting actions

Chapter 5
Application interaction design

In this Chapter is described how the interaction design was developed. The interaction design of the *GifPix Maker* was based on functions chosen in Chapter 4.2. More important factor was that the application will be controlled by motor-impaired people and that there will be limited number of input signals. That requires to create an application environment with limited number of buttons and multi-layer model of the menu system.

5.1 Input signal adaptation

To implement several cases of an interaction, the *GifPix Maker* enables 3 modes of control. All modes use EMG signals but they differs in number of input signals. The interaction environment of the application is adapted to the number of input signals which is set after the application start. The start up dialog shown in Figure 5.1 enables to set a number of operable input signals and whether the application should connect to the MYO device closely described in Chapter 5.5.

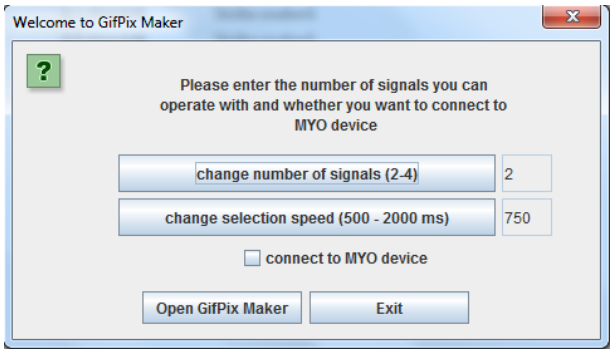


Figure 5.1. Screenshot of the start up dialog of the GifPix Maker where the number of input signals is set.

Lets see how the interaction environment of the *GifPix Maker* changes according to the number of input signals:

- **2 input signals are available**
  The worst scenario is when the user can operate with only two signals. In the start up dialog has to be also set the selection speed. The application generates selection signal in loop with specified delay in milliseconds. Every element is highlighted for that period of time. This method is generally called the scanning method [19]. It provides sequential access to elements of a graphical user interface by activating another signal (in our case the confirmation signal) when the element receives the scanning focus.

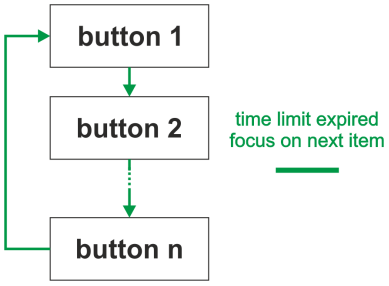


Figure 5.2. Diagram of the scanning method in the GifPix Maker where the selection signal is generated automatically.

- **3 or 4 input signals are available**

  Better scenario is when the user can operate with three or four signals. In this case is no need to generate the selection signal – user moves in menus on his own. The environment of the application was originally designed for three input signals and therefore there is no need of an additional adaptation.

  The 4^{th} signal enhances the application control in the Drawing mode (focus is on the drawing panel with the brush). It enables to change the brush movement direction without leaving the Drawing mode which speeds up the drawing process.

After the number of input signals is set in the start up dialog, the application informs user about what signals he or she can use in dialog shown in Figure 5.3. It also describes how signals work and which key on keyboard represents them. This enables to control the *GifPix Maker* without the MYO device and therefore every user can use or just try this application.

Figure 5.3. Screenshot of the information dialog of the GifPix Maker where are described functions of all input signals (2 signals available).

5.2 Application environment

The *GifPix Maker*'s GUI is designed for the sequential access. Menus, where all buttons for the application control are placed, are in a few layers. User moves among these layers with the confirmation and back signal. The selection signal is used only to move in one layer (menu or drawing panel). The movement among layers is based on logical sorting. The first menu is the Initial menu shown in Figure 4.1. As it was mentioned earlier, this

menu provides possibility to open existing project or to set properties and create the new one. Once this menu is left by clicking on the Create button, user can not go back to this menu. This reduces complexity of the environment during the editing phase. Then, whole menu environment consists of only 3 layers – the Orientation menu, Tools menu and Colors menu. This solution enables to have easy-to-remember environment with still rich functionality. The Orientation menu shown in Figure 4.1 provides possibility to exit the program and to manage frames of an animation. The Edit frame button opens the Tools menu shown in Figure 4.2. There are functions for drawing and enables to move one level lower to the next menu – the Colors menu shown also in Figure 4.2 – or to switch to the Drawing mode. The back signal enables to move one level upper in menu hierarchy or to leave the Drawing mode. Screenshot of the whole application can be seen in Figure 5.4.

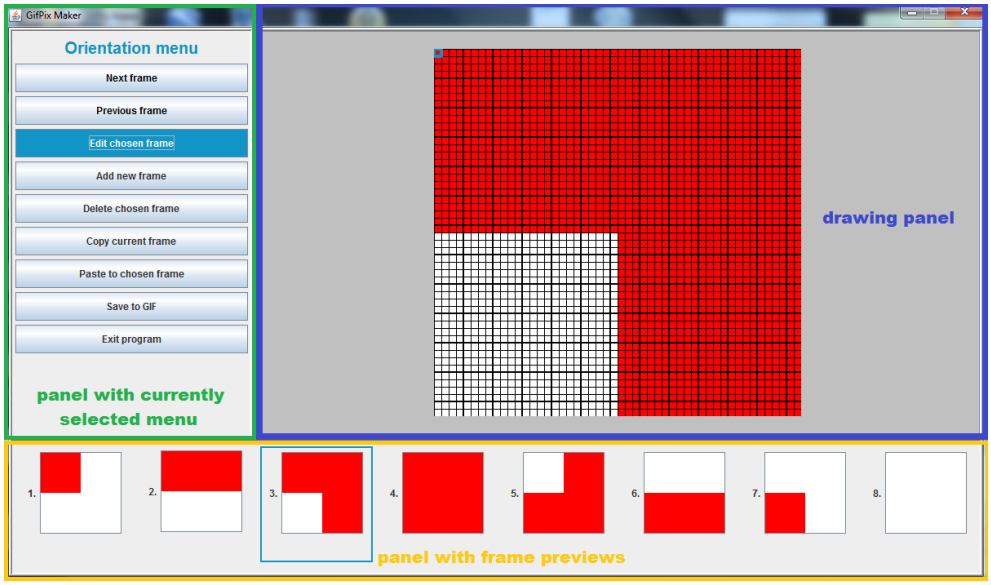


Figure 5.4. Screenshot of the the GifPix Maker's environment with description.

As it was mentioned earlier, there are two modes of control. The **Default control mode** and **Drawing mode**. The Default control mode is when the focus is on menus. The selection signal moves focus among buttons and the confirmation signal presses them. The back signal moves to the upper menu in the menu menu hierarchy. After the Draw button in the Tools menu is pressed, the application switches to the Drawing mode. The focus is on the drawing panel and signals are used to move with the brush and to apply color. The drawing panel contains magnified selected frame for editing with grid lines. These lines helps user to orient oneself in the image. Each box represents one pixel and every 4^{th} grid line is thicker. The brush is displayed as a blue thick square. Dimensions of the square vary according to the set value of the brush thickness. In this mode, the selection signal moves the brush in the set direction. The confirmation signal applies color on the brush position. By pressing the confirmation signals repeatedly, user can switch among colors in the color palette. Which colors are in the palette and which color is currently selected is displayed in the Tools menu in section with the color palette. This feature enables to use several colors at the same time without leaving the Drawing mode. The back signal is used to leave the Drawing mode. The focus is then set to the Change direction button. This reduces number of signals to be send in need of change the brush movement direction. The last signal which is supported by the application is the additional signal. This signal is used only in the Drawing mode and

it enables to change brush movement direction directly from the Drawing mode. This feature saves a lot of time during the drawing process. A preview of modes, menus and signals is displayed in the state diagram shown in Figure 5.5.

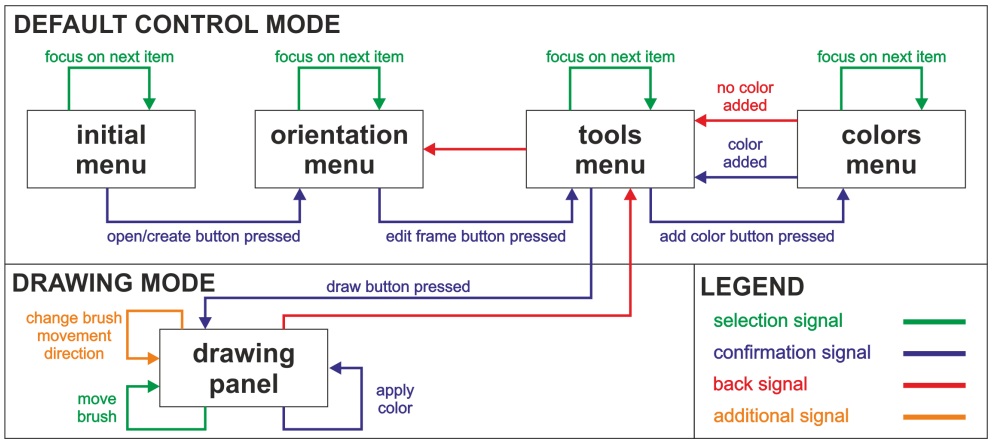


Figure 5.5. Diagram of an interaction in the GifPix Maker where input signal actions are shown.

5.3 Workflow that we support

The *GifPix Maker* provides several improvements supporting the workflow of an animation creation process. These improvements mainly reduces number of signals needed to complete desired action.

Firstly, user can open an existing GIF image. This image has to be smaller than the maximum defined by the application. Images of different types, with higher resolution than 64 x 64 pixels or with more than 50 frames will not be opened. If user chooses to create a new project, he or she has to set image dimensions. Because of wide range of values (8-64), there are extra buttons enabling faster movement among these values. Buttons with label of double plus or minus moves among preset frequently used values (8, 12, 16, 24, 32, 48, 64). After the width and height of an image is set, user has to set an initial number of frames. That could be any number from 1 to 50. This value can be incremented and decremented only by 1. Special buttons are not needed as it was in case of dimensions because the number of frames can be adjusted in the Orientation menu by adding or deleting frame.

Once project is set up, user can move among frames and see their magnified preview on the drawing panel. Frames can be additionally added or deleted. User can also copy whole frame and paste its content to another frame. Naturally, user can edit frame which opens the Tools menu and enables user to draw. User can select color from pre-prepared colors and add it to the color palette. The color palette can have 1-4 colors plus the white color which is placed in the palette by default. User can also set thickness of the brush (1-5 pixels square) and one of the four basic directions of the brush movement. After user sets these values, he or she can change to the Drawing mode by pressing the Draw button in the Tools menu. This changes focus to the drawing panel and user can move and apply colors from the palette. User can also undo or redo drawing actions. After the user ends with frames editing, he or she can finally save the image or exit the program. The resulting image is saved to the GIF file named by the current system date and time.

5.4 Overview of functions

The *GifPix Maker*'s implementation provides these functions:

- control with 2,3 or 4 EMG signals or with keys on keyboard
- in case of 2 available signals use scanning method and set the selection signal delay
- open existing GIF image with dimensions 8-64 pixels and with 1-50 frames
- create new GIF image with dimensions 8-64 pixels and with 1-50 frames
- add/delete frame
- copy/paste frame content
- manage color palette
- set brush thickness 1-5 pixels square
- change brush movement direction (in case of 4 available signals directly from the Drawing mode)
- undo/redo drawing actions
- save GIF image

5.5 Mapping low-channel input

In this part is briefly described how EMG signals are detected, mapped and used in the *GifPix Maker*. The detection of these signals is a non invasive method of recording muscle contractions. The main advantage of this method is that almost any muscle on the body (especially on the limbs) can be used for detection. Signals are then recognized by the specialized hardware and software and send to the *GifPix Maker*.

In our case, signals were detected from the extensor muscles on the forearm. These muscles are located near the elbow as it is shown in Figure 5.6. This setup enables custom made MYO device [20] to recognize and report the movement of two fingers (typically the index finger and the little finger) and the palm. It is important to mention that the setup can be different – for example electrodes can be placed on both forearms and device can detect movement of the left hand, right hand palm or both palms at the same time. This setup also gives 3 operable signals.

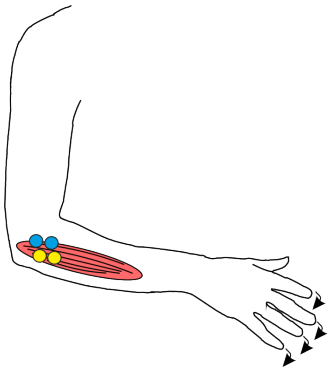


Figure 5.6. Placement of the electrodes on the forearm [16].

The acquisition of signals is performed also by using MYO device which is portable and connected to the computer over USB port. For this purpose was created a driver containing recognition algorithm. Signals are further sent to the advanced editing application[1]). This application is designed as a server sending recognized signals to the

[1]) Author of this application and MYO device is Ing. Antotnín Pošusta.

client which is connected to it. The *GifPix Maker* connects as a client using TCP/IP protocol to this application. Signals are sent in agreed format and further decoded in the *GifPix Maker*.

5.5.1 Mapping in the GifPix Maker

When the *GifPix Maker* starts, the connection to the MYO device software over TCP/IP protocol is established. Then, the application waits for incoming data in agreed format and sends them to the class handling the signal decoding. The *GifPix Maker* provides an interface which can be implemented to decode any input data. The interface provides default methods for sending control signals. These methods can not be overridden but has to be used in the `decode()` method. It is on a programmer to implement the `decode()` method and determine which signal was received and send appropriate control signal to the application.

This method can be implemented in many ways. For example even one signal input can be used. Implementation could be written to recognize how long the signal was and determine for example short and long signal. By implementing decoding interface can be this application adapted and used with different devices and combinations of input signals.

In the *GifPix Maker*, control signals are generally transformed to clicks of keys on the keyboard. This enables to use the application even with regular keyboard keys. Keystrokes are caught by key event dispatchers which performs actions according to the control signals. How the mapping is handled in code inside the application is closely described in Chapter 6.

Chapter 6
Implementation

This chapter presents how the *GifPix Maker* is implemented. There are presented used technologies and interesting parts of code. There are also explanations how and why are part of the application implemented in such way. Source files and the documentation to the application can be found on the CD attached at the end of this work.

6.1 Requirements

As it was already mentioned, the application should fit some defined criteria. It should be a compromise between desired functions and minimalist design. Also, the application should be ready for further edition and improvements.

The application requirements are divided to the functional and non-functional. Functional requirements are based on the choice of relevant functions mentioned in Chapter 4.2. Non-functional requirements are based mainly on the target group of users – motor-impaired people.

- Functional requirements – the application will be able to
  - be controlled by different number of input signals
  - open existing GIF file for editation
  - create new project
  - manage number of frames
  - draw with several frequently used colors
  - undo/redo drawing actions
  - copy/paste frame content to another
  - save project to GIF file

- Non-functional requirements – the application will be
  - accessible
  - platform-independent
  - minimalist
  - easy to edit and improve
  - stable

6.2 Used technologies

- The application was written in **Java** language. This enables to run this application on any computer independently on the platform. Java has to be in version 8 or higher because of default methods used in the definition of the interface.

- In the application were also used 2 external classes for a GIF image encoding and decoding.

- The `AnimatedGifEncoder` class enables to encode a GIF image file consisting of one or more frames. This class also enables set animation properties such as an animation speed in frames per second or how many time the animation should repeat.
- The `GifDecoder` class enables to decode a GIF image file into one or more frames. These frames are then transformed to `int[][]` arrays and saved into a Model.

6.3 Used design patterns

To simplify the application source code some design patterns were used. These patterns hopefully helps to clarify what is happening inside the application and makes it easier to edit.

Model-View-Controller

Whole application uses a MVC pattern. Basic principal is that a View (generally GUI) provides user a graphical interpretation. User makes an action in the View and this action is passed to a Controller. The controller represents the application logic. The controller modifies a Model which represents data with which the application is working. The controller updates the View according to the changed Model and waits for the next user action.

In the *GifPix Maker* the View is the GUI of the application consisting mainly of panels with buttons. A composition of the View package is shown in Figure 6.1. To buttons are assigned `ActionListener`s which represent the Controller – the application logic. According to the event, the `ActionListener` changes the Model which represents raw data of a GIF image.

Singleton

The application core – also called the Model – is represented by two classes. The first is the `GifImageStats` class which contains frame image representations in two-dimensional arrays. The second one is the `PaintingStats` class which contains information about current painting properties. Both classes are designed as a singleton classes. These classes are meant to be instanced just once and then only modified. This guarantees that when the Controller asks model for data, it always returns relevant and up-to-date data.

6.4 Raw data implementation

A class containing raw image data is called `GifImageStats`. In this class, images are represented by two-dimensional `int[width][height]` arrays. Each array field represents one pixel of an image with the `Integer` value of a color in RGB format. These images are ordered in an `ArrayList<int[][]>` according to their position in a GIF image. A drawing action is implemented as a simple pixel coloring on specified positions in the image array. Actions such as add/delete frame are implemented as a manipulation of these arrays. Copy/paste actions are handled as a deep copy of specified array so the clipboard is independent on the source frame. This enables to paste once copied frame to several frames even after the source frame is deleted. This class also contains information about calculated magnification ratio for image previews in the panel of frame miniatures and for the frame displayed in the drawing panel.

The second class containing data about the painting state is called `PaintingStats`. In this class are held information about the current position of the brush, brush thickness and brush movement direction. It also contains information about the selected color and holds current history of used colors – the color palette composition. In this class is also held the painting history for currently edited frame. This history is represented again by an `ArrayList<int[][]>` containing deep copies of an image painting process. This history enables to undo/redo painting actions. Once user goes back in the painting history and then performs a new drawing action, the history is deleted from this point to the and replaced by currently drawn image.

6.5 GUI implementation

The GUI of the *GifPix Maker* consists mainly of a system of `JPanels` nested into each other in the `MainWindow JFrame`. This frame is divided into 3 main panels which is shown in Figure 5.4. How classes are composed together is in Figure 6.1.

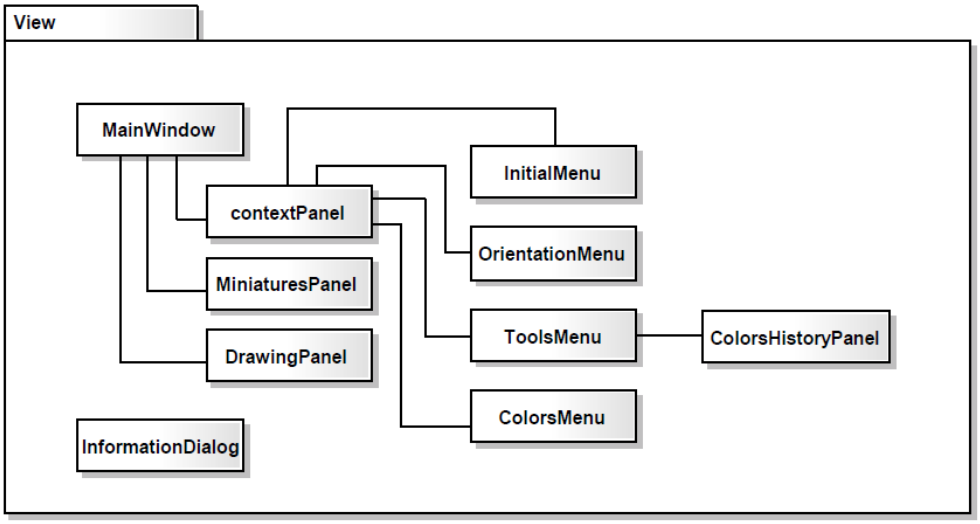


Figure 6.1. Diagram of the View package showing class composition.

Context panel

On this panel is displayed currently selected menu. Panel has `CardLayout` which enables simple switching among these menu panels. Each menu consists of a set of `JButtons` and other GUI primitives such as `JLabels` or `JTextFields`. These buttons are the only one that can gain focus by pressing the selection signal. Each button has unique functionality. These functions are implemented by `ActionListeners` assigned to the buttons.

Miniatures panel

On this panel are displayed small numbered previews of all frames in the GIF image. This panel has `GridLayout` with one row where a horizontal scroll bar appears if it is needed. The currently selected frame is highlighted with a blue border and always visible in the panel by setting visible rectangle calculated from the selected frame position. Miniatures are created by using a buffer. The image saved in the `int[][]` array is converted to a `BufferedImage` and then rescaled by calculated magnification ratio for the miniatures preview.

Drawing panel

A currently selected frame is displayed on this panel. This preview is also created by using buffer. The image saved in the `int[][]` array is converted to a `BufferedImage` and then rescaled by calculated magnification ratio for the preview of the selected frame. After image is rescaled, grid lines are added to this image with respect to the magnification ratio and also the brush represented by a rectangle is added. After every color application is this preview and the miniature preview updated and redrawn.

6.6 Input signal adaptation

After the application starts, instance of a `SignalReciever` is created. This class handles connection to the MYO device software. If connection could not be established the application displays an information dialog and closes itself. The `SignalReciever` is designed as an independent thread which is started during the main window initialization. This class creates a new client socket and opens buffered input stream reader. Data obtained from the reader are sent to the class implementing the `ControllingInterface` which provides the `decode()` method.

```
public interface ControllingInterface<E> {

    void decode(E data);

    default void fireConfirmationSignal() {
        Robot robot = new Robot();
        robot.keyPress(KeyEvent.VK_SPACE);
        robot.keyRelease(KeyEvent.VK_SPACE);
    }

    default void fireBackSignal() {
        Robot robot = new Robot();
        robot.keyPress(KeyEvent.VK_BACK_SPACE);
        robot.keyRelease(KeyEvent.VK_BACK_SPACE);
    }

    default void fireSelectionSignal() {
        Robot robot = new Robot();
        robot.keyPress(KeyEvent.VK_TAB);
        robot.keyRelease(KeyEvent.VK_TAB);
    }

    default void fireAdditionalSignal() {
        Robot robot = new Robot();
        robot.keyPress(KeyEvent.VK_SHIFT);
        robot.keyRelease(KeyEvent.VK_SHIFT);
    }
}
```

The `ControllingInterface` class contains 5 methods. Four of them are default which means that are available in all classes implementing this interface and can not be overridden. These methods serves for firing selection/confirm/back/additional signals. These methods are meant to be used in the `decode()` method which has to be implemented by user. This interface can be implemented with any data type. This data type should be the same as the data type in which data are received.

In our case the class implementing this interface is called `SignalDecoder`. Input data is sent as `String` lines which are decoded in the `decode()` method and according to which signal is received is called one of defined methods.

6.7 Input signal interpretation

Signals fired by default methods in the `SignalDecoder` are simulated keystrokes on the keyboard. These keystrokes are generated by using the `Robot` class which is available in Java. This class enables to simulate key press and key release which simulates keystroke of any keyboard key. These keystrokes are further handled by `KeyEventDispathers` which are assigned to the `KeyboardFocusManager` of the main window.

```
private static KeyboardFocusManager kfm;
kfm = KeyboardFocusManager.getCurrentKeyboardFocusManager();
```

A keyboard focus manager can contain many dispatchers at the same time. Generally, every `KeyEventDispatcher` has `dispatchKeyEvent()` method which return `boolean` value. If the returned value is `true`, `KeyEvent` is not passed to another dispatcher. Otherwise, the `KeyEvent` is passed to other dispatchers which can do something else.

In the *GifPix Maker* are used two modes of control. These two modes are in code handled by changing the `KeyEventDispatcher` of the main window. This change is made by the `DispactherChanger` class that determines which is the current mode and to which mode the manager has to change. The `DispatcherChanger` calls one of these methods provided by the `MainWindow` class.

```
public static void changeToDrawingDispatcher() {
    kfm.removeKeyEventDispatcher(defaultDispatcher);
    kfm.addKeyEventDispatcher(drawingDispatcher);
}

public static void changeToDefaultDispatcher() {
    kfm.removeKeyEventDispatcher(drawingDispatcher);
    kfm.addKeyEventDispatcher(defaultDispatcher);
}
```

Each of the mentioned dispatchers detects keystroke events and performs appropriate action.

- `DefaultKeyEventDispatcher`

It handles only the back signal which moves to the upper menu in the menu hierarchy and returns `false`. This means that keystrokes are passed to other key event dispatcher which are present in the `KeyboardFocusManager` by default. These default dispatchers for example presses the button when it has focus while the SPACE key is pressed. It also moves focus to the next button in the application GUI hierarchy while the TAB key is pressed.

- `DrawingKeyEventDispatcher`

It handles all signals and returns `true`. This means that keystrokes are not passed to other key event dispatchers, even not to them which are present in the `KeyboardFocusManager` by default. This dispatcher is activated when the focus is on the drawing panel. When the selection signal is received, the dispatcher checks bounds and moves the brush to the next position. When the confirmation signal is received,

the dispatcher applies given color to the brush position and saves copy of an image to the painting history. When the back signal is received, the dispatcher calls the `DispatcherChanger` and the application changes to the Default control mode. When the additional (4^{th}) signal is received, the dispatcher changes the brush movement direction.

6.7.1 2-signal input

When the application is started in mode with only 2 input signals, the application starts the Scanning method. This method is implemented by the `AutomaticSelector` class. This class is designed as an independent thread which is started during the main window initialization. This class generates a keystroke of the TAB key on the keyboard with given delay in milliseconds.

```
public class AutomaticSelector extends Thread {
    private Robot robot;
    private volatile boolean active;
    private final int selectionSpeed;

    public AutomaticSelector(int selectionSpeed) {
        this.selectionSpeed = selectionSpeed;
        active = true;
        try {
            robot = new Robot();
        } catch (AWTException ex) {
            System.err.println("Robot initialization failed.");
        }
    }
    @Override
    public synchronized void run() {
        while (active) {

            //delay
            try {
                Thread.sleep(selectionSpeed);
            } catch (InterruptedException ex) {
                System.err.println("Thread interrupted.");
            }

            //stopping checkpoint
            if (!active) {
                break;
            }

            //keypress
            robot.keyPress(KeyEvent.VK_TAB);
            robot.keyRelease(KeyEvent.VK_TAB);

        }
    }
    public void kill(){
        active = false;
    }
}
```

This thread can be stopped by the `kill()` method which turns running condition to `false`. This condition is checked at the beginning and in the middle to stop the thread immediately independently on the executed line to prevent unwanted key press.

The main problem of an independent thread generating keystrokes is that it is still running even after the application is minimized. This problem has to be solved so this thread will not continue if the application window loses focus. Java enables to add `WindowFocusListener` to a `JFrame`. The `WindowAdapter` provides methods called when the window loses or gains focus that can be overridden to suit our needs. The same trick is used for dialogues that opens when some action has to be confirmed.

```
this.addWindowFocusListener(new WindowAdapter() {
    @Override
    public void windowGainedFocus(WindowEvent e) {
        restartAutomaticSelector();
    }

    @Override
    public void windowLostFocus(WindowEvent e) {
        stopAutomaticSelector();
    }
});
```

Chapter 7
Testing

In this Chapter is described the testing of the *GifPix Maker*. It contains information about the setup of the testing, tested use cases, testing process description and summary of findings. Usability testing is necessary to reveal shortcomings of current implementation. Photos and video from usability testing can be found on the CD attached at the end of this work.

7.1 Setup

Testing took place on May 12^{th}, 2014 at the Academy of Sciences of the Czech Republic. There were four able-bodied participants involved in the test. Each participant spent one hour in the lab. After the participant's arrival, he or she was seated behind an office desk. On the desk was a monitor with the original image model displayed on it. On the desk was also a notebook with the Windows 7 OS and *GifPix Maker* ready for start. Then, participant was informed about the nature of the test and what is expected of him or her. After that, electrodes were attached on the participant's forehand followed by a quick calibration of input signals. Participant could choose between two ways of the electrode attachment. They could control the program by lifting whole palm on both hands or by lifting fingers and palm on one hand. The setup time was approximately 5 minutes. Participant's hands rested on the office desk during the test. Photo taken in the lab during testing showing the setup is in Figure 7.1.

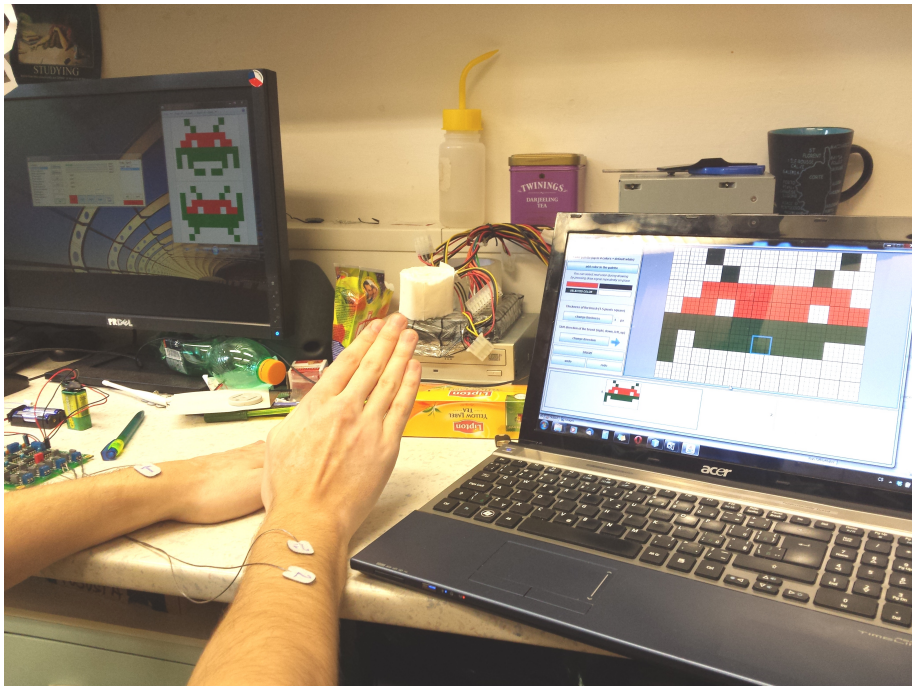


Figure 7.1. Photo taken during testing showing setup in the lab.

7.2 Participants

Sadly, test had to run with healthy able-bodied participants instead of the target group. All participants were students in age between 20 and 30 years. Two of them were men and two of them were women. All participants have never seen the tested program before so they could not prepare for the measurement.

The **first participant** was a woman with no experience with EMG signals detection. She was familiar with the nature of the animated GIF creation. She has chosen the placement of the electrodes on both hands. She could use three signals by lifting right, left or both palms at the same time. The anatomy of her hand caused problems with calibration due to weak muscle contractions and high signal noise of unknown source.

The **second participant** was a man with little experience with EMG signals detection. He was also familiar with the nature of an animated GIF creation. He has chosen the placement of the electrodes on both hands. He could use three signals by lifting right, left or both palms at the same time.

The **third participant** was a man with experience with EMG signals detection. He was also familiar with the nature of an animated GIF creation. He has chosen the placement of the electrodes on one hand. He could use three signals by lifting the index finger, little finger and palm.

The **fourth participant** was a woman with no experience with EMG signals detection. She was not so familiar with the nature of an animated GIF creation but quick briefing cleared that topic for her. She has chosen the placement of the electrodes on both hands. She could use three signals by lifting right, left or both palms at the same time.

7.3 Choice of use case

Use cases were chosen mainly to explore as many functions as it is possible. The importance was focused on the overall impression of the application on the user. Originally several tasks were prepared but due to the lack of time and higher time-consumption was used only the first task.

The task was to replicate an original simple animation consisting of two frames shown in Figure 7.2. The model image was displayed on the monitor during whole test. Participant was told to create a new project with specified dimensions and to draw the image as big as possible.

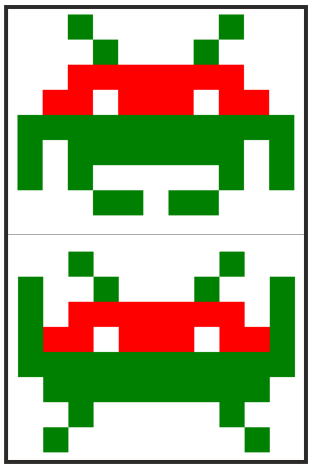


Figure 7.2. Decomposed original animation used as a model for testing.

This use case tested almost every function in the program. At the beginning, participant had to set dimensions to the width of 44 pixels and height of the 32 pixels. That required to use all changing value buttons in the Initial menu. During the drawing phase, participant had to move among frames and also use copy/paste buttons. Participant also had to add colors to the palette, change the brush thickness and movement direction and draw. The EMG signal detection is not as reliable as the keyboard or mouse input so participant had to also use undo/redo buttons. At the end, participant had to save the animation and exit the program.

7.4 Testing process

Every participant had approximately one hour for the calibration and task itself. After the placement of the electrodes, participant was familiarized with the test process. The calibration consisted of the slight relocation of the electrodes and threshold setting on the side of the MYO device. On the side of the *GifPix Maker* calibration consisted of the adjustment of the signal decoding interface according to the participant needs. Once the participant felt ready to start, the *GifPix Maker* was started in the desired mode.

Lets see how each participant dealt with the task. This decomposition shows some shortcomings of the program discovered during testing.

The **first participant** had problem with calibration. Muscle contractions were so weak that the sensitivity of the MYO device had to be set higher that usual. This caused that the device received a lot unwanted signal noise. Due to bad calibration, participant had quite big problem with unwanted signals. She has spent the most time by setting image dimension in the Initial menu. Eventually, she has been able to draw part of the first frame but due to the time limitation she has not finished the task. During the task she used the color palette by adding red and green color to it. Problem was that she found out accidentally that she could switch among these colors by pressing drawing signal on place. She has not finished the first frame due to the lack of remaining time.

The **second participant** had no problem with calibration. While he was trying to set initial dimension values he tried to use back signal to set focus on the first button in the menu. He praised the possibility of setting common values quickly by special buttons. While he was setting brush properties in the Tools menu he expected that the change of the brush thickness would be visible in real time on the drawing panel. While he was drawing the first frame he praised the possibility of changing the brush color by pressing the drawing signal repeatedly on place. After the participant successfully finished the first frame (in approximately 20 minutes) he was asked to try to draw the second frame with 2 input signals. The application was restarted in mode for 2 input signals and participant had to draw the second frame. This method was useful but participant stated that the control of the application was much better in mode with 3 input signals. He had mainly problem with pressing the desired button before the focus changed to the next one. He has not finished the second frame due to the lack of remaining time.

The **third participant** had no problem with calibration. He was the only one who has chosen to use only one hand. He has already used the EMG signal detection before so he has had minimum error rate. He was the only participant who has finished task successfully and has drawn the image almost exactly according to the given model image. He was also the fastest participant. He has finished the task (both frames) in approximately 30 minutes. He also tried to draw the same image with only 2 input

signals. Due to higher accuracy and better signal detection participant had no problem with the scanning method. He stated that both input methods were usable but using 3 input signals was more comfortable.

The **fourth participant** had no problem with calibration. She successfully set the initial values and created a new project. During the drawing phase she also used the drawing palette by adding red and green color but she found the possibility to change color during drawing accidentally. Due to pretty high error rate she would appreciate UNDO for the last signal, not just for the last drawing action. She stated that it would save her a lot of time. Due to the lack of time she has been able to finish just the first frame.

All participants stated that it was easy to orient oneself in the application and that they have no general problem with functionality. All of them found it usable and they enjoyed completing the task. Frames drawn by participants are shown in Figure 7.3.

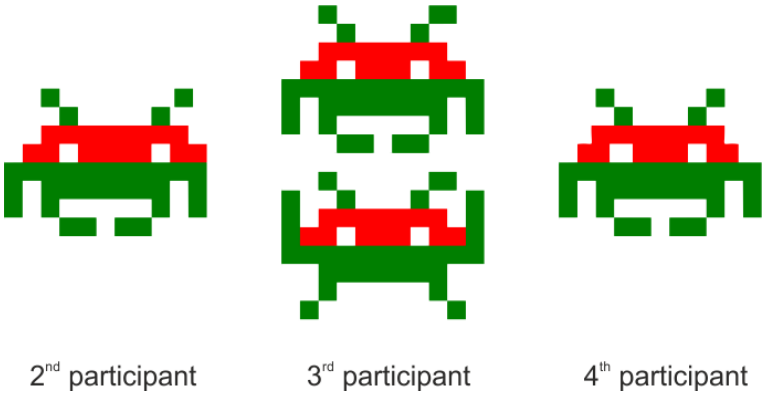


Figure 7.3. Images created by participants during testing by using the *GifPix Maker*.

7.5 Overview of findings

Testing has shown that the application works with the MYO device and can be controlled by EMG signals. It has also shown that the application is usable and easy to understand. On the other hand, testing has also revealed a few shortcomings of the *GifPix Maker*.

The main problem is reliability of the EMG signal detection. The detection of these signals is far from perfect and its reliability differs from person to person. If we compare the first participant, who has had no experience with it, and the third participant, who has already worked with the EMG signal detection, we can see that also training helps to improve reliability of this control method. Unwanted multiple detection of one signal is problem that can partially solve proper implementation of the decoding interface in the *GifPix Maker*. But it can not filter other unwanted signals so to solve this problem it is necessary to improve the signal detection on the side of the MYO device.

Other problems are just shortcomings of the application. Summary of the main problems noticed by participants is shown in Table 7.1. Every problem has assigned its priority. Priorities are defined as follows:

- **1 – high priority** Problem makes program impossible to use or does not behave as the user would expect. This problem has to be repaired as soon as possible.
- **2 – medium priority** Problem disturbs the user from using program and may deter some users. This problem has to be repaired immediately after the problems with the highest priority.

- **3 – low priority** Problem causes that the user may be confused by using this program but does not affect significantly its usability or functionality. This problem has to be repaired after more important problems are repaired.

It is important to say that there was no finding categorized as Priority 1. The *GifPix Maker* was stable and all participants found it usable.

priority	problem description	solution design
2	user would appreciate UNDO for last control signal (mostly unwanted)	add control signals history with undo/redo feature
3	user did not notice or understand message that informs how to switch colors in color palette during drawing	change the information message or add an animation demonstrating this feature
3	back signal does not have any function in the Initial and Orientation menu, user expected to use it to return focus to the first button in menu	add back signal function in the Initial and Orientation menu to return focus on the first button
3	change of the brush thickness is not visible until the user enters drawing mode	change the size of the brush square on the drawing panel immediately according to the changing value

Table 7.1. Overview of findings revealed by the testing with users.

Chapter 8
Conclusion

The main goal was to provide a tool for graphic content creation for motor-impaired people. That was achieved by designing and implementing the *GifPix Maker* which provides a simple pixel art animation creation. This program is designed to be primary controlled by EMG signals. Requirements stated in Chapter 1.5 were an accessibility, easy of use, minimalist design and adaptation to different input signals.

This program is **accessible**. It is designed to react on the natural keys on keyboard used for an interaction with GUI with buttons. Those keys are for example TAB, SPACE and BACKSPACE. The program could by controlled even only by 2 keys on keyboard or other input signals. Other input signals are transferred to keystrokes of already mentioned keys which provides a freedom in the input signal adaptation.

This program is **minimalist** and **easy-to-use**. Its GUI composes of only a few panels containing menus and image previews. Its design is clear and simple without any unnecessary functions and buttons which could increase the program complexity. Testing showed that users have no problem with orientation in the application and everything was exactly where they expected would be.

This program is **adaptable**. It provides the interface enabling adaptability of any input signals. The program is originally designed to be controlled by EMG signals. It provides adaptation for 2 to 4 input signals. Testing showed that the program could be controlled easily not just in mode with 3 input signals available but even in mode with 2 input signals available. The adaptation of input signals was used even during testing due to different user detection sensitivity and hand anatomy.

The *GifPix Maker* meets all requirements stated at the beginning of this work and represents an EMG-controlled tool for image editing.

8.1 Future work

Testing revealed some problems of the current implementation. These problems are summarized in Table 7.1 with the solution design. Once these problems are solved there should be other usability testing with improved EMG signal detection and target group of participants. The *GifPix Maker* application could be improved in many ways. One way is to improve functionality by adding new functions such as creating own color by setting RGB values, possibility to zoom in the image or to adjust animation properties. Another way of an improvement could be considering different modalities as an input such as voice recognition which could make the application accessible even to wider spectrum of users.

References

[1] Robert M Kaplan. Quality of life measurement. *Measurement strategies in health psychology*, pages 115–146, 1985.

[2] M.S. Green. *Self-Expression*. Oxford scholarship online. OUP Oxford, 2007.

[3] Claire Wallace and Florian Pichler. More participation, happier society? a comparative study of civil society and the quality of life. *Social Indicators Research*, vol. 93(issue 2):255–274, 2009.

[4] Simeon Keates, Faustina Hwang, Patrick Langdon, P. John Clarkson, and Peter Robinson. Cursor measures for motion-impaired computer users. In *Proceedings of the Fifth International ACM Conference on Assistive Technologies*, Assets '02, pages 135–142, New York, NY, USA, 2002. ACM.

[5] Eric Bergman and Earl Johnson. Towards accessible human-computer interaction. *Advances in human-computer interaction*, 5(1), 1995.

[6] Andrew Sears and Mark Young. The human-computer interaction handbook. chapter Physical Disabilities and Computing Technologies: An Analysis of Impairments, pages 482–503. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003.

[7] Susumu Harada, Jacob O. Wobbrock, and James A. Landay. Voicedraw. *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility - Assets '07*, pages 27–34, 2007.

[8] Susumu Harada, T. Scott Saponas, and James A. Landay. Voicepen. *Proceedings of the ninth international conference on Multimodal interfaces - ICMI '07*, pages 178–185, 2007.

[9] Robert JK Jacob and Keith S Karn. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. *Mind*, 2(3):4, 2003.

[10] Andrew T Duchowski. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers*, 34(4):455–470, 2002.

[11] Alex Poole and Linden J Ball. Eye tracking in hci and usability research. *Encyclopedia of Human-Computer Interaction, C. Ghaoui (ed.)*, 2006.

[12] Melissa Dawe. Desperately seeking simplicity: How young adults with cognitive disabilities and their families adopt assistive technologies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1143–1152, New York, NY, USA, 2006. ACM.

[13] Anthony J. Hornof and Anna Cavender. Eyedraw: Enabling children with severe motor impairments to draw with their eyes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 161–170, New York, NY, USA, 2005. ACM.

[14] Anthony Hornof, Anna Cavender, and Rob Hoselton. Eyedraw: A system for drawing pictures with eye movements. *SIGACCESS Access. Comput.*, (77-78):86–93, September 2003.

[15] Torsten Felzer and Bernd Freisleben. Hawcos. *Proceedings of the fifth international ACM conference on Assistive technologies - Assets '02*, pages 127–134, 2002.

[16] Antonín Pošusta, Ondřej Poláček, Adam J. Sporka, Tomáš Flek, and Jakub Otáhal. Text entry methods controlled by myoelectric signals. *Accepted for publication in Bulletin of Applied Mechanics.*

[17] Mordy Golding. Developing a graphic design workflow. In *Sams Teach Yourself Adobe Creative Suite All in One.* Sams, 2004.

[18] Kurt Bittner. *Use Case Modeling.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[19] Stavroula Ntoa, George Margetis, Margherita Antona, and Constantine Stephanidis. Scanning-based interaction techniques for motor impaired users.

[20] Jakub Otáhal and Antonín Pošusta. Recording and conditioning of surface emg signal for decomposition. *Bulletin of Applied Mechanics*, 8(30), 2012.

Appendix A
Abbreviations

To simplify the text, some abbreviations were used. Here can be found the whole list.

HCI	Human-computer interaction
EMG	Electromyography
GIF	Graphics interchange format
AT	Assistive technology
GUI	Graphical user interface

Appendix B
CD Content

B.1 Software

- **Java** in version 8
- **GifPix Maker** – executable JAR file
- **GifPix Maker** – source code in Java with documentation in javadoc

B.2 Text of this thesis

- **PDF** version of this thesis
- **TEX** source code of this thesis

B.3 Other files

- **Video** taken during usability testing
- **Photos** taken during usability testing