VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

BRAIN SIGNALS VISUALIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE IVAN ŠEVČÍK
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VIZUALIZACE ELEKTRICKÝCH SIGNÁLŮ MOZKU
BRAIN SIGNALS VISUALIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE IVAN ŠEVČÍK
AUTHOR

VEDOUCÍ PRÁCE Ing. MICHAL KOŠÍK
SUPERVISOR

BRNO 2015

Abstrakt

Tato práce pojednáva o problematice zpracování a vizualizace mozkových signálů a jejich využití pri analýze dat měřených pomocí EEG. Práce obsahuje úvod do anatomie mozku a povahy mozkové aktivity, který je pak využit při popisu jedné z metod meření této aktivity, konkrétně EEG. Dále se práce zabýva metodami zpracování signálů, především filtrací, a metodami vizualizace. Rovněž je zde uveden přehled předchozích rešení zabývajících se vizualizací biosignálů, které sloužily jako výchozí představa při tvorbě konceptu vlastní aplikace. Tato aplikace byla napsána v jazyce C++ s využitím knihovny Qt pro vytvoření multiplatformního grafického uživatelského rozhraní. Důležitým aspektem bylo také využití rozhraní OpenGL, které umožnilo hardwarově akcelerovat vykreslování a tím pádem dosáhnout uspokojivé překreslovací frekvence. Dále bylo využito knihoven, ktére implementují načtení EEG dat ze souboru, rychlou Fourierovou transformaci, matematické operace v 2D a 3D prostoru, a načítání 3D objektů ze souboru. Implementace pak řeší několik problémů. V první řade je to vykreslování 2D a 3D modelů mozku a vizualizace mozkové aktivity nad těmito modely formou animovaných elektrod. Za tímto účelem byl stažen model skutečného lidského mozku, který byl následně optimalizován v grafickém 3D editoru. S využitím tohto modelu a modelu lebky byly určeny 3D pozice elektrod podle standardního pozičního systému 10-10 využívaného při EEG měřeních. 2D pozice byly definovány tak, aby co nejvíc odpovídali obrázkům uváděným ve vědeckých článcích. Tyto pozice je možné předefinovat pro využití s jinými než standardními systémy. Elektrody jsou na těchto pozicích vykresleny jako kruhy vždy natočené k uživateli a animovaný jsou změnou barvy, která je odvozena od průběhu signálu. Aplikace dále obsahuje prohlížeč mozkové aktivity ve formě vícenásobného čárového grafu. U něj bylo využito decimace signálu metodou max-min, která poskytuje výrazné zrychlení překreslování zredukovaním počtu vzorků signálu při zachování jeho charakteristické křivky. Aplikace podporuje filtraci signálů pomocí dolní a horní propusti a dvojici váhovacích oken – Hammingovo a Blackmanovo. Implementace této části byla testována porovnáním výsledků různých konfigurací s výstupem programu MATLAB, ve kterém byli vytvořeny shodné filtry. Výsledná aplikace představuje nástroj pro analýzu mozkových signálů kombinující prohlížení signálů s prostorovými vizualizacemi. Prostorová vizualizace poskytuje způsob pro zkoumání průběhu všech signálů v čase a umožňuje snadnou vizuální identifikaci mozkové aktivity. Ta může být podrobněji analyzována právě pomocí prohlížeče signálů. Aplikace byla testována pod operačními systémy Windows 7 a 8, pro které je primárně určena, ale byla také úspěšne přeložena, sestavena a spuštena pod operačním systémem Linux. Aplikace nepodporuje řadu žádaných funkcí, např. nahrávaní a zpracování signálů v reálnem čase. Tyto nedostatky spolu s navrhovanými řešeními jsou uvedena v závěru práce. Současna implementace však poskytuje vhodný základ pro plánovaný budoucí vývoj.

Abstract

EEG is a non-invasive method for measuring brain activity, which gives an important insight into mental processes. This work presents an application that serves as a visual EEG data analysis tool. The application concept is based on the existing solutions, which were analyzed as a part of this thesis. The application combines signal browser in form of a view composed of multiple charts with 2D and 3D visualizations that are helpful in recognition of particular brain activity. In addition, it implements convenient signal processing methods. The thesis is concluded by a summarization of accomplishments and future work's directions.

Klíčová slova

lidský mozek, elektroencefalografie, zpracování signálu, vizualizace dat, OpenGL, 3D model mozku

Keywords

human brain, electroencephalography, signal processing, data visualization, OpenGL, 3D brain model

Citace

Ivan Ševčík: Brain Signals Visualization, bakalářská práce, Brno, FIT VUT v Brně, 2015

Brain Signals Visualization

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Košíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

....
Ivan Ševčík
May 19, 2015

Poděkování

Týmto by som sa rád poďakoval svojmu vedúcemu Ing. Michalovi Košíkovi za odbornú pomoc a cenné rady pri vypracovaní bakalárskej práce.

Contents

Chapter 1

Introduction

A brain is a central organ of a human nervous system and as such it has a very important role in almost every human activity. When performing conscious and subconscious tasks, the brain exhibits an electrical activity caused by communication between neurons – elementary units of a neural tissue [1]. The complexity of the brain makes it difficult to study and understand. This is further complicated by the fact that a person must be alive and conscious during many trials.

Because of that, the brain activity can provide a valuable insight into mental processes and brain functions. The activity can be measured using multiple approaches. This thesis is interested in *electroencephalography* (EEG), a non-invasive method that measures the activity from a scalp using electrodes and additional electronic equipment.

EEG has many uses. It can be used to diagnose patients and represents an important tool for studying human brain. Many brain-computer interfaces (BCI) are also based on EEG. These are devices that provide direct communication pathway between the brain and an external device, typically personal computer. BCIs can be used to provide severely impaired patients a way to interact with their surroundings [7], but they can be also used by healthy persons to augment cognitive or mental functions, for example to control a device without any motor movement. [17].

Data measured by modern EEG are usually just signal amplitude values in digital form, which are hard for humans to interpret. Another issue is that, generally, users are not interested in raw values but in some features that the EEG signal carries, such as intensity at certain frequency or specific patterns that represent a certain activity performed by a measured subject [10].

This creates a need for applications capable of presenting measured data interactively. There are already existing applications for this purpose, some of which are analyzed as a part of this thesis. However, these applications are often incomplete and either offer only basic visualizations, miss any methods of signal processing, or have performance issues with large datasets. This requires a user to use two or more applications and switch between them.

The goal of this work is to create a single, modern application that could serve as an EEG data analysis tool. The application will implement a chosen set of data processing methods, including signal filtering and classification by frequency. More importantly, the application will provide visualization methods, such as charts showing waveform of measured signal in time domain, and both a two and a three-dimensional models displaying brain activity. Another task will be to create an intuitive and clean, yet customizable graphical user interface that will be responsive for real-time signal inputs. However, the application should be able to process also large offline data measured earlier without noticeable performance decrease.

The text is structured into chapters that start as a general theory and progressively get more specific, concluded by an application concept and implementation.

Chapter 2 offers an introduction to a human brain and methods for measuring its activity using EEG.

Moving on, Chapter 3 provides fundamentals of signal processing and computer visualization relevant for this thesis.

Chapter 4 contains reviews of several existing solutions for biosignals visualization, which serve as an inspiration for this work.

The concept for the application is introduced in Chapter 5 and eventually transformed into the implementation, which is described in Chapter 6.

Chapter 2

Human Brain

A human brain is a central unit of a human nervous system and has an important role in almost every aspect of life. It performs both conscious tasks and an automatic operation of vital organs, including breathing, maintaining blood pressure, and releasing hormones. It also processes all the sensory inputs, such as odors, sounds, or light, interprets them, and makes associations with representations preserved in memory.

2.1 Brain Biology

A biology of the human brain is complex and many mental processes are still not well understood. This section will, therefore, provide rather a very brief introduction to brain anatomy and communication mechanisms that are relevant for this work.

2.1.1 Anatomy

From an anatomy point of view, this work is interested primarily in a cerebrum because it is the part of the brain where an EEG activity occurs. The cerebrum is the outermost structure of the brain. It is divided into two separate hemispheres that are bridged by a bundle of fibers. The cerebrum is associated with a high-order functioning and a control of a voluntary behavior. This includes thinking, perceiving, planning, and language understanding. The cerebrum is covered with a sheet of a tissue called cerebral cortex, which is further divided into regions called lobes that are functionally differentiated. There are four lobes, each with a different role [1]:

Frontal lobe is where initiation and coordination of motor movements take place, along with higher cognitive skills, such as thinking, planning, and organizing. It is also important for a personality and an emotional makeup.

Parietal lobe is responsible for processing sensory inputs, attention, language and spatial orientation.

Occipital lobe helps to process visual information and to assist with a recognition of shapes and colors.

Temporal lobe is a region where auditory information is processed and information from other senses is integrated. It also has a role in the short-term memory and in learned emotional responses.

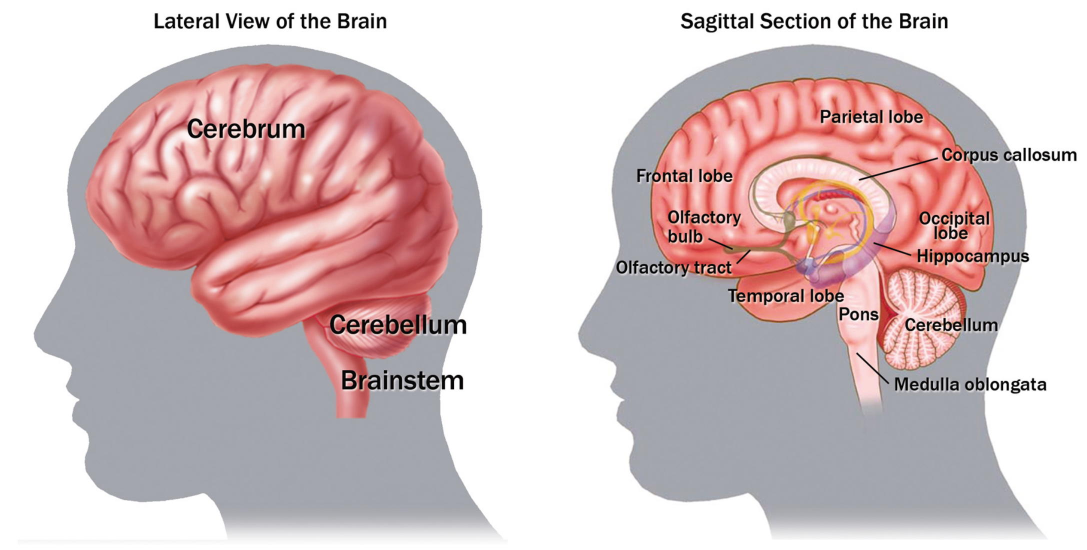


Figure 2.1: Brain anatomy[1]

Figure 2.1 shows all the discussed structures on a model of the brain.

2.1.2 Neurons

The neural tissue is made of cells called neurons that communicate with each other and represent basic working units of the brain. The neuron consists of a cell body, dendrites and an axon, which can be seen in Figure 2.2. The cell body contains a nucleus and

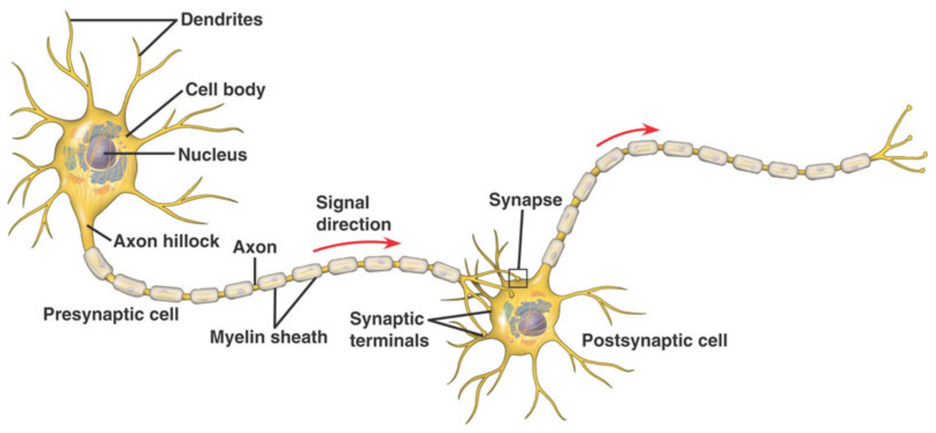


Figure 2.2: Neuron structure[2]

a cytoplasm, and produces peptides required for communication. Dendrites extend from the cell body and receive signals from other neurons through synapses – contact points where axons of other neurons connect to dendrite. The axon is responsible for transmitting

[1]Image source http://www.compelvisuals.com/wp-content/uploads/2013/12/BrainAnatomy-Newsletter.jpg

[2]Image source http://2012books.lardbucket.org/books/beginning-psychology/section_07/6a3f0732c22683476ea201ffc5e428ad.jpg

electrical impulses outward from the neuron. These electrical impulses originate from a sudden change of an electrical potential caused by a flow of ions through a cell membrane. The change, also called an action potential, then passes along axon's membrane and upon hitting its end releases neurotransmitters in nerve terminals. The neurotransmitters diffuse across the synapse and bind to receptors on the surface of a target cell's dendrite, which causes response in the target [1].

2.2 Electroencephalography

As has been presented, the nature of the human nervous system is electrical. When many neurons are active at the same time, the collective charge of ions is large enough to propagate as a wave to the surface of a brain or even to a scalp. Therefore, it is possible to observe a brain activity in underlying brain structures by measuring the variation of the surface electrical potential on the scalp. This is done by placing a conductive electrodes on the scalp and taking a measurement of the potential. The method just described is known as *electroencephalography* (EEG) and it is a noninvasive way to monitor and study human brain. Earliest works used galvanometers and sophisticated techniques to take measurements [3], which has been since substituted by A/D converters and computers or other equipment for recording digital data. The recorded data can be then subjected to further *digital signal processing* (DSP) [10].

The main advantage of EEG over other methods for monitoring brain activity, namely MRI[3] and PET[4], is speed as it can record responses to a stimulus within fractions of a second [18]. Applications of EEG include:

- Research – monitoring of brain activity during cognitive and motoric tasks in high time resolution
- Medical – diagnosis of brain diseases or confirmation of brain death
- Human computer interaction – execution of commands using brain activity

Recently, the EEG based devices were introduced also to a consumer market so users can monitor their brain activity at home, play simple games, or control toys using their thoughts[5].

2.2.1 Basic Principles

The neural activity has mostly asynchronous nature. Because of this, it is impossible to measure or distinguish electrical activity of each neuron with EEG as the individual potentials are combined by the time they propagate to the scalp. What EEG measures is a summated potential of a neural activity, which includes a spontaneous electrical activity of the cerebral network and cortical responses to external and internal events. The responses to events, characterized by their onset latency and voltage amplitude, are referred to as *event-related potentials* and are either results of physical stimuli or behavioral responses [7].

[3] *Magnetic resonance imaging* is using magnetic fields and radio waves to form images of the body.
[4] *Positron emission tomography* is using radioactive tracers to map functional processes in the body.
[5] http://www.livescience.com/43250-mind-controlled-quadcopter.html

2.2.2 Equipment for Taking EEG Measurements

The setup used for encephalographic measurements usually consists of:

- Electrodes with conductive media
- Amplifiers with filters
- A/D converter
- Recording device

Electrode Caps – Electrodes exist in various forms but this thesis will focus on headsets and electrode caps, such as one displayed in Figure 2.3. These caps usually consist of small discs serving as electrodes made of a very conductive material, such as gold, silver, or stainless steel. The electrodes have long leads that can be plugged into an amplifier [18]. The discs are in a direct contact with the scalp and can already measure the electric potential produced by the brain. However, a conductive media in form of a gel or a paste is often used to increase conductivity even more by lowering contact impedance and improve readings at the lowest level. It also helps electrodes to stick to the scalp surface so accidental shift in a position is less likely.

There is also another type of electrodes that doesn't require application of gel. These are called dry electrodes and they use additional electronics to lower skin impedance, such as local impedance converting amplifier. Because of this, systems using dry electrodes are more bulky and more expensive but they don't require skin preparation and usage of gels that cause skin irritation [16].

The electrodes are prepositioned on a silicon cap in locations according to standardized placement system, which will be discussed later in this section. This both speeds up the setup stage and allows for unified mapping of electrodes to head surface. However, it fails to account for different shape and features of each individual's head [18]. This problem is usually solved by involving a method for finding 3D coordinates of electrodes on the head. Such methods include using magnetic field digitizer or elaborate algorithms that can calculate position of each electrode from few distance and angle measurements by utilizing specific properties of standard placement system [8].

Amplifiers – The strength of the signal measured by electrodes is in order of microvolts and normally range from $10\,\mu V$ to $500\,\mu V$ [19]. Therefore, an amplifier is needed to bring the amplitude to higher voltage levels that can be processed by other electrical circuits and components. The requirements on amplifiers for use with EEG are high as they have to provide amplification only for the physiological signal, which should not be distorted in any way, reject superimposed noise and interference signals, and protect both the equipment and measured subject from current surges. Amplifiers conforming to these requirements are known as *biopotential amplifiers.* [9]

The trend in a development of neural recording devices is heading in a direction of small and portable systems that can be used by patients with severe disorders in everyday life. Consequently, energy efficient amplifiers are being designed lately that are very small, may run on battery for long periods of time and dissipate only little heat [19].

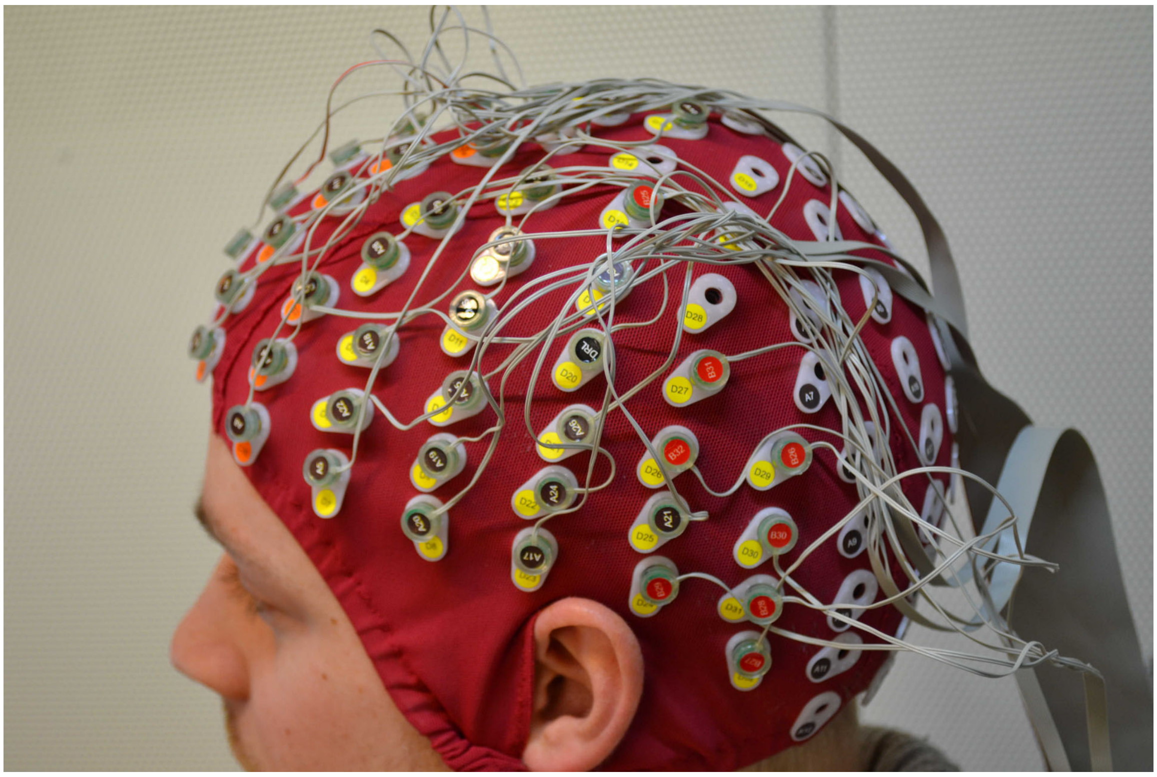


Figure 2.3: Electrode cap[6]

Filters – An analog band pass filter is used to limit frequencies into a certain range of interest after a signal amplification. In some cases it may be already a part of the amplifier. Another common filter is a notch filter, which can used to filter out noise at frequency of a power line. Depending on a country, it may be set to 50 or 60 Hz [2]. Such filter is required only if it is desirable to keep also high frequencies. Otherwise, a low pass filter would be sufficient because the interesting frequency bands usually lies bellow the frequency of the power line.

A/D Converter – An A/D converter unit is used to convert filtered analog values to digital representation. The converter should have a resolution of at least 12 bits and ideally 16 bits or more. With a high number of electrodes, analog multiplexers are sometimes used to lower the number of necessary converter units at the expense of a limited sampling frequency. The sampling frequency should be at least double of the highest recorded frequency, for example the upper frequency limit of a band pass filter. This is known as Nyquist-Shannon sampling theorem which is fundamental for correct signal reconstruction without aliasing artifact. The preferred sampling frequency is 256 Hz to 400 Hz [11].

Recording Device – Converted samples from A/D converters are stored in a digital memory for further processing which will be discussed later in this section 2.2.4. A recording device may be represented by a computer or by a different piece of specialized equipment. The computer can be additionally equipped with digital signal processor unit so that CPU load is lower and can be used for other tasks.

[6]Image source `http://www.psychologie.uzh.ch/fachrichtungen/plasti/Labor/\gls{eeg}-03.jpg`

2.2.3 Standard Electrode Placement

The need for a standardized electrode placement was evident as early as 1947 when the first International EEG congress held place in London. Various methods of standardization were proposed and this effort finally resulted in definition of a 10-20 electrode system in 1958 by H.H. Jasper [12]. This system consists of 21 electrodes placed evenly between certain landmarks and the labels of electrodes are derived from cerebrum lobes above which the electrodes are located. The first contour can be found by connecting inion and nassion. Inion is a small protuberance at the back of a head and nassion is located just above the bridge of a nose. The length of this line is measured and electrodes are placed along it using following procedure. First electrode – Fpz, is placed at 10% of the contour length from the nassion. The electrodes Fz, Cz, and Pz are placed in this order from front to back with spacing defined as 20% of the length. Fz and Fpz are also spaced by 20%. Last electrode, Oz, is placed at 10% of length from the inion which equals 90% of length from the nassion. It should be clear now why the system is called 10-20. The rest of electrodes are placed in similar fashion.

The 10-20 system offers only limited resolution but it was sufficient at the time of its creation because the main limiting factor was technology. However, with advances in electrode technology and miniaturization of electrodes, using more than 21 channels became feasible. Naturally, many laboratories sought to take advantage of the higher resolution. With more electrodes than what is 10-20 model capable of accommodating, first extension to the standard placement system was needed. The extension, named 10-10 system, was proposed in 1985 and extended the number of supported electrodes to 74. It uses additional coronal contours lying halfway between original contours and combines the labels of these original contours to create new label. For example, a contour lying between original contours F and C is labeled FC [12]. The positions of electrodes according to this system are shown in Figure 2.4.

The 10-5 electrode placement system further extends the 10-10 system to allow an even higher number of channels to be used as systems with over 128 channels are not uncommon anymore and have become commercially available[7]. The total number of possible locations supported by 10-5 system is 345 [12]. This system is in a proposition stage and is yet to be recognized as a standard.

2.2.4 Data Analysis

The recorded data are subject to further processing using numerical filters and transforms in order to improve the signal-to-noise ratio and extract desired signal features. One such feature is brain activity at a certain frequency. The frequency of brain waves range from 1 Hz to 150 Hz. This range can be further divided into six common categories [5]:

- Delta waves – below 4 Hz
- Theta waves – between 4 and 8 Hz
- Alpha / Mu waves – between 8 and 13 Hz
- Beta waves – between 13 and 30 Hz
- Gamma waves – between 30 and 80 Hz

[7]http://www.egi.com/company

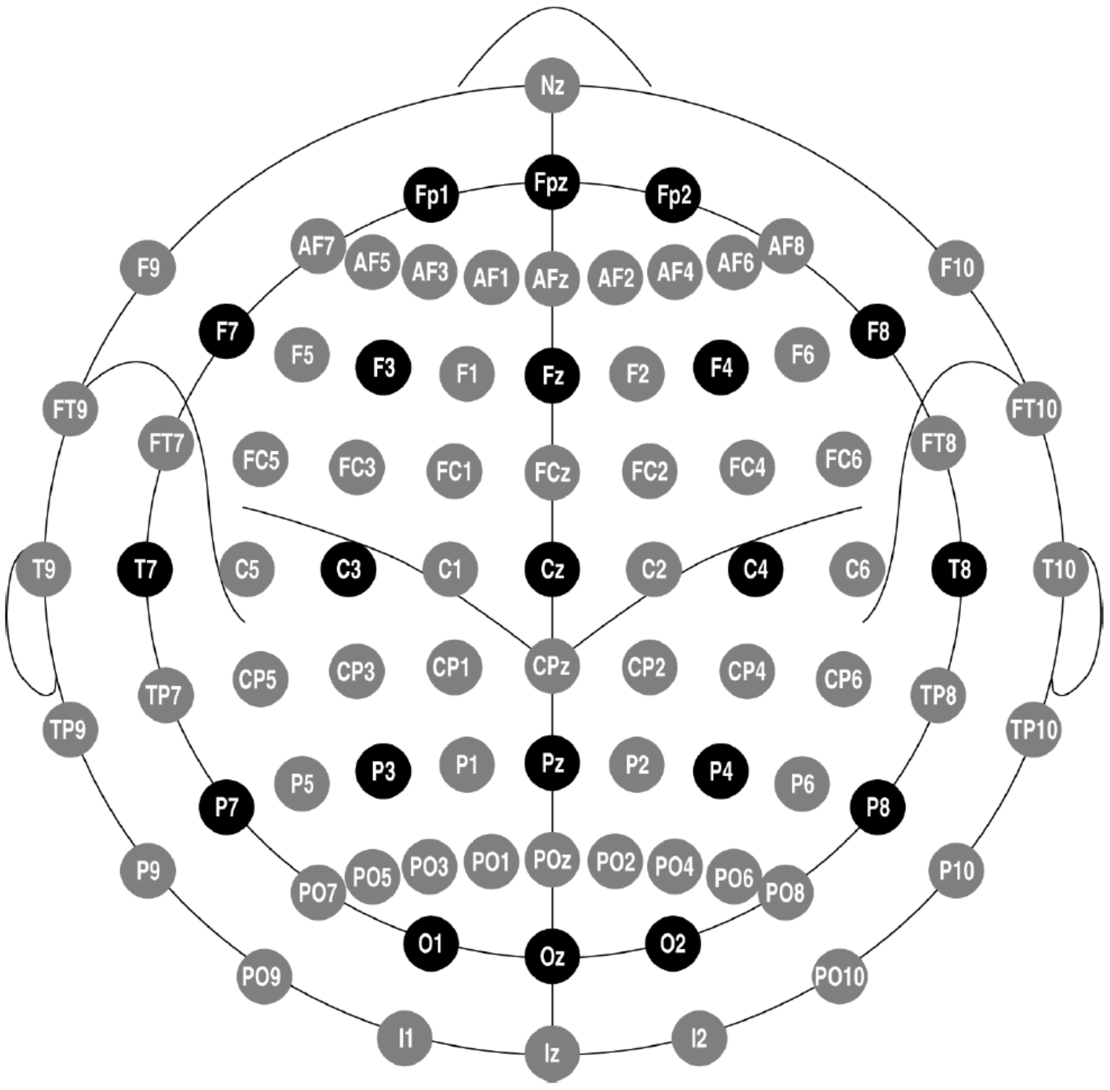


Figure 2.4: International 10-20(black) and 10-10(gray) system of EEG sensor placement[12]

- High gamma waves – above 80 Hz

A healthy person exhibits brain waves mostly in the first four categories. Each band is related to a certain group of an activity in the brain [5]. Delta waves are normally distributed over scalp and may be significant during deep sleep, in childhood, or in serious organic brain disease [10]. Theta waves are related to cognitive tasks such as working memory and error monitoring tests. Alpha power is increased during inattention and lack of visual input, therefore related to perception. Mu power, on the other hand, is linked to movement and is decreased when person performs a motor action. The distinction between alpha and mu waves is made by considering the location of the electrode. While alpha waves form at the back of the scalp, mu waves can be observed in a strip of brain from ear to ear called motor cortex. Beta waves are low amplitude, high frequency waves that have power reduced at the onset of a movement, rebound if the movement is sustained, and are enhanced if the movement is suppressed. Finally, peaks in gamma and high gamma bands are observed in some brain areas but this activity varies too much across individuals to be reliably characterized [5].

Another set of features are patterns, such as those of event related potentials, that represent a certain activity more clearly than classification by frequency. Event related potentials are electrical brain responses time-locked to a physical stimuli or behavioral responses and characterized by their voltage amplitude and their latency in relation to stimulus onset [7]. For example, the imagined movement of finger can be distinguished from imagined movement of leg with this method. The classification accuracy is, however, highly individual and a classification system usually needs a tuning or a calibration process for each user [7]. Common approach to this problem is to train a neural network that can classify patterns in brain waves with accuracy high enough for normal usage.

Chapter 3

Approaches to Signal Processing and Visualization

Once the data from electrodes have been recorded they can be processed and visualized according to chosen analysis method. This chapter will provide information about methods that are commonly used for processing EEG signals. The chapter will also discuss general data visualization methods and how these methods can be performed on modern graphics card to produce a visual output in form of a raster graphics. A more specific terminology will used in the following text. The term *signal processing* will be from now on restricted to digital signal processing as the processing will be performed on software level. Likewise, the term *signal* will be restricted to a discrete-time signals represented by a sequence of numbers.

3.1 Signal Processing

In general, signal processing is concerned with representation, transformation, and manipulation of signals and the information they contain. For example, it can be used after an EEG measurement to ease analysis of recorded data by filtering out noise and selecting signal features, as presented in Section 2.2.4. This section, therefore, provides general information about methods of signal processing and specific implementation details are discussed in Section 6.2.

3.1.1 Discrete Fourier Transform

Discrete Fourier transform (DFT) is an operation that produces a corresponding frequency spectrum for any given sequence of numbers. Formally the DFT is defined by Equation 3.1 where the sequence $x(n)$ represents the input, usually a signal [15, p. 401].

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-2j\pi kn/N} \qquad 0 \leq k \leq N-1 \tag{3.1}$$

The product of the DFT is a new sequence $X(k)$ of complex numbers equally spaced in frequency domain. Each number represents a sample in the frequency spectrum. Because the numbers are equally spaced, each sample belongs to a certain frequency interval that is often referred to as a frequency bin. A bin resolution, that is the length of each frequency interval, can be determined through Equation 3.2 where f_s is the sampling frequency of the input signal.

$$bin_{res} = \frac{f_s}{N} \tag{3.2}$$

The direct computation of DFT is inefficient and contains repeating, and therefore redundant, operations. Because of that, a more efficient algorithm called *fast Fourier transform* (FFT) is used in practical applications. The FFT algorithm takes advantage of symmetry and periodicity to reduce number of operations needed to compute the result. The number of complex multiplications can be used as a criterion when comparing the efficiency of DFT and FFT. Indeed, while DFT requires N^2 multiplications, the FFT algorithm requires only $(N/2)\, log_2 N$ multiplications to produce the exactly same results. For example, the speed is improved 204.8 times for a 1024 point DFT [15, p. 459].

3.1.2 Filters in General

A filter in the broadest context is any system that modifies certain frequencies relative to others. For the purposes of this thesis the term filter can be narrowed to a system that passes certain frequency components and totally rejects others [13, p. 439]. All the discussed filters will be causal. A system is causal, if for every choice n_0, the output sequence value at the index $n = n_0$ depends only on the input sequence values where index $n \leq n_0$ [13, p. 21]. The stability of filters will be also discussed in section 3.1.3. A system is stable if and only if every bounded input sequence produces a bounded output sequence. The input $x[n]$ is bounded if there exists a fixed positive finite value B_x, such that [13, p. 21]

$$|x[n]| \leq B_x < \infty, \qquad \text{for all n.} \tag{3.3}$$

The system is stable if for each bounded input there exists a fixed positive finite value B_y, such that [13, p. 22]

$$|y[n]| \leq B_y < \infty, \qquad \text{for all n.} \tag{3.4}$$

The filtering can be performed as convolution of the input signal with filter's impulse response in time-domain. Another, equivalent method is to first perform DFT for the input signal, multiply the obtained spectrum with filter's frequency response and perform an inverse DFT.

3.1.3 FIR and IIR Filters

One of the most important decisions when considering the use of filters is the choice between finite and infinite impulse-response filters. They are two fundamentally different groups of filters with many differences, some of which will be discussed. The major difference between the systems can be seen from difference equations 3.5 and 3.6 [15, p. 500]. While the FIR system uses only current and past input values, the input to the IIR system includes also previous output values.

$$\text{FIR system} \qquad y(n) = \sum_{k=0}^{M-1} b_k x\,(n-k) \tag{3.5}$$

$$\text{IIR system} \qquad y(n) = -\sum_{k=1}^{N} a_k y\,(n-k) + \sum_{k=0}^{M} b_k x\,(n-k) \tag{3.6}$$

Another difference between the FIR and IIR system involves stability. The FIR systems are inherently stable from the definition but a great care must be taken in order to design a stable IIR system. The IIR systems also suffer from limited precision of numbers in a digital representation. Therefore, an error accumulates over time due to the feedback.

Considering the stated properties of both filter types it was chosen to further use only FIR filters as they are more suitable for EEG data processing. This claim is further supported by frequent use of FIR filters in other EEG related works, for example [2] and [10].

3.1.4 Low-pass and High-pass FIR Filters

A low-pass filter is a system that ideally passes only frequencies below a specified threshold called cutoff frequency f_c. Accordingly, a high-pass filter is a system that passes only frequencies higher than the threshold and attenuates the others [15, p. 331].

The filter is created by finding the coefficients b_k for Equation 3.5. This can be done in general by designing the desired frequency response in frequency domain and using inverse Fourier transform to obtain filter's impulse response $h(n)$ which represents the coefficients. However, the low- and high-pass filters are so common that the impulse response can be obtained using equations 3.7 and 3.8 where $\omega_c = 2\pi f_c$ and M is the filter's length [13, p. 472].

$$\text{Low-pass} \qquad h(n) = \begin{cases} \frac{\sin(w_c(n-M/2))}{\pi(n-M/2)} & ,\text{if} \quad n \neq M/2 \\ 2f_c & ,\text{if} \quad n = M/2 \end{cases} \tag{3.7}$$

$$\text{High-pass} \qquad h(n) = \begin{cases} -\frac{\sin(w_c(n-M/2))}{\pi(n-M/2)} & ,\text{if} \quad n \neq M/2 \\ 1-2f_c & ,\text{if} \quad n = M/2 \end{cases} \tag{3.8}$$

3.1.5 Window Functions

In general, the filter's impulse response obtained through inverse Fourier transform is infinite in length. Therefore, it must be truncated at some point, say at $n = M - 1$, to yield the FIR filter of length M [15, p. 624]. The truncation is equivalent to multiplying the infinite impulse response by a rectangular window which is defined by Equation 3.9.

$$w(n) = \begin{cases} 1 & 0 \leq n \leq M-1 \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

However, such truncation produces undesired effects in the filter's frequency response. For example, let's consider an FIR low-pass filter with a cutoff frequency $f_c = 0.4\,\text{Hz}$ and length $M = 31$. The left graph in Figure 3.1 depicts the frequency response of such filter after truncation. It can be seen that the gain at frequencies below cutoff moderately fluctuates and also that the attenuation at frequencies above cutoff is relatively small. The graph on the right side shows the frequency response for the exact same filter but the truncation was done using Hamming window. Clearly, the fluctuations are much smaller, almost invisible at this level of detail. Also, the filter better attenuates undesired frequencies. However, this comes at a cost of a longer transition to reach the stop band. While the filter with rectangular window reaches the stop band at frequency little over 0.4 Hz,

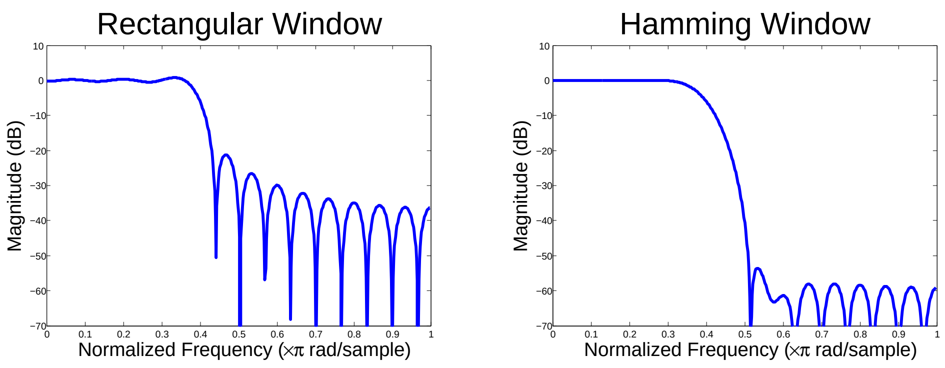


Figure 3.1: Comparison between filters with Rectangular and Hamming window

the filter using Hamming window gets to the stop band at approximately 0.5 Hz. This is because the window attenuates data at its edges. Decreasing width in the time domain increases uncertainty in the frequency domain, which is represented by a filter response that has a shallower transition band stretching farther from the cut off frequency. Therefore, a choice must be made but the advantages of using window function usually outweigh the disadvantages.

The window function in essence weighs samples entering the filter so that transition into truncated area is smooth rather than sudden as in the case of rectangular window. The Equation 3.10 defines the already discussed Hamming window and Equation 3.11 defines another commonly used Blackman window [15, p. 626]. The definitions are only for $0 \leq n \leq M-1$ as the rest of the impulse response is still truncated.

$$\text{Hamming window} \qquad w(n) = 0.54 - 0.46\cos\frac{2\pi n}{M-1} \tag{3.10}$$

$$\text{Blackman window} \qquad w(n) = 0.42 - 0.5\cos\frac{2\pi n}{M-1} + 0.08\cos\frac{4\pi n}{M-1} \tag{3.11}$$

3.2 Data Visualization Methods

So far, only methods for measuring and processing data have been discussed but the primary goal of this thesis is to visualize them. Therefore, this section will discuss methods of data visualization using PC and common display devices such as monitors and screens, and how these visualizations can be used to help humans analyze data more easily.

3.2.1 2D Representations

Display devices and other media used for presenting information have mostly two-dimensional nature. This has great implications for how the data are usually visualized and manipulated. The most natural representations are those that are already organized spatially as two-dimensional, without need for any projections and mappings. It is still possible to add a next dimension of information through usage of colors, different shapes and sizes, but the diagram layout remains two-dimensional. The most commonly used diagrams are based on graphs and charts, although schematics and maps are also popular.

The graph is a type of diagram made of interconnected objects. The connection may be represented by a line or curve which may be directed, meaning that it goes only one way. Graphs are usually used for visual representations of data where individual entities and their relations can be recognized, for example a network of roads between cities, but they are also useful for displaying hierarchies.

The chart is a visual representation of data useful for plotting, categorizing, and showing trends. It may come in many forms, like bars or lines drawn in an area designated by axes with scales, or slices of a pie. It is important to choose the most suitable one, so that the most significant fact stands out.

3.2.2 3D Models

The world and objects within it are perceived by humans as three-dimensional and, therefore, it is preferable to use three-dimensional representations of objects called 3D models. However, there is lack of technology for displaying such representations truly three-dimensionally, with holograms being the most advanced one. Because of that, the models must be projected into two dimensions so that they can be displayed using existing devices. This is done using projection matrices during process called rendering. The two most common projections are orthogonal and perspective.

There are multiple ways to create the 3D model. It is possible to scan real world objects by various means, such as tomography and optical surface imaging. Another method is to use special modeling software and reconstruct the model from basic primitives. The model may have defined volume using volumetric or voxel representations, or it can be just a surface of an object which is usually represented by a mesh. The mesh is a collection of points in 3D space called vertices which are interconnected to form basic primitives, like triangles and polygons.

3.3 Programmable Graphics Pipeline

A graphics pipeline represents the process of rendering and its stages. The input data in form of vertices, colors, and other information are passed to the first stage which processes and passes them down the pipeline to another stage. The result of the final stage is rendered image of a scene. Before a programmable pipeline was developed, programmers could only use vendor defined operations and combine them to create their application. That imposed certain constraints on what could be done, for example the number of light sources was limited. This approach was called a fixed pipeline because the action performed by each stage was fixed and could be only modified through parameters or states. With introduction of the programmable pipeline, it was possible for programmers to step into the rendering process in several places and define their own techniques. The custom programs that have become part of pipeline are called shaders. The Figure 3.2 displays an OpenGL graphics pipeline [20]. The three most important types of shaders with respect to their position in the pipeline will be presented in following text. The terminology used in this section complies with OpenGL but the basic principles are similar for most of the graphics card APIs. For example, OpenGL shader programs are written OpenGL Shading Language (GLSL) and DirectX shader programs are written in High Level Shading Language (HLSL), both of which are derivatives of C language and provide similar features. A good introductory tutorial to modern OpenGL can be found at `http://www.opengl-tutorial.org/`.

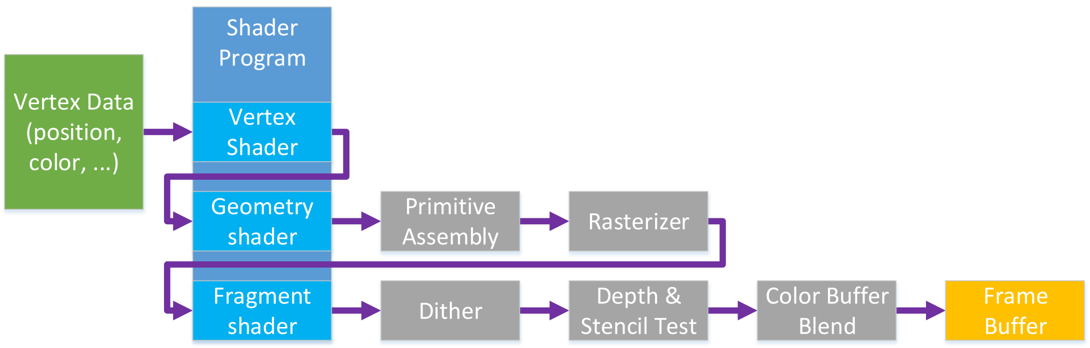


Figure 3.2: OpenGL 3.3 programmable graphics pipeline [20]

3.3.1 Vertex Shader

A vertex shader is at the very beginning of the graphics pipeline, just after the primitive processing. It allows properties of vertices to be modified before they enter the geometry shader or the primitive assembly stage. The input to vertex shader is single vertex with all the related information. Typical usage of this stage consists of applying projection and transformation matrices, performing Gouraud shading [4], and controlling particle effects.

3.3.2 Geometry Shader

A geometry shader, if present, is executed right after the vertex shader but instead of working with isolated vertices, it works with whole primitives like points, lines and triangles. It may perform transformation from one type of primitive to another and also emit completely new geometry. For example, it is possible to have as input only one point and output may be a circle with that point as a center. The circle can be made of varying number of triangles, depending on how far from camera the point was. This not only saves time-expensive transport operation between CPU and GPU, but also optimizes performance by dynamically lowering details.

3.3.3 Fragment Shader

A fragment shader works with individual pixels. However, the three-dimensional information have not been discarded yet and is accessible during processing. This allows three-dimensional effects, such as Phong shading [14], to be applied at the finest level. The output of this program is a single color that will be displayed by that specific pixel.

Chapter 4

Overview of Existing Solutions

This chapter analyzes existing solutions focusing on their user interface, features, extensibility and intuitiveness. The work of other individuals or companies can serve as a source of inspiration when creating new application with similar purpose. It is possible to identify good ideas, which can be reused, by analyzing their solution.

4.1 Emotiv

Emotiv is a company specializing in the area of EEG devices for consumer market. The product of the company is a headset accompanied with basic software and additional software can be bought or downloaded for free through a store separately. On one side, the store allows the developers to create new applications independently but as a result, there is a lack of a single, unified user experience. This is caused not only by different styles the authors of each application are using, but also because each application has its own set of settings, controls, and displays, many of which are duplicated between applications. For example, a chart with signals from electrodes is included in both TestBench and 3D brain activity map applications. There is also another brain activity map that displays each frequency band in 2D. The settings, such as gain factor and buffer size, are not shared in any way between the applications, resulting in different behavior which can be quite confusing. However, the focus on ordinary computer users can be felt as the applications are visually appealing and user-friendly. The TestBench and 3D activity map are displayed in figure 4.1.

Feature-wise, the applications are ranging from visualization software and data importers to games. One important feature that seems to be missing from most of the applications is lack of any control of time, meaning they are only able to display current values. It was only observed in the TestBench application. This is a very limiting factor for data analysis, as one of basic tasks is comparing data. Being able to do so between two or more moments in time is often desirable. The SDK for creating new applications is freely available, so anyone can start developing.

Currently, the company offers two options – 14 channels EPOC and 5 channels Insight headsets. These headsets come with built-in amplifiers, filters and A/D converters, and communicate wirelessly with PC through bluetooth.

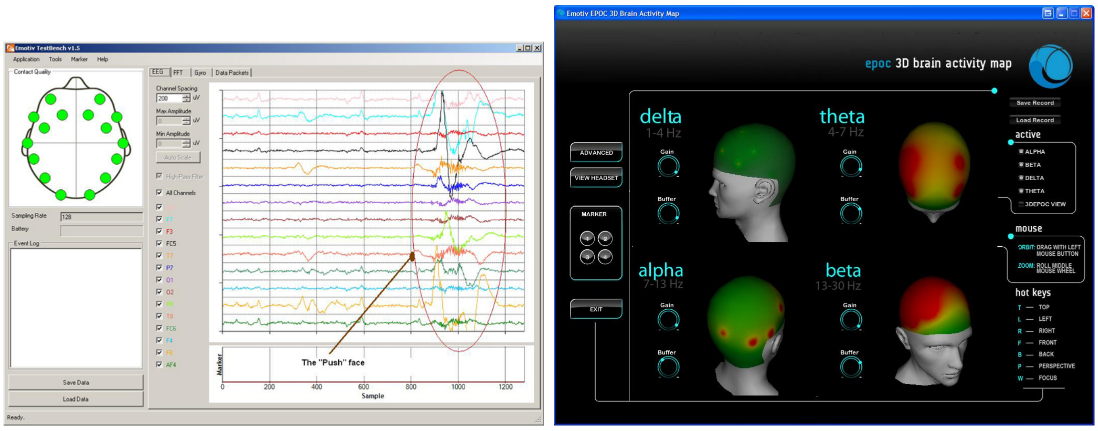


Figure 4.1: Emotiv TestBench and 3D brain activity map[1]

4.2 Svarog

Svarog is an acronym that stands for Signal Viewer, Analyzer and Recorded on GPL. It is intended for use as an advanced analysis tool capable of both online and offline signal viewing and processing. The data may be in any data format as long as SignalML description is provided.

The application offers only basic visualization methods. Main GUI element visible after starting the program is a chart with EEG waveforms. It allows a user to:

- Modify the amplitude range, time scale and amount of space between signals.
- Zoom in and out by reducing the number of displayed signals.
- Select part of a signal spanning one or multiple electrodes.
- Perform FFT and show frequency spectrum for a segment of signal in superimposed window as shown in Figure 4.2.

An interesting element is a time-frequency map that is used for presenting results of a matching pursuit algorithm, which is one of the available processing tools. Time is plotted on a horizontal axis, frequency on a vertical axis and amplitude is represented by a color in a range from blue to red. However, there is no brain or electrode map, or any other visually appealing elements.

The GUI is sometimes cluttered with too many options, which could have been hidden in some advanced view, but overall is clean, quite customizable, and informs a user about progress of operations. It also allows auxiliary signal plots to be created, which are useful for signal comparison. Svarog is implemented in Java.

[1]Image source `http://common.ziffdavisinternet.com/util_get_image/22/0,1425,sz=1&i=221770,00.jpg` and `https://emotiv.com/upload/iblock/801/mainscreen.jpg`

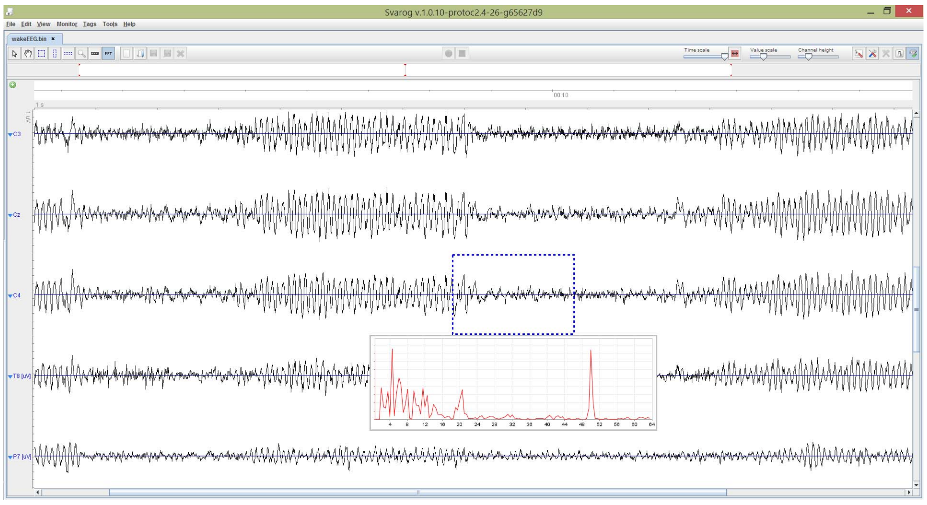

Svarog v.1.0.10-protoc2.4-26-g65627d9
File Edit View Monitor Tags Tools Help
wakeEEG.bin
Time scale
Value scale
Channel height
C3
Cz
C4
T8 [uV]
P7 [uV]

Figure 4.2: Svarog's main window

4.3 SigViewer

SigViewer is a viewing application for biosignals designed to display EEG and other biosignal data. It also supports creating annotations to select artifacts or specific events. Furthermore, it features signal processing modules such as the average over selected epochs and the power spectrum of selected signals.

The main window has a large area that is composed of individual charts for each signal. The chart can be vertically scrolled and zoomed independently of others, which is useful as waveforms have often very different shape. The annotation tool allows annotations to be created directly over the waveforms and right click on the annotation provides next options. The responsiveness somewhat degrades with too many displayed signals, probably because the plots are rendered in software. SigViewer is implemented in C++ and Qt.

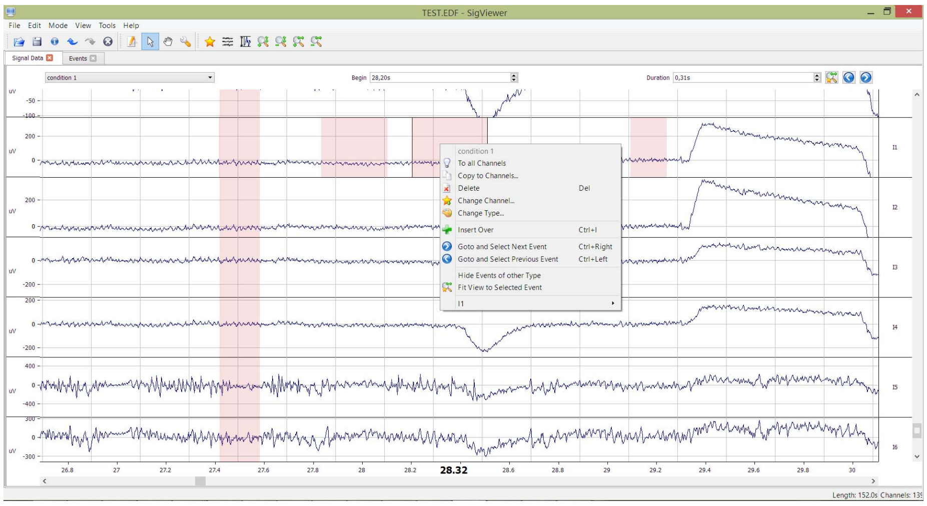

TEST.EDF - SigViewer
File Edit Mode View Tools Help
Signal Data
Events
condition 1
Begin 28,20s
Duration 0,31s
condition 1
To all Channels
Copy to Channels...
Delete Del
Change Channel...
Change Type...
Insert Over Ctrl+I
Goto and Select Next Event Ctrl+Right
Goto and Select Previous Event Ctrl+Left
Hide Events of other Type
Fit View to Selected Event
Length: 152.0s Channels: 139

Figure 4.3: SigViewer's main window

4.4 Comparison of Solutions

The analyzed solutions are diverse enough so that comparison can be made. From the targeted audience view point, it is noticeable that Emotiv software package with all its games and visualization tools targets casual users that are not so much interested in signal analysis, but in the utilization of a brain-computer interface for various tasks. On the other hand, Svarog and SigViewer are more suitable for researchers interested in data analysis. Svarog is a more feature complete and sophisticated application than SigViewer. However, the SigViewer's signal browser, which is the most important element in both applications, is much more convenient to use and signals are easier to interpret. Table 4.1 provides a short summary of features for each solution.

	Emotiv	Svarog	SigViewer
Recording Support	Yes	Yes	No
Signal Browser	Limited	Yes	Yes
Visual Electrode Maps	Yes	No	No
Annotation Support	No	Yes	Yes
Open Source	No	Yes (GPL)	Yes (GPL)

Table 4.1: Comparison of solutions

Chapter 5

Application Concept

It is usually a good idea to create a concept before implementing a non-trivial project. A concept development process helps to make clear which are the important parts of the project and how they are going to communicate and work together to provide desired results. The concept also helps to maintain a good code structure and class hierarchy. Therefore, the text in this chapter defines the concept that will be used during development of the application. Unless explicitly stated otherwise, the term *application* will further refer to a program that has been developed as a practical part of this thesis.

5.1 Logical Division

The application should serve as an EEG signal browser with support for a 2D and a 3D electrode visualization. Therefore, the application will be mostly graphical, as opposed to textual. A part of the application will have to support reading EEG data and another part will be responsible for displaying the data. Because the data are digital signals, a built-in support for signal processing would further improve usability of the application. After a deeper analysis of the problem it is possible to split the whole application into individual subsystems:

- GUI
- Model importer
- 3D math library
- Signal input module
- Signal processing module
- Rendering system
- Animation system

The following sections will address each subsystem individually.

5.1.1 Graphical User Interface

A user interface provides both a user input to the application and an output or feedback from the application. The *graphical user interface* (GUI) is a user interfaced composed of graphical elements, such as windows, buttons, text areas, scrollbars, etc. The elements can be grouped and create layouts that are visually appealing.

An important question to ask is who will be the target audience. The difference between targeting ordinary users and researchers can be seen in Chapter 4. For example. The original assignment predetermines this work to be used primarily for academic purposes. Nevertheless, the ideas of visually appealing graphical elements and friendly user interface will be also incorporated as they can dramatically improve the experience.

A common approach to GUI design is to first create a simple model called mockup. The mockup can be drawn on paper or created using a specialized software. It should capture all the important elements and define their position and purpose. The GUI is then created to resemble the mockup using a technique that is specific to a platform. Section 6.3 presents GUI that was created for application.

5.1.2 Model Importer

Loading 3D mesh models into a memory in a representation that may be used for rendering is a non-trivial task. This is further complicated by the diversity of file formats that are used to store 3D models. One of the most popular file formats is `Wavefront .obj` file. It can store whole scenes with multiple objects that are represented using polygons. Most of the 3D modeling software also allows exporting the model composed solely of the triangles that can be rendered directly by OpenGL. Another advantage of the `Wavefront .obj` file format is that it uses ASCII and human readable data representation. The Listing 5.1 shows an example of `.obj` file that defines a triangle. Due to the stated properties, the format was chosen to store the model of brain. The model of brain was provided by the *Brain for Blender* project[1]. This brain is an MRI scan of a real human brain.

```
# object Triangle
v  -0.5 0 0
v  0.5 0 0
v  0.25 0.5 0
# 3 vertices

g Triangle
f 1 2 3
# 1 face
```

Listing 5.1: The .obj file example

A model importer has to be able to open `.obj` file, list stored objects, and provide data for each of them. The data, such as vertex positions, are then passed to rendering system that will render the objects. A *Tiny obj loader*[2] library is a perfect solution for our needs. It is light-weight C++ library implemented by single file that is simply compiled with the rest of the application and has no dependencies except for C++ STL. After loading the triangulated model from `.obj` file, the geometry and other information is conveniently stored inside STL `vector` that can be used directly by rendering system.

[1] http://brainder.org/download/brain-for-blender/
[2] http://syoyo.github.io/tinyobjloader/

5.1.3 3D Math Library

A 3D math library is required in order to manipulate objects in 3D space. Minimal requirement is a support for translation, rotation and projection is needed. The library should also include algorithms for creating transformation matrices used by OpenGL. Naturally, the data structures such as vectors, matrices, and basic geometry should be part of the library.

The *OpenGL Mathematics*[3] library covers most of these requirements. It provides the same functionality and data types that are found in the GLSL language plus additional useful features such as the matrix transformations, quaternions, data packing, etc. The only drawback is that the library does not provide any 2D or 3D objects. Therefore, the application will have to implement these as needed.

5.1.4 Signal Input Module

The EEG data is required by the application in order to produce actual results. The problem is that there are many file formats actively used, some of which are specific to a single medical laboratory or recording equipment. However, there is an effort to create single standard file format. The European Data Format (EDF) is an example of such effort. This format stores electrode signals in form of data records which are time-continuous blocks of samples. The electrode signal may consist of multiple data records and each data record has specified start time. Therefore, the data records may not be contiguous, resulting in a sparse recording. The format also provides additional information, such as duration of the data record and duration of the whole file, maximum and minimum of the signal amplitude, information about a patient, etc. [6].

The EDF format was chosen to be a primary source of EEG data because of its popularity and versatility. However, the data format is not trivial to read. Because of this, an already existing C library was used. The *EDFlib*[4] library supports not only the original EDF format but also its extension EDF+ and derivations BDF and BDF+. The library has also some limitations, such as that the files must be continuous. In other words, sparse recordings are not supported. However, discontinuous files can be converted to continuous files with *EDFbrowser*[5]. After opening a file in the EDF format, each signal can be read individually as a stream. The signal input module then provides the data to the rest of the application in an internal format.

5.1.5 Signal Processing Module

The signal processing module implements filters and transformations presented in Section 3.1. As discussed in that section, the discrete Fourier transform is usually implemented as the fast Fourier transform. There are already many libraries implementing FFT. One such library is *Kiss FFT*[6] and it was chosen due to its simplicity, compact size, and benevolent license. However, no simple library could be found that implements FIR filters. It was decided the few filters that are necessary will be implemented anew, rather than including large digital signal processing DSP library with lots of dependencies. Section 6.2 provides implementation details for these filters.

[3]`http://glm.g-truc.net/0.9.6/index.html`
[4]`http://www.teuniz.net/edflib/`
[5]`http://www.teuniz.net/edfbrowser/`
[6]`http://sourceforge.net/projects/kissfft/`

5.1.6 Rendering System

The rendering system provides a necessary functionality to produce a visual output. It is the most complex part of application and can be further divided into:

- General rendering support for OpenGL primitives
- 3D mesh object rendering support
- Electrode visualization
- EEG data visualization

The whole rendering system is hardware accelerated in order to provide a good performance. The hardware acceleration is provided by the OpenGL API and graphics pipeline introduced in Section 3.3. Because of that, a shader program is needed for each rendering task. The implementations details of shader programs are discussed in Section 6.4.

A *Unishader*[7] library is used to load and utilize the shader programs. The Unishader is previous work by author of this thesis that creates an automatized wrapper around OpenGL focusing on the shader support. It greatly simplifies the setup and rendering using shader programs. For example, it can automatically query a shader variable type and use appropriate functions, compile and link programs when needed, perform configuration checks, and manage GPU resources. It also helps to keep OpenGL context in a valid state, which is a non trivial task that requires tracking of all the performed operations.

5.1.7 Animation System

The animation system prepares the EEG data for visualization. The system resembles a media player that can play, pause and rewind the content, in this case EEG data. However, the amplitude of signal still needs to be somehow transformed into visual information. It was decided that the color of electrode will be used to represent the amplitude. The green color is used for zero amplitude. The maximum positive amplitude is represented by red and negative amplitude by blue color. The resulting gradient is shown in Figure 5.1. If no signal is assigned to the electrode it remains gray. A purple color is used if there is not enough data available. The section 6.5 describes the conversion of signal data to the color in more detail.

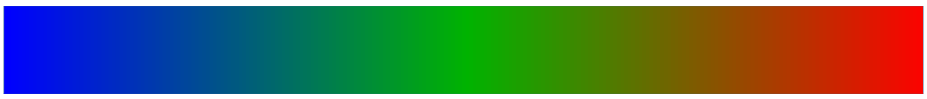


Figure 5.1: Electrode color gradient

[7]https://github.com/BetaRavener/UniShader

Chapter 6

Implementation

This chapter provides more details about significant algorithms and methods used during development of application. The information provided here is still at some level of abstraction so that the algorithms can be understood without knowledge of a specific programming language.

6.1 Electrode Placement

It is possible to use a custom layout and electrode labels by importing `Electrode map file`. The structure of file is simple with a single line header as a first line that is used to check compatibility. The rest of the file are tuples of an electrode name, a 2D position, and a 3D position. Each tuple specifies a single electrode. If an electrode position is unknown or was not supplied, it may be omitted by using `*` . An example of such file is shown in Listing 6.1.

```
Electrode map file v100
AF1 -5.91481 30.148 16.8014 24.7601 98.6503
AF10 29.3892 40.4509 *
AF2 5.9148 30.148 -17.1713 24.3079 98.5821
```

Listing 6.1: Electrode map file example

If there is a file `default.elmap` present inside `electrodes` folder during the application startup, it will be used to load electrode positions. Otherwise, electrodes will be named according to the 10-10 electrode placement system presented in Section 2.2.3 and 2D positions will be generated by program but no 3D positions will be available because automatic 3D placement is not supported. The 2D layout tries to resemble those found in scientific papers. The 3D positions may be additionally loaded from `.obj` file. The objects inside this file must be named after electrodes so mapping can be made. The shape of the object doesn't matter as long as the object's center of gravity is in desired position. The Figure 6.1 illustrates this process using Autodesk 3ds Max.

1. The brain model that is used in application is imported to 3ds Max.
2. The model of brain is then fitted by a model of skull[1] that provides features necessary for electrode placement, such as nassion, inion, and auricular points.

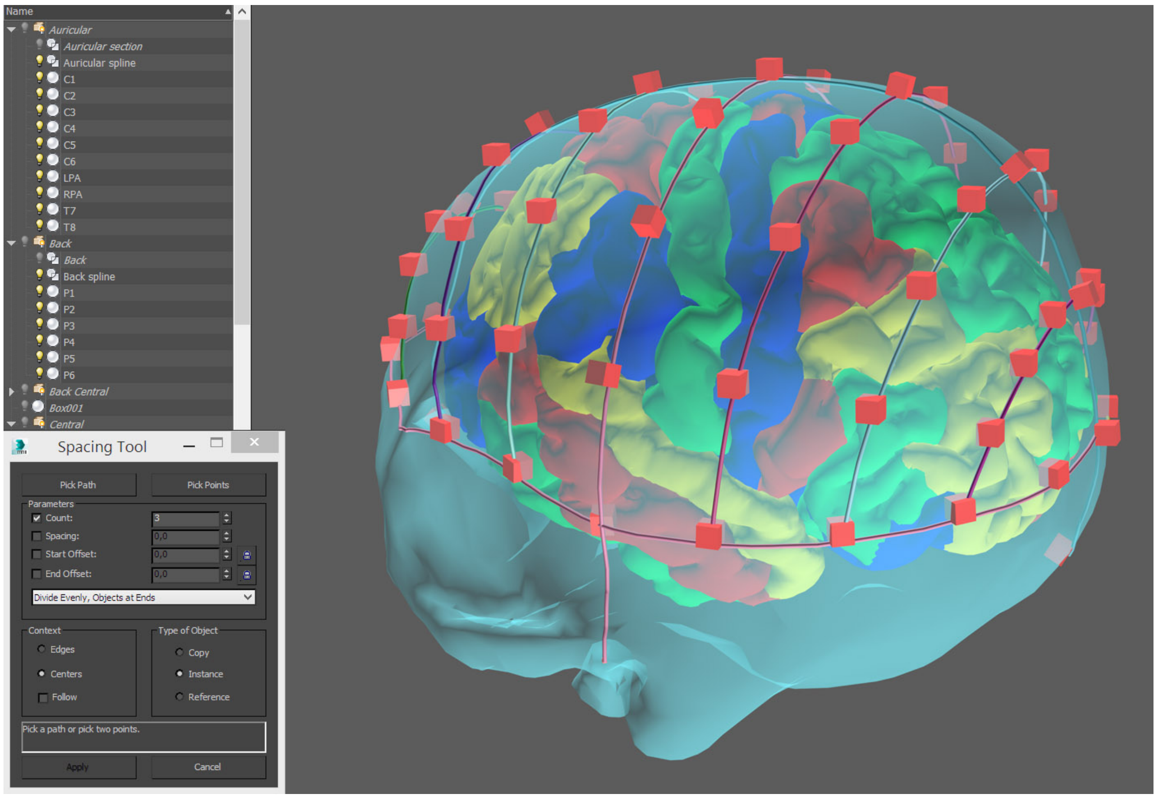
[1]The model of skull was obtained from http://www.sharecg.com/get_file.php?upload_file_id=43787&PSID=7e83ca0fc9bc28c873e73f17939f7b13

3. The important contours are marked according to Section 2.2.3 and objects, in this case boxes, are spaced evenly along the contours. The contour is obtained by slicing the skull with a plane that passes through marked points.

4. The objects are named after the electrodes.

5. `electrodes.obj` file is exported from 3ds Max.

This file can be imported into application to provide 3D electrode positions.



[✓] Count: 3
[ ] Spacing: 0,0
[ ] Start Offset: 0,0
[ ] End Offset: 0,0
Divide Evenly, Objects at Ends
Context: Edges [✓ Centers] [ ] Follow
Type of Object: Copy [✓ Instance] Reference
Pick a path or pick two points.

Figure 6.1: Electrode placement in 3ds Max

As mentioned before, the 2D layout can be generated by the application if the default file was not found. This is done by following procedure:

1. A horizontal and a vertical line is created so that their middle points cross at a coordinate space origin. Both lines have length of 1. The lines are evenly divided into 10 segments. The end of each segment represents an electrode position.

2. Two circles are created with center at the origin. First circle has a radius of 0.5 and second has a radius of 0.4. Both circles are evenly divided into 19 segments. Again, the end of each segment represents an electrode position.

3. 6 arcs are created by circumscribing circle to the 3 already defined points. The first point is on the left side of the smaller circle, followed by a point on the vertical line, and the last point is on the right side of the smaller circle. This is better described by Figure 6.2. Each arc is then divided into 8 segments and an electrode is placed at the end of each but last segment.

The result is the 2D layout centered at the origin and bounded by a square with a side length of 1. The layout can be scaled as needed to prevent electrodes from overlapping when rendered. Figure 6.5 shows the described layout rendered by the application.

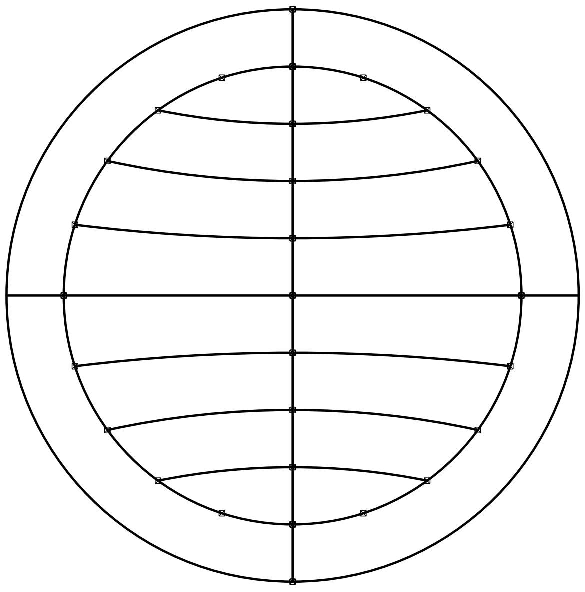


Figure 6.2: Automatic 2D electrode placement

6.2 FIR Filters

As discussed in Section 3.1, the filtering is done by convolution of the input signal with a filter's impulse response. The filter's impulse response is computed directly from the equations presented in Section 3.1.3. To implement convolution, a sliding window of predefined length is first created. The order of the filter determines the length of the sliding window. The sliding window moves over the signal and at each step it is filled by values it covers. The values are then multiplied by the filter's impulse response and window function coefficients. The values are summed and stored as a single element in a new, filtered sequence. The process is illustrated by Figure 6.3.

However, there is a problem at the beginning of the input signal because there are not enough values to fill the sliding window. This is solved by prepending the input sequence with initial conditions. This is a short sequence which per-fills the sliding window with values, so that first input sample can already produce an output sample. The conditions are generated for each signal individually before it is processed. They may be just filled with zeros, repeat the value at the edge of the signal, or copy the signal values in reversed order. The choice usually depends on the type of the filter.

In order to test the implementation, the results were compared with MATLAB output. First, a set of testing data was created in plain-text format. This data was then processed by the implementation described here using multiple combinations of cut off frequencies, window lengths and window functions. The output was stored in a new file and compared to the output produced by MATLAB filters with the same configuration. The difference between the two was only marginal and can be attributed to floating point errors.

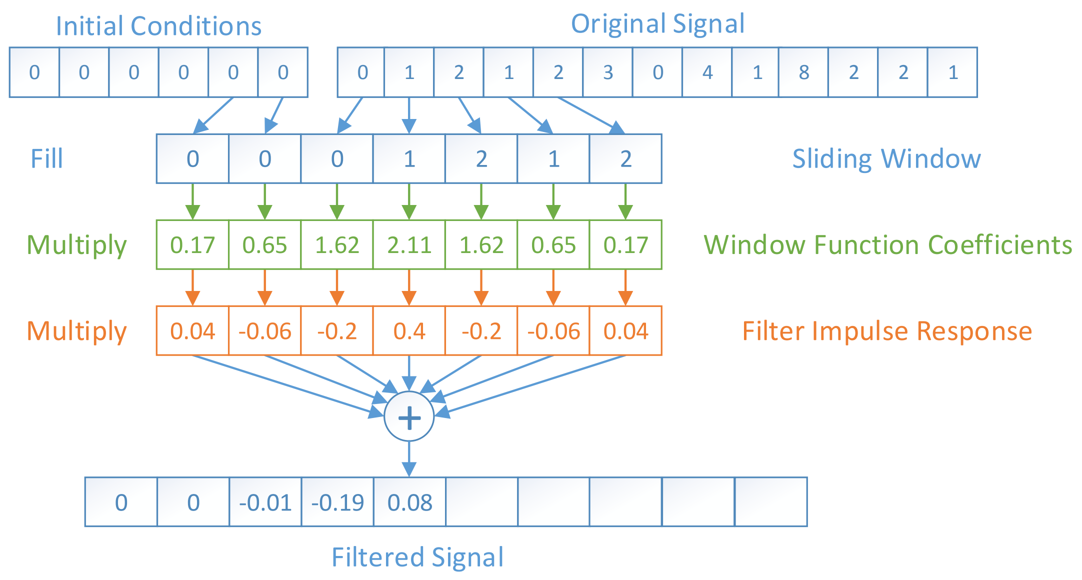


Figure 6.3: Illustration of signal filtering

6.3 Graphical User Interface

The application user interface was created with emphasis on simplicity and clean look. When the application is executed, only a main window appears. Its dominant element is the chart view described in Section 6.4.2 and displayed in Figure 6.4. The main window features only two more buttons for playing and rewinding the animation. The current animation time is shown in the chart as a yellow line. Using a right mouse button over the chart moves the line into the mouse position and sets the animation time accordingly. The main window menu bar gives access to additional dialogs and windows:

Open signal file dialog provides interface for opening a file with data and listing available signals. The user then selects which records he wants to work with by moving them into the second list. The labels used for records in the file may not conform to names used for electrodes in the electrode placement system. To address this, the user can assign the electrode to each recording manually. However, if the electrode and data record labels match, an auto-assignment button can be used to pair them. The electrode assignments can be exported to a file and imported later as needed.

Electrode map dialog allows importing and exporting electrode map files. It also allows electrode positions to be set using `.obj` file as described in Section 6.1.

Filter dialog provides an interface to the signal filtering. It allows choosing between the low pass and high pass filter and specifying the cutoff frequency, window function, and length of the window. The dialog also provides feedback on progress in form of a progress bar as the filtering operation can take a long time to finish.

Player settings dialog allows the animation settings to be modified. The gain factor introduced in Section 6.5, refresh rate, and animation speed can be changed here. The

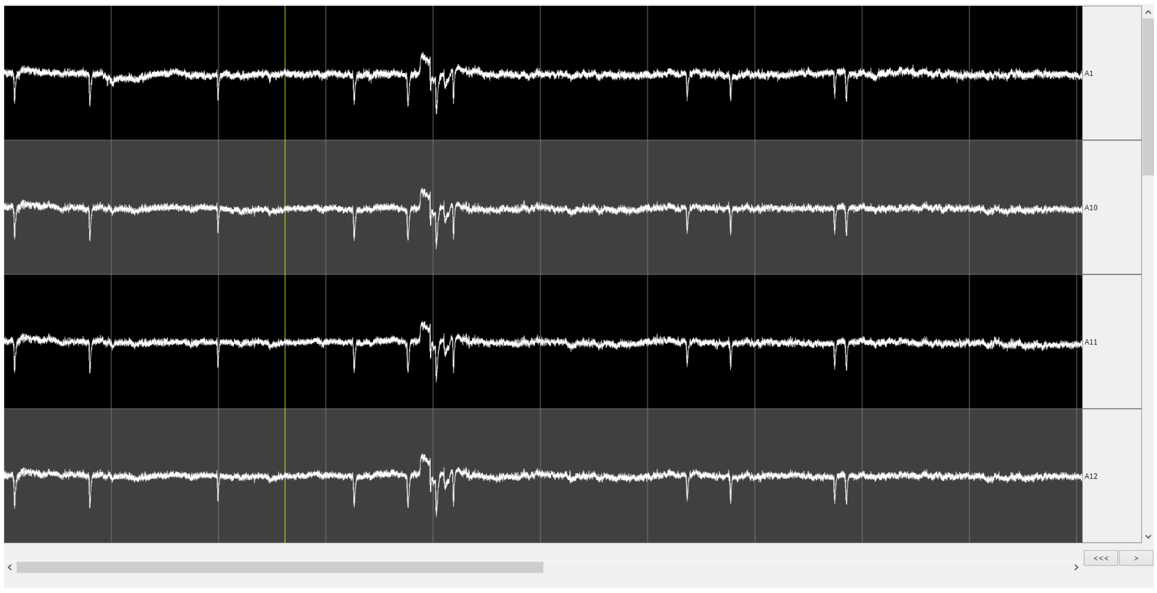


Figure 6.4: Chart view

user can also select the transformation used by animation and specify the frequency range. Changes to this dialog are applied immediately to make it easier to experiment with different values.

Electrode 2D and 3D view can be displayed, each in its own window. Both views can be manipulated using mouse. The panning can be done by moving the mouse while holding the right mouse button and using the scroll wheel the user can zoom in and out. In the 3D view the user can also rotate the view by holding the left mouse button. Figures 6.5 and 6.6 show the 2D and 3D view respectively.

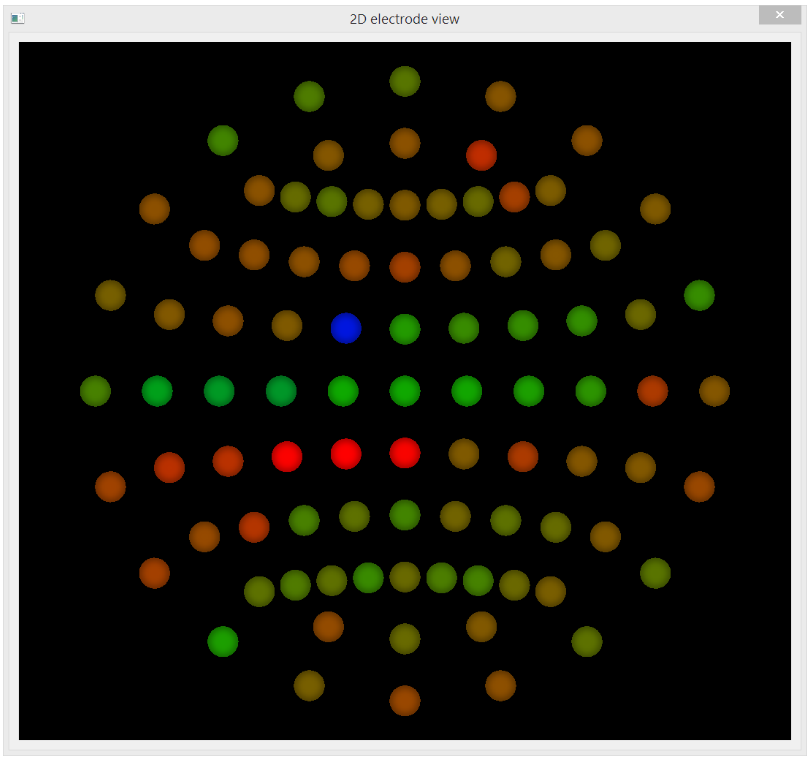


Figure 6.5: Electrode 2D view

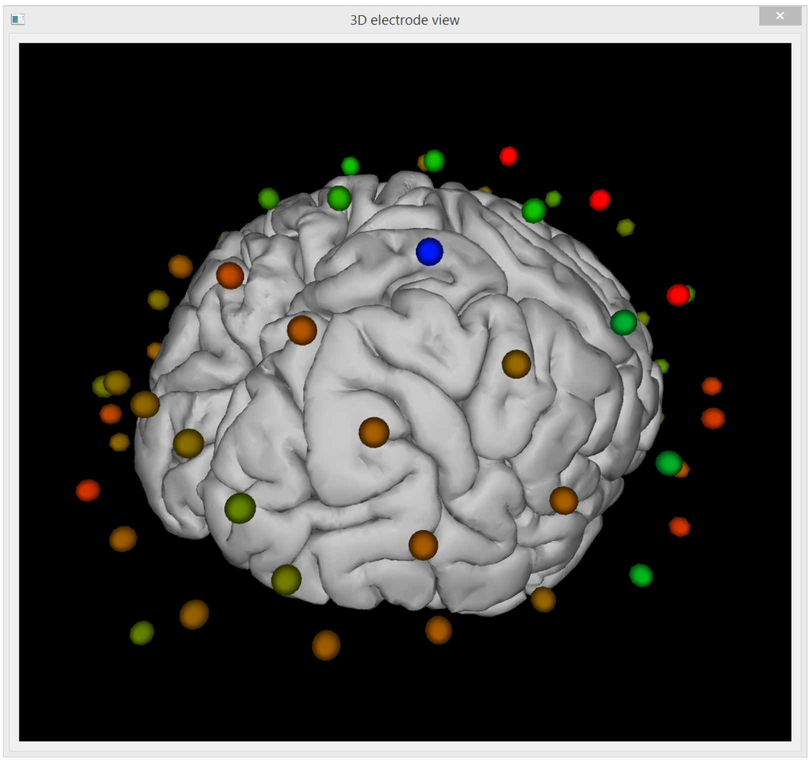


Figure 6.6: Electrode 3D view

6.4 Rendering System

All the rendering tasks are hardware accelerated using OpenGL in order to provide a reasonable performance. Because OpenGL is only a low level API, a group of shader programs had to be developed to provide required rendering support. The first shader program allows shaded mesh objects to be rendered in a 3D space. A shader program that is used for electrode visualization is described in Section 6.4.1. Another shader program provides functionality to render signals in time domain using lines with custom thickness. More details about this shader program are provided in Section 6.4.2. Finally, a general purpose shader program for rendering 2D geometry is included that can be used to render miscellaneous objects in orthogonal projection.

6.4.1 Electrode Visualization

The visual appearance of electrodes should resemble an illuminated sphere. Normally this would be done by creating a mesh of the sphere that could be rendered the usual way. However, in order to save memory and to demonstrate capabilities of shader programs, we used a different approach. The input to the shader program consists only of a sphere center position, a color, and a radius. The vertex shader passes the data unaltered to geometry shader. The geometry shader then generates a triangle fan around the center position that is always directed towards the camera. The number of triangles in the fan is variable and when high enough, the fan starts to resemble a circle. The result is then passed to the fragment shader that applies Phong shading. The shading gives a 3D effect to the circle which now looks like a sphere. Figure 6.7 illustrates the described rendering process.

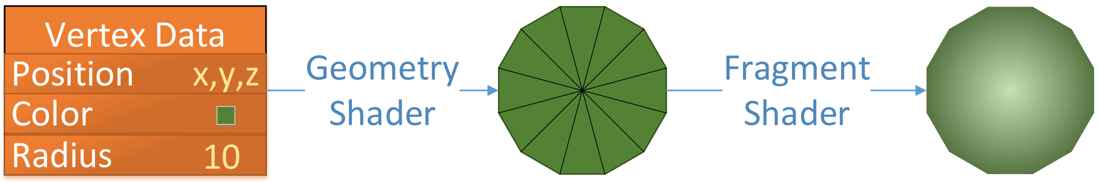


Figure 6.7: Electrode rendering process

6.4.2 Chart View

The chart view is used primarily to display EEG signals in a time domain. The view can be zoomed horizontally to control displayed duration. A vertical zoom is also supported so that the number of signals displayed in the view can be changed. However, the amount of data from an ordinary EEG measurement is fairly large. 10 minutes of recording at sampling frequency of 200 Hz produces 120000 samples for each electrode. Rendering so much data would be a very demanding task. Moreover, let's consider a FullHD computer monitor with horizontal resolution of 1920 pixels and a chart view that is fully zoomed out. In this case, approximately 63 samples map to each monitor column, which is obviously redundant.

A solution to this problem is a signal decimation. The most simple way to decimate a signal is to take each N-th sample. However, this would produce very poor results. A better method is to find maximum and minimum value from the samples that would map to the same monitor column and render both of them. While this method is more computationally demanding than just taking N-th sample, it is still much more efficient than rendering all of the data and preserves the shape of the signal. The difference between an original signal and a signal decimated using N-th sample and Min-Max method can be seen in Figure 6.8. The decimated signal contains in both cases 100 times less samples than original signal. The decimation was performed on the actual EEG data. To optimize even further, the signal decimation is performed only when horizontal zoom changes and the result for that zoom level is cached. This allows scrolling the view swiftly without any further preprocessing. This approach can be also used for real-time recording when data are not static. A newly

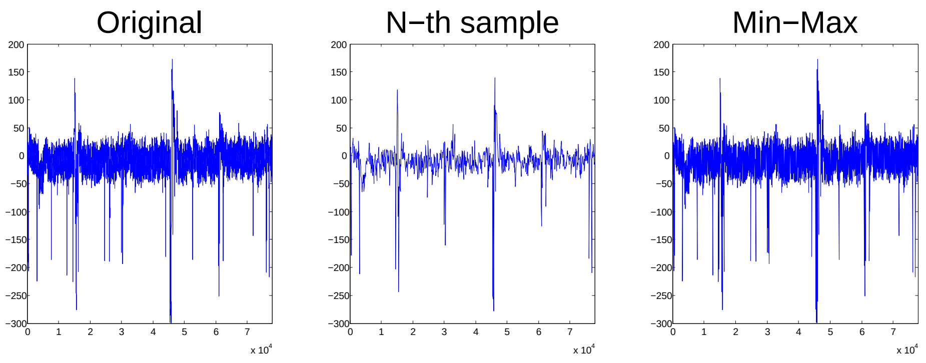


Figure 6.8: The original and decimated signals

recorded sample is simply appended to the signal and only last value of cache needs to be updated.

The decimated signal can be finally rendered using a specialized shader program. The main feature of this shader program is the support for rendering lines with specified thickness. This must be done by generating two triangles for each line in a geometry shader because OpenGL can draw only lines without thickness. However, this produces gaps between lines as the slope changes. In order to render a smooth, continuous curve, the geometry shader also creates capped line joins. Figure 6.9 illustrates the situation where the points A, B, and C are the samples that define two lines. Furthermore, each signal has designated area for drawing to prevent overlaps which are confusing when analyzing data. A pixel that is produced outside of this area is discarded by fragment shader.

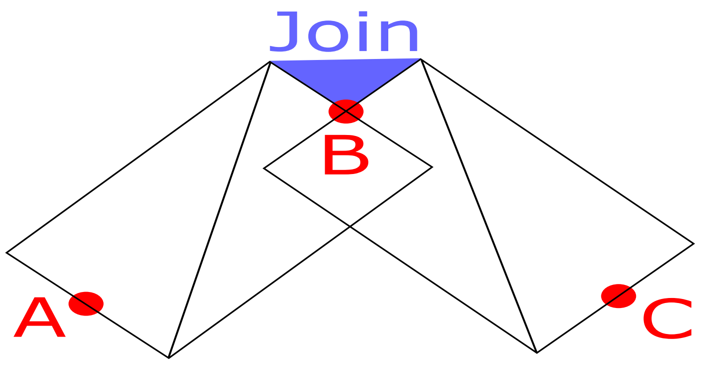


Figure 6.9: The lines with a thickness and a capped line join

The chart view is interconnected with Qt elements. When scrollbar is moved or the window size changes, the view is scrolled or updated accordingly. Therefore, it was necessary to somehow define the displayed section so that both the GUI and the rendering system would be aware of what is displayed. At first this was done by calculating the number of samples that are visible and scrolling was done by changing the index of first sample. However, this approach was difficult to work with and had many limitations, such as that sampling frequency each displayed signal had to be the same. Because of this, the viewed section was later redefined as a time period. Scrolling is done by changing the beginning of the time period and horizontal zoom is performed by altering the duration.

6.5 Animation of Electrodes

The electrodes are animated by changing their color over time. The electrode remains gray if there is no signal assigned. Otherwise, a normalized value in range from -1 to 1 determines the color of the electrode using gradient presented in 5.1.7. A user can choose between two animation modes. The first one produces normalized value directly by dividing the amplitude of the signal at the current animation time by the maximum signal amplitude range according to Equation 6.1. The second mode uses FFT to obtain the spectrum of the signal around the current animation time. The user can specify frequency range that will be analyzed. A frequency component with maximum amplitude over this range is then selected and normalized. The electrode color turns purple if there is not enough data to perform FFT. Because most of the time the signal amplitude is much lower than maximum amplitude, a gain factor can be specified. The normalized value is multiplied by this factor and the result is clamped back to the allowed range.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{6.1}$$

Chapter 7

Conclusion and Future Directions

The goal of this thesis was to implement an application for visualization of brain activity measured by EEG. Visualization methods were expected to include two- and three-dimensional brain models along with methods for graphical representation of brain signals, and a graphical element for browsing EEG signals.

To better understand the problem, the thesis provided necessary information, beginning with an introduction to a human brain and description of neuronal activity that occurs within the brain. EEG, a method for measuring this activity, was presented in detail along with information about electrode placement, which was required later for correct electrode positioning. This work also provided information about basic methods of signal processing, such as DFT and signal filtering, and presented modern techniques of data visualization using GPU accelerated shader programs.

Following the theoretical background, a concept for a multi-platform application was presented. The concept built upon previously analyzed applications created by other individuals or companies and defined the structure of the application. This concept was later transformed into an implementation. Significant algorithms along with solutions to problems that arose during implementation were described. One such problem was that the amount of data produced by EEG recording was large and had to be optimized in order to be visualized with a reasonable performance. The performance had to be considered also when designing the signal processing module as it encompasses demanding operations.

All rendering tasks were hardware accelerated by a GPU. Therefore, the host application, which was executed on a CPU, had to be accompanied with a set of shader programs that can render brain, electrodes, signal waveforms, and other visual elements. The hardware acceleration allowed large data sets and models to be rendered in quality that would not be possible with software rendering.

The work provides solid foundations for an application which can serve as an EEG data analysis tool. The application combines the ability to browse signals with interactive graphical visualizations and capability to replay measured data. A built-in signal processing module allows data to be filtered directly by the application, which obviates the need for another application and data reloading.

The application was tested under operating systems Windows 7 and 8, which were the targeted platform. The application was also compiled, linked, and executed successfully under Linux distribution Lubuntu. The signal processing methods were tested against MATLAB implementation.

EEG, signal processing, and data visualization are broad subjects and as such provide room for a lot of improvements. The application was formerly designed as both an online and an offline data analyzer and visualizer. However, only the offline part was implemented due to complications with real-time signal processing and unavailability of recording equipment.

Therefore, the plans for the future development include the real-time input module and modification of signal processing algorithms. A preliminary concept is to provide an interface for creating a sequence of filters that will be applied to newly acquired samples. Once the sample passes all the filters, it can be simply appended to existing data. However, such filter sequence can create a considerable delay, which will prevent real-time visualization. To address this issue, a minimum-phase FIR filters or IIR filters should be used.

Other than that, the application could benefit from a wider range of data processing tools and visualization methods, such as frequency spectrum chart that could display a result of FFT. There could be a window with a grid of charts, where each chart would belong to a single signal and would display it's frequency spectrum at the current animation time. Such visualization would be very useful when analyzing EEG data for brain waves. These new methods of visualization are important because they provide better insight into analyzed data and highlight signal features.

The plans for future development also include a plugin system. Currently, the only way to provide a new feature is to add source code to application and recompile. A better approach would be to create the plugin system which could automatically load new functionality from libraries at the application startup.

Bibliography

[1] *Brain Facts: A Primer On the Brain and Nervous System.* Society for Neuroscience, Washington, D.C., 4th edition, 2002.

[2] Odile Benoit, Agnès Daurat, and Jacques Prado. Slow (0.7–2 hz) and fast (2–4 hz) delta components are differently correlated to theta, alpha and beta frequency bands during nrem sleep. *Clinical Neurophysiology*, 111(12):2103–2106, 2000.

[3] Thomas F Collura. History and evolution of electroencephalographic instruments and techniques. *Journal of clinical neurophysiology*, 10(4):476–504, 1993.

[4] Henri Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, 100(6):623–629, 1971.

[5] David M Groppe, Stephan Bickel, Corey J Keller, Sanjay K Jain, Sean T Hwang, Cynthia Harden, and Ashesh D Mehta. Dominant frequencies of resting human brain activity as measured by the electrocorticogram. *Neuroimage*, 79:223–233, 2013.

[6] Bob Kemp and Jesus Olivan. European data format 'plus'(edf+), an edf alike standard format for the exchange of physiological data. *Clinical Neurophysiology*, 114(9):1755–1761, 2003. Online version `http://edfplus.info/specs/edfplus.html`.

[7] Andrea Kübler, Boris Kotchoubey, Jochen Kaiser, Jonathan R Wolpaw, and Niels Birbaumer. Brain–computer communication: Unlocking the locked in. *Psychological bulletin*, 127(3):358, 2001.

[8] Jian Le, Min Lu, Emiliana Pellouchoud, and Alan Gevins. A rapid method for determining standard 10/10 electrode positions for high resolution eeg studies. *Electroencephalography and clinical Neurophysiology*, 106(6):554–558, 1998.

[9] Joachim H Nagel. Biopotential amplifiers. *The Biomedical Engineering Handbook*, 2000.

[10] Jorge Baztarrica Ochoa. Eeg signal classification for brain computer interface applications. *Ecole Polytechnique Federale De Lausanne*, 7:1–72, 2002.

[11] International Organisation of Societies for Electrophysiological Technology. Guidelines for digital eeg. *American Journal of Electroneurodiagnostic Technology*, 39(4):278–288, 1999.

[12] Robert Oostenveld and Peter Praamstra. The five percent electrode system for high-resolution eeg and erp measurements. *Clinical neurophysiology*, 112(4):713–719, 2001.

[13] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time signal processing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[14] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[15] John G. Proakis and Dimitris G. Manolakis. *Digital signal processing: principles, algorithms, and applications*. Prentice Hall, Upper Saddle River, NJ, 1996.

[16] Babak A Taheri, Robert T Knight, and Rosemary L Smith. A dry electrode for eeg recording. *Electroencephalography and clinical neurophysiology*, 90(5):376–383, 1994.

[17] Desney Tan and Anton Nijholt. Brain-computer interfaces and human-computer interaction. In *Brain-Computer Interfaces*, pages 3–19. Springer, 2010.

[18] Michal Teplan. Fundamentals of eeg measurement. *Measurement science review*, 2(2):1–11, 2002.

[19] Woradorn Wattanapanitch, Michale Fee, and Rahul Sarpeshkar. An energy-efficient micropower neural recording amplifier. *IEEE Transactions on Biomedical Circuits and Systems*, 1(2):136–147, June 2007.

[20] Richard S. Wright. *OpenGL Superbible: Comprehensive Tutorial and Reference*. Addison-Wesley, Upper Saddle River, NJ, 5th edition, 2011.

Appendix A

Content of DVD

This work contains a DVD with following structure:

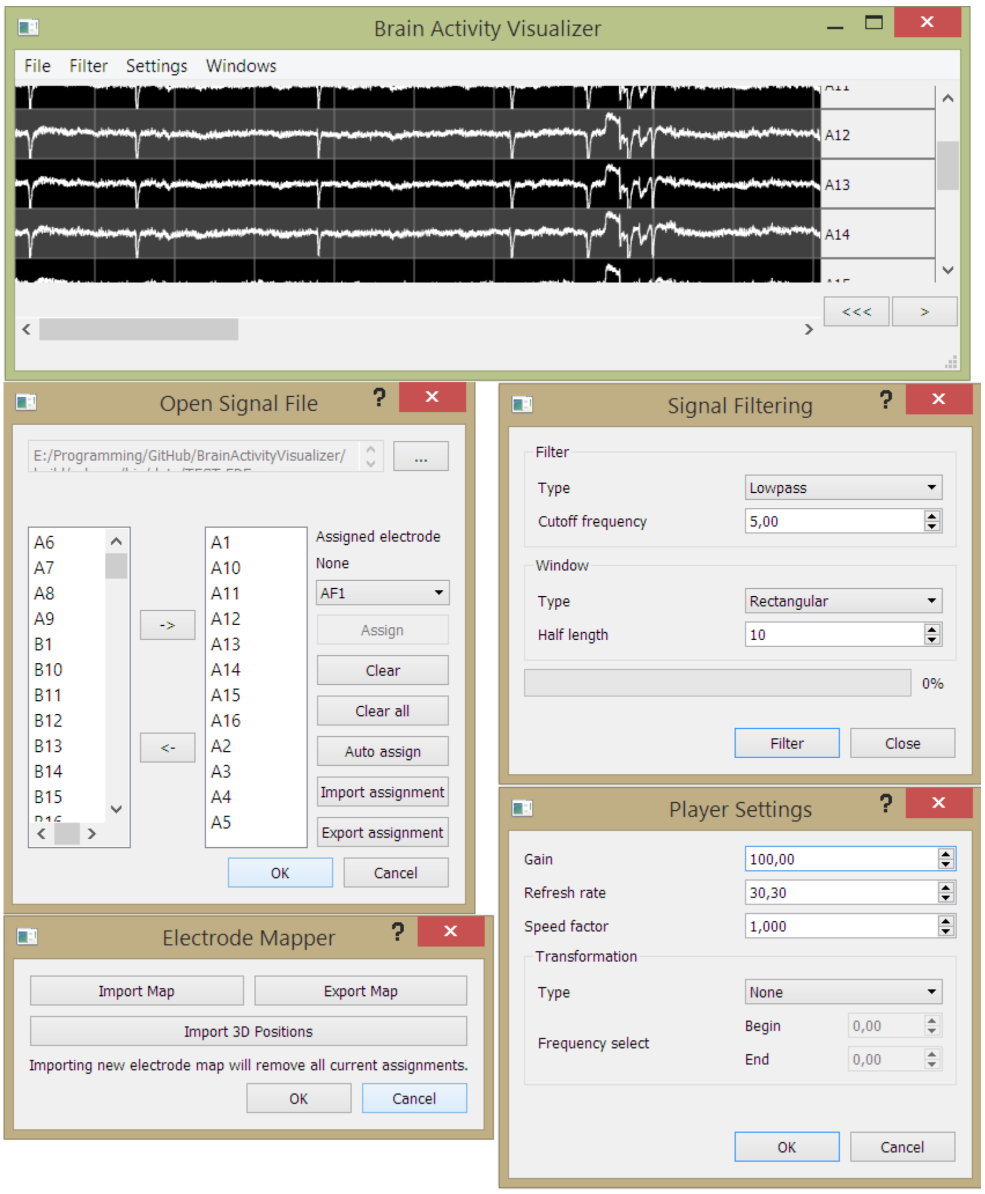
- **bin/** folder with a self extracting archive containing compiled application for Windows along with required libraries.
- **doc/** folder containing source code documentation generated by Doxygen.
- **latex/** folder containing LaTeX source codes for generating text of this thesis.
- **source/** folder containing source codes for building the application. Refer to the included **README.md** for instructions on how to build the application.

The content of the archive with compiled application is following:

- **data/** folder containing EEG data in the EDF format for testing purposes.
- **electrodes/** folder which contains `default.elmap` specifying default electrode placement and `electrodes.obj` providing 3D electrode positions.
- **licenses/** folder containing licenses for used libraries, EEG data and models.
- **models/** folder containing mesh models.
- **platforms/** folder with Qt libraries specific for Windows platform.
- **shaders/** folder with GLSL source codes that are compiled during run-time by the application.
- **bav.exe** – the application executable.
- **.dll files** – dynamic libraries required by the application.
- **MANUAL.txt** – short manual describing basic interaction with the application.

Appendix B

Image of Graphical User Interface



E:/Programming/GitHub/BrainActivityVisualizer/

A6	A1	Assigned electrode
A7	A10	None
A8	A11	AF1
A9	A12	
B1	A13	
B10	A14	
B11	A15	
B12	A16	
B13	A2	
B14	A3	
B15	A4	
B16	A5	

Filter

Type	Lowpass
Cutoff frequency	5,00

Window

Type	Rectangular
Half length	10

0%

Gain	100,00	
Refresh rate	30,30	
Speed factor	1,000	

Transformation

Type	None	
Frequency select	Begin	0,00
	End	0,00

Importing new electrode map will remove all current assignments.

Appendix C

Signal Filtering Tests

The signal module which encompasses signal processing methods was tested against the MATLAB software to check the validity of implementation. First, a random signal with 10000 samples was generated. The signal was then filtered by the testing executable **sigTest.exe**. The testing executable is generated during build if the project was configured with `-DTEST_PROJ=true`. The filtering was done in multiple configuration which are listed in table below.

Filter	Window	Order	Cutoff frequency	Filename
Low-pass	Hamming	10	5	low5Hamm5
Low-pass	Hamming	10	10	low10Hamm5
Low-pass	Hamming	20	5	low5Hamm10
Low-pass	Hamming	20	10	low10Hamm10
Low-pass	Blackman	10	5	low5Black5
Low-pass	Blackman	10	10	low10Black5
Low-pass	Blackman	20	5	low5Black10
Low-pass	Blackman	20	10	low10Black10
High-pass	Hamming	10	50	high50Hamm5
High-pass	Hamming	10	100	high100Hamm5
High-pass	Hamming	20	50	high50Hamm10
High-pass	Hamming	20	100	high100Hamm10
High-pass	Blackman	10	50	high50Black5
High-pass	Blackman	10	100	high100Black5
High-pass	Blackman	20	50	high50Black10
High-pass	Blackman	20	100	high100Black10

Table C.1: Tested filter configurations

Each configuration produced single output file with the filtered signal. This file was read by a MATLAB script and compared to the result of filtering using MATLAB filters in the same configuration. The maximum and average deviation was calculated and printed to output. The difference between signals filtered by testing executable and MATLAB in all configurations was marginal and can be attributed to floating point errors.

Appendix D

Code metrics

Files	89
Lines of code	7517
Percent of comments	11.5%
Statements	3436
Class definitions	47
Methods per class	7.98
Statements per method	5.2
Maximum depth	6
Average depth	1.12

Table D.1: Code metrics