JMFEL

The Java Media Framework Extension Layer

and its application to bio signal processing

Universität zu Lübeck

Institute for Signal Processing

Dipl.-Inf (FH) Olaf Christ

Lübeck, Germany _____

Thesis Declaration

I hereby declare that the work herein, now submitted as a Diploma Thesis in Computer Science at the University of Lübeck, is my own work, except where explicitly stated otherwise. I further declare that the work has not been submitted in whole or in part for any other degree.

Lübeck, (date) _____ (signature) _____

Matriculation Number: 527583

Introduction

With an increasing interest in multimodal data acquisition on general purpose computers running general purpose operating systems and several powerful data acquisition frameworks already available, it is surprising that many people still insist on reinventing the wheel when being faced with difficult tasks such as media synchronization, even though RFC documents for this specific application already exist.

My motivation therefore was to take and extend a multimedia framework and then use it to write bio signal acquisition and bio signal processing applications as, in principle, all these signal acquisition-applications are just multimedia applications.

Therefore, in this thesis, I present an extension layer to the Java Media Framework (JMF) which I will call the Java Media Framework Extension Layer (JMFEL) henceforth. This JMFEL aims to greatly improve the usability of the difficult to use and rather unwieldy JMF. Furthermore, the JMFEL's goal is to speed up application development by preventing unnecessary errors and by making JMF core features a lot easier to use. As an easier to use framework also means better applicability towards a wide range of problems – particularly to bio-signal processing – several examples and experiments will be demonstrated in this thesis.

The Java Media Frame Work version 1.0 was developed by Sun Microsystems, Silicon Graphics and Intel in 1997, followed by version 2.0 developed by Sun and IBM in 1999. Its most recent version is 2.1.1e, released in 2003. It supports a wide variety of codecs, where additional codecs are available through ffmpeg-support. While the JMF has not been updated for many years, and despite its lack of support for more recent codecs (when used without FFMPEG), it is still the most complete media API for the Java language.

My design goal for future developers using the JMFEL was to implement an object oriented design, where using design patterns means writing less code and therefore making less errors. It also means shorter development times and more experiment setups per year for researchers.

In this thesis, I will first deal with the history and technological basis of the EEG and ECG. I will proceed by describing visual perception and visual attention and will then move on towards eyetracking. This will be followed by a description of the core and support classes of the JMFEL. The underlying design of the JMFEL will be shown by using examples. I will then list the experiments with a bio-signal background carried out at the University of Lübeck, incorporating the AdRacer experiment. I will conclude with future prospects for the JMFEL and present possible offline and online editing methods as well as a design idea for a Brain-Computer-Interface.

(cf: http://java.sun.com/products/java-media/jmf/2.1.1/formats.html)

(cf: http://fobs.sourceforge.net/)

Acknowledgements

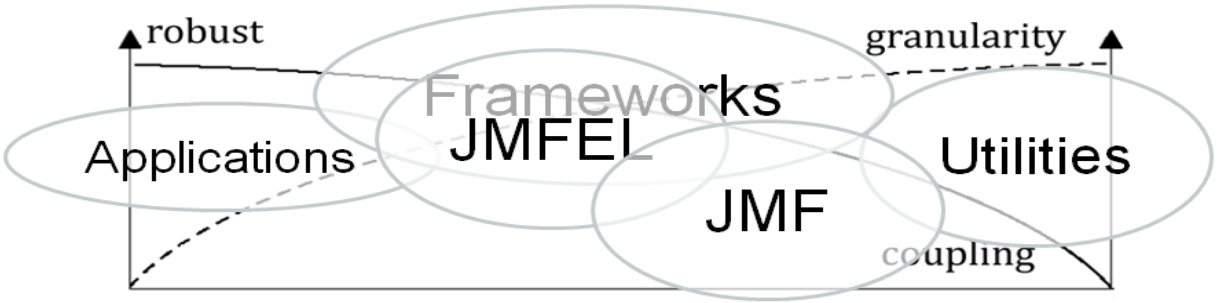
I would like to thank my supervisor PD Dr. rer.nat. Ulrich G. Hofmann for letting me work on my dream project and for constantly coming up with fun and challenging projects.
I would also like to thank Dr. rer.nat André Melzer for giving me the opportunity to work on the Adracer Project, Dr. William W. Armstrong for the Dendronic Learning Engine, Dr. Kevin Dolan for his VecMat Software package, my colleagues and co-workers Sebastian Otto, Radoslaw "Radek" Mazur, Matthias Pohl and everyone at the ISIP.
From the Clinic of Neurology, I would like to thank Andrea Timm and Jana Kujath for teaching EEG and Dipl.-Psych. Andreas Sprenger for helpful tips on EEG technology.
Last but not least, I would like to thank Christian Schmelzer for proofreading.

Table of contents

Methodology

To minimize risks – major design flaws and serious errors – the project has been developed according to the spiral model and parts of the extreme programming paradigm with a strong emphasis on prototyping and extensive testing. All prototypes leading to production code have been thrown away. Consequently, the final implementation of a component has always been written from scratch and vigorously tested against internal errors and other components to ensure interoperability. Bug fixes or enhancements have been made with respect to compatibility to other framework components. That is, a player component can have a new feature, but it should not – by any means – break correct code in other components or corrupt the whole framework. As a result, the framework went to several iterative cycles and versions and major changes have been only made at the beginning a new life cycle.

The JMF is designed for performance and extensibility - not necessarily for usability. The JMF requires quite some knowledge about programming and a lot of experience but provides all the features needed for developing very sophisticated multimedia applications. The idea of the JMFEL is about connecting data sources for reading media, players for media presentation, codecs or plugins for media processing and data sinks for storing media to some destination without having to think about what is going on behind the scenes. The JMFEL covers almost every feature of the JMF, helping the average as well as the more advanced developer to focus on what is important - writing multimedia applications and extending the framework where it is needed. A big design goal was to maintain loose coupling – a low fragility exhibited in the relationship between components – while at the same time keeping the granularity – the number of dependent objects – at a moderate level, thus assuring good reusability and performance. With respect to graphics – Java2D - programming, a high degree of interoperability and reusability was achieved by decoupling rendering code and its graphics contexts. As a result the very same drawing code can be used for passive, active, OpenGL rendering (Java2D/JOGL interoperability bridge). The small graphics framework responsible for this capability is completely independent from the JMFEL.



Stand-alone applications are usually made from specialized tightly coupled components, which makes them hard to reuse in other projects. Utilities, on the other hand are just collections of more or less independent programmes. Large collections are probably hard to maintain. A good framework should hit the sweet spot between coupling/robustness and granularity. The JMF sometimes feels more like a utility collection, because the developer has think about too many components and their dependencies. All this makes it very difficult to develop JMF applications. Because the JMFEL provides high level easy to use simple building blocks, its position is probably slightly more on the right side of the sweet spot.

(cf: Barry W. Boehm: A Spiral Model of Software Development and Enhancement. In: IEEE Computer. Vol. 21, 5th edition, May 1988, S. 61-72.)

(cf: Kent Beck: *Extreme Programming Explained: Embrace Change*, Addison-Wesley, ISBN 0-201-61641-6)

(cf: Kent Beck and Martin Fowler: *Planning Extreme Programming*, Addison-Wesley, ISBN 0-201-71091-9)

History of EEG

The phenomenon of (static) electricity has been around since ancient times (Thales from Miletos, 620-550 BC). In 1663, Otto von Guericke invented the Friction machine, which provided invaluable profound insights into electrically charged bodies. Eventually, Luigi Galvani (1737-1798) showed that body parts of organisms, e.g. a frog leg, responded to electricity. Consequently, he called his finding „Animal electricity“, a theory later disapproved by Volta, who called it „metal electricity“ because of the metal “electrodes“ used in the experiment. Galvani lacked the theoretical background of the laws of electricity to back up his theory. They were discovered by Georg Ohm in 1827.

Subsequent studies by Carlo Matteucci (electricity in muscle injuries, 1831), Herrman Helmholtz (1852, measuring the velocity of nerve conduction) and Emil Du Bois-Raymond (electrically measured nerve impulses, 1843) resolved this scientific controversy. In 1875, Richard Caton (1842-1926), a physician practicing in Liverpool, became interested in electro-physical phenomena. He used an early model of a galvanometer to perform bioelectrical measurements on more than 40 rabbits, cats and monkeys. He published his findings August 24, 1875 in the British Medical Journal. In these groundbreaking experiments, he performed the very first EEG measurements by demonstrating fluctuating potentials between electrodes placed on two different places of the scalp and the grey matter. He also recognized that the grey matter of the cerebrum was positive in relation to its deeper structures. He even showed that functional activity of the grey matter was exhibited by negative variations of its electric current. It took roughly 100 years until „evoked potentials“ as they are called nowadays, gained massive clinical interest.

Caton’s galvanometer only had a frequency response range of 0 to 6 Hz, limiting research. Nevertheless, the work of Caton inspired many scientists to start their research on the electrical phenomenon of the cerebrum throughout Eastern Europe. In the ongoing years, Gustav Fritsch and Eduard Hitzig showed for the first time, that against common believe the (human) brain can be electrically stimulated and that the cortex is comprised of functionally independent regions.

In full credit of Caton’s work, Vasili Yakovlevich Danilevsky worked on spontaneous electrical activity as well as (electric) stimulation.

The polish scientist Adolf Beck (1863-1939) performed studies on rabbits and dogs using non-polarizable electrodes to investigate spontaneous brain activity. He observed the

disappearance of rhythmical oscillations when the eyes were stimulated with light, an effect known as „alpha blocking“ specifically the „Berger Effect“, in accordance with Hans Berger, founder of the Human EEG.

Napoleon Cybulski (1854-1919), Beck's teacher in Krakow, contributed graphical representations of encephalographical measurements by using a galvanometer equipped with a photographical attachment. He presented graphical EEG evidence of an epileptic seizure in a dog, induced by electrical stimulation. With the advent of World War I, the progress in EEG research by Eastern-European researchers was considerably slowed down.
For 50 years these scientists had performed cutting edge research in neuroscience, including investigations on the electrical activity of the spinal cord and the oblongata in frogs (Ivan Michailovich Sechenov (1829-1905)) plus studies of the auditory cortex in the dog (Vladimir Efimovich Larionov).

Having had to work with primitive devices for decades, the introduction of the string galvanometer by Willem Einthoven in 1903 provided researchers with significantly more sensitive instruments. Einthoven's device featured photographic recording and quickly became the standard instrument of choice.

While neuroscience received strong attention in Eastern Europe, the Western European situation looked bleak. Unnecessary rivalries between scientists and questionable research by Ernst Fleisch von Marxow were watering down and interfering ongoing research. Fleisch von Marxow did not observe the oscillatory activity shown by Canton and Danilevsky, not even being aware of their work.

In 1920, Hans Berger, a German neuropsychiatrist, started to work on the human "Electroenkephalogram". His motivation for EEG research was his quest for mental energy and the possibility of transmitting thoughts. He was never able to prove his initial theory, but he did work on the cerebral circulation in the dog's brain. The experiments described in his 1929 report on alpha and small beta waves as well as alpha blocking were conducted with a Siemens double coil galvanometer. Berger was able to achieve a sensitivity of 130 $\mu V/cm$, using nonpolarizable pad electrodes. It goes without saying that Berger was aware of the work done by Caton, Cybulski et al.

Berger was able to simultaneously perform an ECG and a single channel bipolar fronto-occipital EEG recording. He used time markers with durations between 1 to 3 minutes.

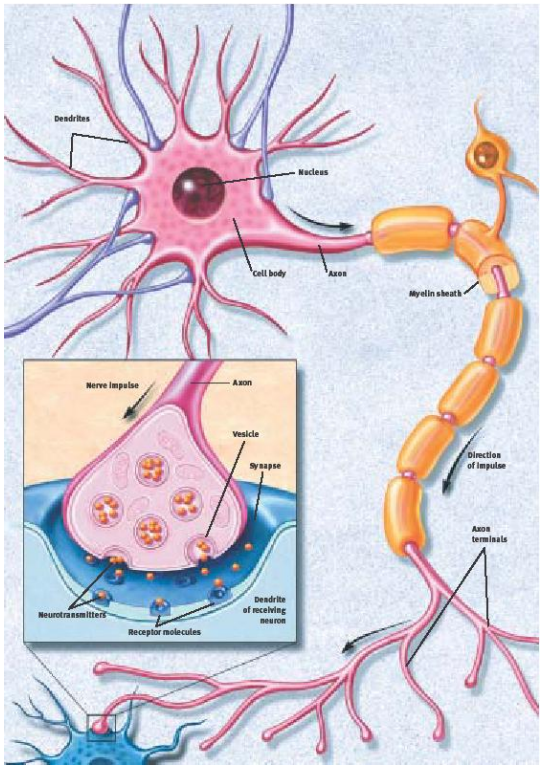
His research yielded a plethora of interesting findings. The acceptance of his work, however, was slow due to an unattractive title ("On the Electroencephalogram of Man") and a rather cumbersome and hard to read text. The recognition he deserved came late, when his work was confirmed by Adrian and Matthews in 1934. In 1937 he presented his results at conferences in Paris and Bologna held to celebrate the birthday of Galvani.

In later reports, Berger describes fluctuations of consciousness sleep EEG (sleep spindles), hypoxia of the human brain, brain disorders and epileptic discharges.
Berger's bad relations with the Nazi regime lead to an early and dishonourable discharge in 1938. Being cut off his work, he evidently developed a severe depression and committed suicide on June 1, 1941, at the age of 68.

The progress in EEG research was slowed down during World War II. In the 1950s Europe caught up with American research and EEG devices had found their way into hospitals laboratories and private practice. Most diseases of the central nervous system have EEG correlates, leading to neurologists as well as psychiatrists showing great interest in EEG research.
In the 1950s, scientists continued what Berger had started. In 1953, Aserinsky und Kleitman at the University of Chicago conducted sleep EEG experiments and observed rapid eye movements and their corresponding REM stages for the very first time.
In 1965 Cooley and Tukey presented their FFT algorithm, making it possible to analyze the power spectrum of the EEG. This shifted the focus of EEGs to computerization and automatic data analysis.
In the 1970s, research started to investigate evoked potentials gained considerable success. New computerized tomography and magnetic imaging technologies had emerged since the 1970s, but none of these technologies managed to replace the EEG. They proved to be a valuable addition instead (cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva).
Today high quality EEG devices are not only portable, they are very affordable too. In practice, EEGs are used to diagnose brain diseases by correlating EEG data with appropriate imaging technology.
Berger was not able to witness the progress of his discoveries that, very much like the PC revolution in the mid 1970s, happened mostly by accident.

Neurophysiological Basis of the EEG

The Neuron



A neuron fires by transmitting electrical signals along its axon. When signals reach the end of the axon, they trigger the release of neurotransmitters that are stored in pouches called vesicles. Neurotransmitters bind to receptor molecules that are present on the surfaces of adjacent neurons. The point of virtual contact is known as the synapse.

(Text and Image cf: Brain Facts - A Primer on the Brain and Nervous System (Fourth edition, Society for Neuroscience, 2002), p.5)

Generator structures

The central nervous system is essentially made up of nerve and glia cells. The nerve cell (neuron) is the central processing unit of the brain, a specialized unit designed to propagate information to other neurons as well as to muscle and gland cells. There are different types of neurons which can be classified by their purpose and the way they process information. *Interneurons* make up the majority of the one hundred billion neurons that form the brain matter. Information received by the dendrites is fed into the soma to process the signal before

it is transmitted via an elongated fibre called the axon. The axon expands into an array of axon terminals each having a virtual endpoint, the synapse, that transfers the information to adjacent neurons (their dendrites) by releasing amounts of neurotransmitters stored within the vesicles in each synapse.

Afferent neurons translate information from outside (tissue and organs) into the CNS whereas *efferent neurons* work in the opposite direction.

The information that is conveyed along an axon is called *action potential*, i.e. a temporary change of the cell's membrane potential. This is believed to be caused by a gain in the permittivity for Na^+ions. As a result, the negative resting potential rapidly changes from -50 to -90mV to about +30mV (cf: Roche Lexikon Medizin, 2003). This overshoot is then followed by closing the Na^+channels and opening the K^+channels. After this repolarization, the cell falls back to its resting potential. If an action potential reaches an excitatory synapse, an excitatory postsynaptic potential (EPSP) occurs in the adjacent neuron. If two action potentials travel along the same fibre in rapid succession, we get a summation of EPSP after reaching the membrane threshold. In the case of an inhibitory synapse, hyperpolarization occurs and we get an inhibitory postsynaptic potential (IPSP)

The second cell type, the glia cells, are non-neuronal cells that lie between the nerve cells and provide support and nourishment as well as maintaining homeostasis and secretion of myelin to cover the axons of the neurons. Myelin, discovered 1854 by Rudolf Virchow (1821-1902), is a lipid membrane that electrically insulates each neuron.

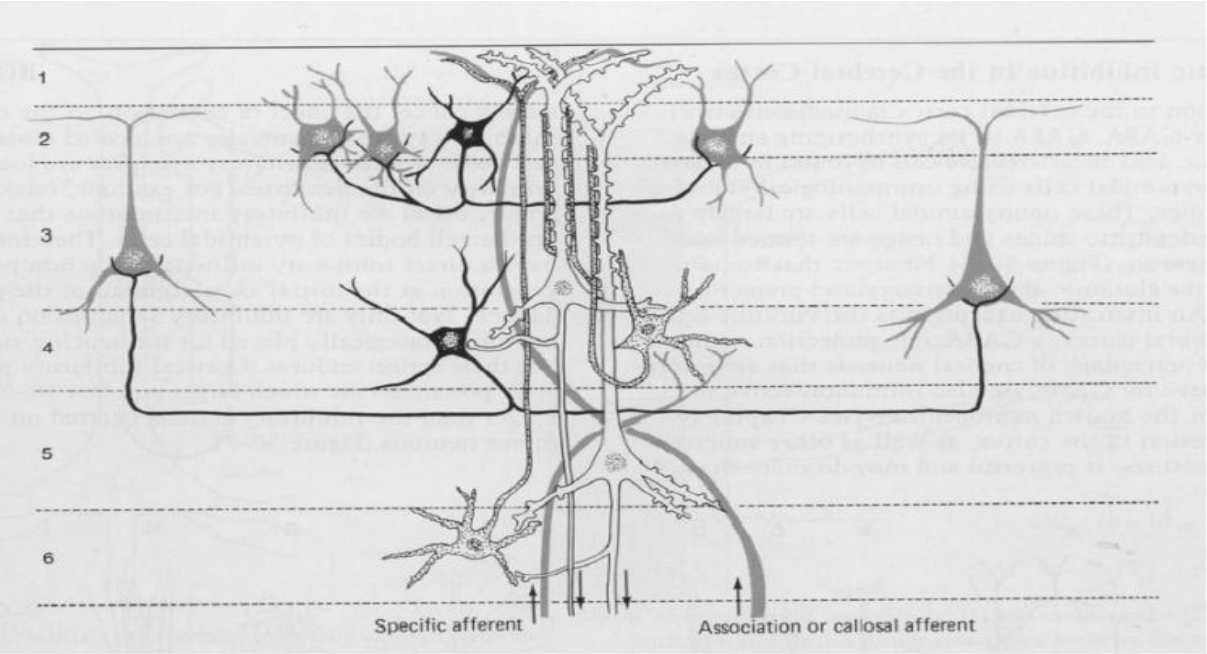
Glia cells also participate in the signal transmission in the CNS and play a role in the generation of extra-cellular field potentials (cf: Kuffler and Nicholls, 1966; Somjen and Trachtenberg; Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva).

The conglomeration of the two cell types we are most interested in with respect to EEG is the neocortex and its connection to the thalamus. It is only found in mammals and exhibits a six layer laminar histological design. Layer I is the outermost and layer VI is the innermost layer. (cf: Brain Facts A Primer on the Brain and Nervous System (Fourth edition, Society for Neuroscience, 2002)

The laminar structure in the neocortex

Layer	Description
I	Molecular layer: Lamina molecularis s. zonalis s. plexiformis, Stratum moleculare (Neuronum horizontale) Contains extensions of apical dendrites and horizontally-oriented axons.
II	External external granular layer II: Lamina granularis externa s. corpuscularis, Stratum granulare externum (Neuronum pyramidale parvum) Contains small pyramidal neurons and numerous stellate neurons.
III	External pyramidal layer III: Lamina pyramidalis externa, Stratum neuronorum pyramidalium externum (Neuronum pyramidale magnum) Contains predominantly small and medium sized pyramidal neurons, as well as non-pyramidal neurons with vertically oriented intracortical axons.
IV	Internal granular layer IV: Lamina granularis interna Contains different types of stellate and pyramidal neurons. It is the main target of thalamocortical afferents as well as intra-hemispheric corticocortical afferents.
V	Internal pyramidal layer V:Lamina ganglionaris s. pyramidalis interna **and** pyramidalium internum (Neuronum pyramidale magnum internum, medium u. magnum) Contains large pyramidal neurons (They are called Betz cells in the primary motor cortex) (Vladimir Alekseyevich Betz, 1874)
VI	Multiform layer VI: Lamina multiformis, Stratum neuronorum fusiformium (Neuronum fusiforme parvum u. medium) Contains few large pyramidal neurons and many small spindle-like pyramidal and multiform neurons Layer VI establishes an efferent connection to the thalamus.

(cf: Roche Lexikon Medizin, 2003)



Laminar structure in the neocortex. Denrites in layer 1; Pyramid cells in layers 3 and 4.
(Martin, J.H. (1991). The Collective Electrical Behavior of Cortical Neurons: The Electroencephalogram and the Mechanisms of Epilepsy. In E.R. Kandel, J.H. Schwartz & T.M. Jessell), *Principles of neural science* (Third edition., p. 777-791). Connecticut: Appleton & Lange.)

The vertical alignment within the neocortex and the fact that ions flow outside as well as inside the neuron causes a current distribution resulting in charge displacements and potential differences, which are usually modelled by several dipoles at different positions within the brain. The electrical fields superimpose linearly and spread through the brain and the skull as electromagnetic waves, where they can be picked up by the EEG electrodes.

The EEG signals (potential fluctuations) mostly originate from inhibitory (IPSP) and excitatory (EPSP) postsynaptic processes within the pyramid cells located in the neocortex (layer III and IV). However, field potentials emerging from subcortical processes are also part of the EEG.

Technological Basis of the EEG

For recording the electrical activities emerging from the brain tissue, the electrodes are placed on the scalp, the cortex surface, inside the brain and even into a single nerve cell. Various types of electrodes are in use, depending on the recording location and the test subject's or patient's behavioural situation.

The recording quality is determined by the material and the way the contact is established. In the clinical (surface) EEG, the electrodes are mostly made out of Ag/AgCl and the connection with the scalp is usually not a direct one. The gap between the electrode and the skin is established by an electrolyte bridge formed by an electrode gel that is applied between the electrode and the skin. The material of the electrode, the type of gel and the condition of the skin heavily affects signal quality.

DC Offset Voltage

When an electrode is placed in contact with a conducting solution and no current is flowing, an electrical potential difference builds up, caused by metallic ions flowing from the electrodes' surface into the solution, from the solution into the metallic surface and from there into the surface of the electrode. The excess of the ions establish an electrical double layer (dynamic equilibrium), called the Helmholtz layer. The value of the potential difference depends on the material of the electrode, the chemical properties of the gel as well as their temperature. The value ranges from millivolts to volts. The In the case of scalp electrodes, the DC potential is formed at the skin/electrolyte bridge. Note, that during measurements the actual DC offset for a single electrode is always the sum of the DC offset of that particular electrode and the DC offset of the ground, the reference and the measuring electrode.

We are unable to get rid of DC offsets. Depending on the electrode material, e.g. steel, these offsets can become quite large compared to the electrical activity recorded from the brain. Ag/AgCl electrodes produce a much smaller offset than electrodes made of steel, which exhibit high resistance to DC currents. As long as the DC offset remains stable over a long period (up to several hours) this does not pose a problem; we simply measure the offset beforehand and then subtract it from the signal during measurement With steel electrodes, the offset changes over time, producing a (DC) drift, which is very hard to come by. Measurement of the DC offset is usually done during calibration of the EEG apparatus, after application of the electrodes. For longer EEG measurements, it is important to wait for the electrodes and gel to reach skin temperature and the equilibrium to build up.

Impedance

When DC voltage is applied between electrodes in a conductive electrolyte solution, the dynamic equilibrium is disturbed and current flows. The value of the current depends on the amount of ions moving within the metal/electrolyte junction.

This particular type of frequency dependent resistance is caused by the ion diffusion described above and is called Warburg Impedance. Warburg Impedance becomes smaller with an increase in frequency, since the distance the ions moving within the alternating field decreases. It becomes high at low frequencies since the ions' movement distance increases. For EEG measurements we would like to have electrodes that exhibit a perfectly unhindered exchange of charge. Such an electrode is called a *perfectly* nonpolarized (reversible) electrode, although real electrodes do exhibit some change in the electrode/electrolyte potential due to passing current. The commonly used Ag/AgCl electrode would be an example of a *nonpolarized electrode*.

When using (*nonpolarized) EEG electrodes*, impedance should not exceed 10 kOhm. With proper skin preparation, clean electrodes and electrolyte gel recommended by the manufacturer of the electrodes, impedances of just 1 kOhm or less can be easily achieved. Waiting some time for the equilibrium to build up and stabilize and the electrodes and gel to reach skin temperature will give low impedances that will remain stable over long periods.

Artefacts

Artefacts are caused by changes in the electrical double when moving the electrodes after application. The material Ag/AgCl produces a small and stable potential which mitigates these artefacts. To reduce artefacts , we have to ensure that the electrodes do not move after their application and that they are not in direct contact with the skin.
A common solution to his problem is "glueing" the electrodes in place by using a special paste that will become hard and keep the gel underneath moist for a long time. This solution works well, but is very uncomfortable for the patient, especially on hairy regions of the skin.

Another, more comfortable solution is to use an EEG cap, which is worn by the patient or test subject. Theses caps have holes at anatomically correct places and a mechanism to hold the electrodes in place, so they do not move during the measurements. These caps also assure that electrodes are not in direct contact with the skin. The gap between the skin and the electrode is filled with a generous amount of electrode gel. For sleep EEGs or for measurements where the subject is likely to move, the cap can be additionally secured by attaching it to a chest belt. Attaching the cables to the chest belt with some tape will help to prevent accidental pulling.

(cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva, p.127, 128**)**

Signal Attenuation Accuracy

Let Z_{el_t} be the sum of all electrode impedances and let t denote the time and let Z_{cl}be the shunt capacitance of the cable connection between the electrodes and the amplifier. Let also Z_{in} be the amplifiers input impedance with noise current i_n(the Interfering and unwanted electrical current) and voltage sources e_n inherent in the amplifier. The noise voltage at the input of the amplifier is taken into account by e_{nt}. Furthermore, the input voltage of the amplifier is given by e_{in} and the electrode open circuit voltage, the electrical potential difference between the amplifiers terminals (the electrodes), is given by e_{sig}

The signal attenuation is then:

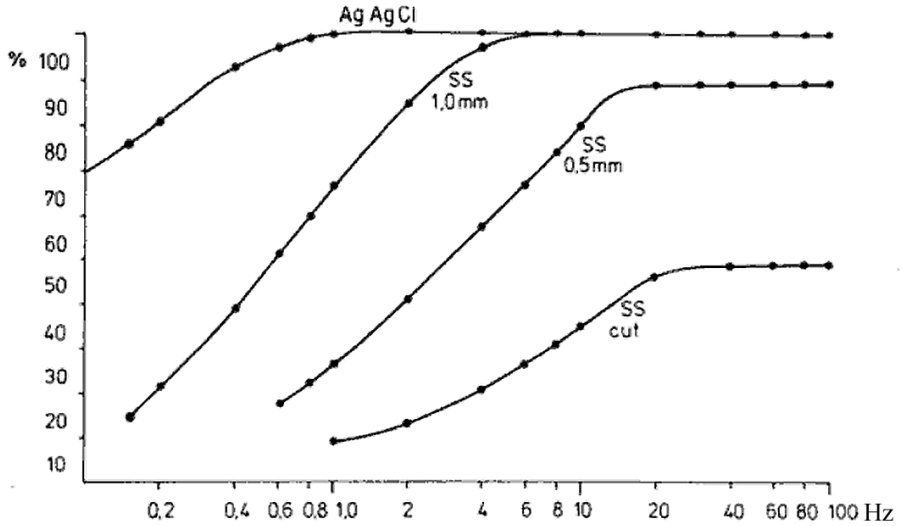
$$\frac{e_{in}}{e_{sig}} = \frac{1}{1 + \frac{Z_{el_t}}{Z_{in}} + \frac{Z_{el_t}}{Z_{cl_t}}}$$

It becomes obvious why the input impedance of the amplifier has to be at least in the megaohm range for impedances of the electrodes in the kOhm range. For a very small attenuation of 0.99, Z_{in} and Z_{cl_t} must be larger than 100 times Z_{el_t}. An input impedance of several megaohms for the EEG amplifier is therefore needed to prevent observable signal attenuation. e_{in} approximates e_{sig} if the electrode impedance Z_{el_t} is sufficiently low compared to the input impedance of the amplifier given by Z_{in} and the cable impedance Z_{cl_t}. The problem becomes even more obvious for electrodes that have a high impedance and/or frequency dependent impedance. For high cable capacitances the shunting impedance Z_{cl_t} causes considerable attenuation at high frequencies for electrodes with high impedance such as e.g. steel electrodes. A solution to this problem is an active electrode, where the amplification is performed in the electrodes' box as close as possible to the skin surface.

The accuracy of the measurements depends on the signal to noise ratio and is given by

$$\frac{S}{N} = \frac{e_{in}}{e_{nt}}$$

with e_{in} being independent of the connection between amplifier and electrode. The noise current source however affects $Z_{el_t}/Z_{cl_t}/Z_{in}$ and therefore the value of the resulting noise voltage does depend on the electrode impedance Z_{el_t}.



Example of the effect of the charactersitics of different electrodes and amplifier input impedance on the recording sensitivity. The amplitudes recorded at various frequencies, using stainless steel wire electrodes (ø = 100µm) with different tip sizes, are compared to the amplitudes recorded with an AgAgCL electrode (ø = 500µm). Amplifier input impedances: 6MΩ in parallel with a capacitor of 270 pF.

(cf: Electroencephalography Fifth Edition Niedermeyer,Lopes Da Silva, p.128, 129)

Interference from Electrical Sources

Artefacts can also arise from capacitive coupling C_{cm} between the subject's body and the power mains conductors. The alternating current i_{cm} (frequency 50 or 60 Hz) flows via the GND-electrode $Z_{el_{GND}}$ of the subject to GND, resulting in an alternating voltage

$$e_{cm}=i_{cm} * Z_{el_{GND}}$$

between the ground and the subject's body. The result is an alternating potential equal in amplitude and polarity present at both recording electrodes with respect to GND. Naturally, it is of interest that the interference caused by the magnitude of the resultant AC potential difference e_{diff} at the amplifiers input should be kept at a minimum level with respect to the amplitudes of the EEG-signal e_{sig}. The two parallel impedances developed by the amplifiers input impedance Z_{in} and the cable capacitance impedances Z_{cl} are assumed to be large,

compared to the electrode impedances $Z_{el_1}$and Z_{el_2} (considering just two electrodes in this example). Thus, e_{diff} can be approximated by:

$$e_{diff} \approx \frac{Z_{el_1} - Z_{el_2}}{Z_{cl} * Z_{in}/(Z_{cl}+Z_{in})} * e_{cm}$$

It is quite obvious that in order to minimize e_{diff} the electrode impedances must be small as possible. Observing different electrode differences, a common mode voltage e_{cm} with an amplitude of several millivolts, the parallel impedance Z_{cl}, Z_{in} must be at a magnitude of several megaohms to keep e_{diff} sufficiently small. Overall, the problem is similar to the "*signal attenuation*"-problem. A solution to this problem are active electrodes, as the mains' interference is not affected by cable capacitance anymore.

Interference from magnetic fields

Changing magnetic fields, emitted with low or high frequency, will also induce voltages e_{diff} in the electrode cables.

$$e_{diff} = B * A * 2 * \pi * f$$

A is a loop area between the cables while f denotes the frequency of the magnetic flux density B. A way to reduce the magnetic coupling into the signal is to reduce A by twisting the electrode cables. This also reduces the crosstalk between neighbouring pairs. Additionally, the electromagnetic interference can be reduced by shielding the cables. The shielding is made out of metal. Shielding applied to a collection of pairs, not individually to each pair, is called screening. In order to work, the shielding must be grounded. The magnitude of common components of the potentials at the electrodes can be comparatively large with respect to the components that differ at the electrodes. Therefore, the amplifier should be made more sensitive to the signal components that are different at the electrodes, a principle called *common mode rejection*.

Oversampling and filtering

The classical slow EEG covers frequencies from 0.1 to 60Hz and sometimes up to 80Hz. According to Shannon-Nyquist a sampling rate of about 160Hz would be sufficient for perfect reconstruction. But this calls for nearly ideal analog antialiasing and reconstruction filters with extremely sharp cut-off frequencies to stay within the Nyquist limit. Those filters are very difficult to implement, and therefore very expensive. Oversampling increases the

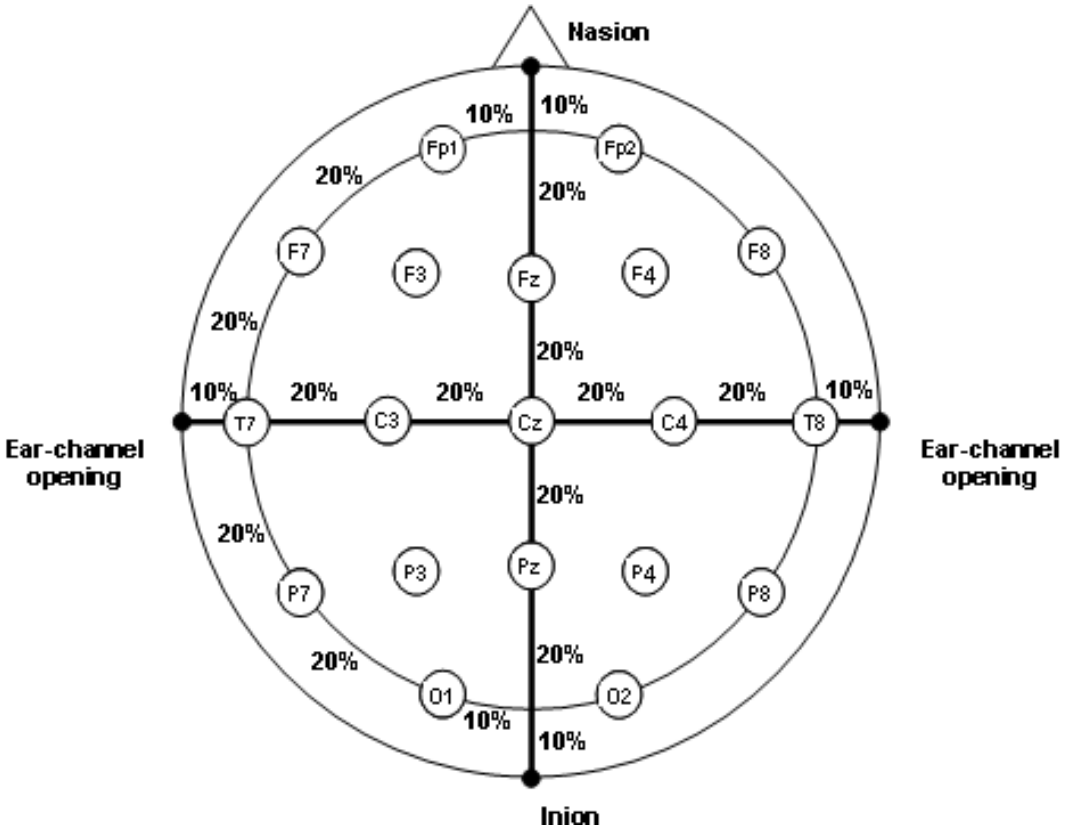
bandwidth of the sampled signal. We can therefore use digital filters and simple, inexpensive analogue filters at the output. In modern amplifiers with up to 256 channels, the computational load can easily be tackled by inexpensive, yet extremely powerful, digital signal processors. Without going into details, oversampling yields to a better *signal to noise ratio* through averaging. It also leads to better linearity, i.e. the relation between the input and the output signal within the working amplitude range is constant.

EEG Recording

When Berger did his experiments he was using a fronto-occipital electrode montage. Berger saw the EEG as a measure of global cortical activity; later it was discovered (cf: Adrian and Matthews,1934; Adrian and Yamagiwa 1935; Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva p.139), that this was not the case and that the EEG activity varies in different locations. To ensure reproducibility, a system was needed to be used by researchers and everybody doing EEG measurements. With a standardized system, there would be no need to record electrode positions for each patient individually. The 10-20 system pictured below made recording and comparing results a lot easier. It was recommended in 1958 by a committee of the International Federation of Societies for Electroencephalography and Clinical Neurophysiology.

The 10-20 system contained recommendations for electrode placements that did not rely on distance measurements in mm or cm, worked equally well on babies, small children and adults and sufficiently covered the underlying functional topography of the cortex.

The 10-20 system uses specific anatomical landmarks, the nasion and the inion. Between these two landmarks, measurements are made in either 10% or 20% intervals of their total distance. This marks the distance that electrodes are placed from one another. The resulting electrode distribution can be seen in the picture below:



Standard montage of electrodes according to the 10-20 system
(http://www.easycap.de/easycap/_img/content/ea_DL_10_20.gif)

Reference and GND

For grounding the EEG system, its ground connector is connected to a ground electrode which is usually placed on the forehead. However, the position is irrelevant and could be anywhere on the body e.g. a wristband. The GND electrode is used for common mode rejection to get rid of ground referred interference.

The purpose of the reference electrode is to provide a signal that is neutral to cortical activity. Common places for the reference electrodes are the earlobes or the mastoids. Unfortunately, in epilepsy temporal lobe abnormality may be present in as many as 70% of the patients (cf: Electroencephalography Fifth Edition Niedermeyer,Lopes Da Silva, p.143**).** To avoid the temporal area, the reference electrodes can also be placed on the nose. A more complicated solution is to use two noncephalic reference electrodes connected with a 20kOhm resistor. One electrode is then placed over the right sternoclavicular junction and another one over the spine of the first thoratic vertebra to virtually cancel out ECG influences. (Stevenson and Gibbs, 1949; cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva, p.143).

Unipolar and Bipolar Derivation

All the electrodes are measured against the reference electrode and the GND. The reference electrodes themselves are also measured against GND. Especially when storing the EEG to hard disk for later analysis, a unipolar recording is advised, because in a bipolar recording, electrodes are measured in reference to other electrodes. E.g. $P_3$is measured in reference to O_1. In a bipolar recording:

$$P_3' = P_3 - O_1$$

gives the new value for P_3.

When using a unipolar recording, these calculations can be made afterwards without changing the precious raw data. Bipolar recording schemes are used to detect specific diseases and their EEG correlates as well as improving focal localization (Hjorth 1975, 1976, 1979). This technique, its variants and improvements are often used in examination of evoked potentials

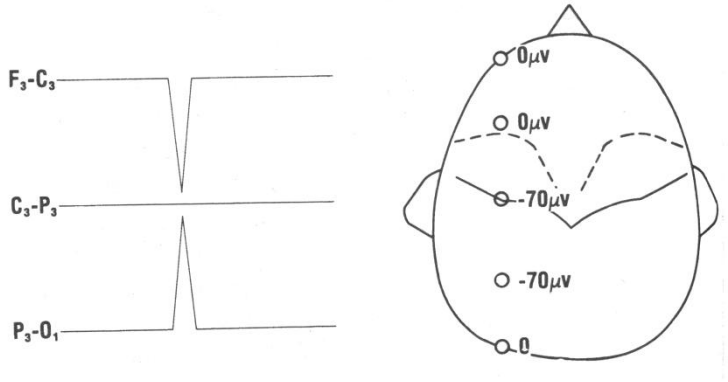
As a consequence of the calculations involved in bipolar recordings, a phenomenon called *phase-reversal* can be observed.

Let's consider the following scheme:

$$F_3' = F_3 - C_3$$

$$C_3' = C_3 - P_3$$

$$P_3' = P_3 - O_1$$



This scheme can result in a phase reversal with equal surface negative activity in the C_3 and P_3 electrodes with no deflection (a peak in the up or downward direction) in that channel. The major deflections can instead be observed between the uninvolved electrode F_3 and the heavily involved C_3 electrode as well as between the uninvolved O_1electrode and the heavily involved parietal electrode P_3.

(cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva, p.144).

(cf: Hjorth, B. An on-line transformation of EEG scalp potentials into orthogonal source derivations. Electroencephalography and Clinical Neurophysiology 39: 526-30, 1975.)

(cf: Hjorth, B. (1976). Localization of foci in the scalp field. In Quantitative Analytic Studies in Epilepsy, ed. Kellaway, 0. & Peterson, I., pp. 483-492. New York: Raven.)

(cf: Hjorth, B. (1979). Multichannel EEG preprocessing. Analogue matrix ... in the study of local effects. Pharmakopsychiatr. Neuro-Psychopharmakol. 12:111-118)

The Human EEG

In general, the clinical EEG uses five different frequency bands called Alpha, Beta, Gamma, Delta and Theta. Additionally, the Mμ rhythm is mentioned because of its relevance to Brain Computer Interfaces. The pi rhythm (posterior slow waves of 3 to 4Hz) has hardly been used since the 1990s (cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva, p.167) and is therefore omitted. The cut-off frequencies are not standardized and depend on the age and the alertness of the subject. The amplitudes are average values.

Alpha waves

Rhythms of 8-13Hz occur during wakefulness over the posterior regions of the head, enerally with higher voltage over the occipital areas.

The amplitude is variable but mostly below $50\mu V$ in adults. It is best seen with the subject's eyes closed and under condition of physical relaxation and relative mental inactivity. Alpha waves are blocked or attenuated by attention, especially visual and mental effort (cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva, p.168; IFSECN, 1974)

Alpha waves are believed to arise from somewhere within the thalamic region. Stimulus-induced alpha-blocking or attenuation is conceived as an "event related EEG desynchronization"

(cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva, p.170)
(cf: IFSECN. A glossary of terms most commonly used by clinical electroencephalographers (International Federation of ... Neurophysiol., 1974, 37: 538– 548)

Mμ rhythms

These rhythms are encompassed by the alpha band covering the range of 10Hz to 12 Hz. They are believed to reflect desynchronization of large amounts of pyramidal neurons within the motor cortex during contralateral motor acts.

"Mμ rhythm is not detectable in every mature subject […] its prevalence is limited unless the "hidden" mμ rhythm is visualized with special methods. Mμ stands for motor; this rhythm is strongly related to functions of the motor cortex but the contribution of the adjacent somatosensory cortex must not be ignored."
(cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva, p.174).

Beta waves

Beta waves cover the range of 14-30Hz and designate a state of waking consciousness. They are usually *not* rhythmic in healthy, sober subjects and reach amplitudes of up to $30\mu V$ in the fronto parietal region. Rhythmic beta waves are associated with several pathological conditions. They can also be increased by giving sedative or hypnotic drugs e.g. benzodiazepines or barbiturates. Absence or partial absence can indicate cortical damage.

Gamma waves

These waves can be observed above 30Hz throughout the whole cortex with an amplitude of up to 200 μV. The upper limit is not defined. Frequencies from 36-40Hz reach their maximum just *before* a movement onset. The ERS within the beta band have their maximum just *after* the movement onset. From 60 to 90Hz, we can observe oscillations in subdural recordings (electrocorticography ECoG) that are associated with movement. (cf: Crone et al. 1998 and Pfurtscheller et al 2003; cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva, p.1012). Gamma oscillations reflect a state of active information processing. In order to develop gamma bursts, a desynchronization within the alpha band might be necessary. Induced gamma and alpha oscillations are encompassed in desynchronized alpha band activity. A phasic increase in 40Hz event-related coherence between the contralateral sensorimotor and supplementary motor areas during brisk unilateral finger movements has been reported by Andrew and Pfurtscheller in 1996. The 40Hz coherence between the left and right sensorimotor area shows no change in coherence during the movement. Also, during the performance of cognitive and motor tasks, separate foci of synchronized gamma activities can be observed in clearly separated cortical regions, even different lobes. Rodriguez el al.(1999) observed gamma activity in subjects viewing either faces or meaningless shapes.

(cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva, p.1012)

(cf: Crone NE, Miglioretti DL, Gordon B, Sieracki JM, Wilson MT, Uematsu S, Lesser RP.Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis I. Alpha and beta event-related desynchronization. *Brain* 1998;121:2271-2299)

(cf: Crone NE, Miglioretti DL, Gordon B, and Lesser RP. Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis II. Event-related synchronization in the gamma band. *Brain* 1998;121:2301-2315.)

(cf: Pfurtscheller G, Graimann B, Huggins J E , Levine S P, Schuh L A. *Spatiotemporal patterns of beta desynchronization and gamma synchronization in corticographic data during self-paced movement.* Clinical Neurophysiology, 114(7):1226-1236 (2003))

(cf: Rodriguez, E., George, N., Lachaux, J. P., Martinerie, J., Renault, B., Varela, F. J., 1999, Perception's. shadow: long-distance synchronization of human brain activity. Nature 397:430-433)

Theta waves

Theta waves, introduced in 1944 by Walter Dover, denote the range between 4 to a frequency below 8Hz (IFSECN,1974), 7.5Hz that is. The origin is presumed to be the thalamic region, hence the name theta (Knott, 1976b). Its maximum amplitude is about $50\mu V$ in the parietal area. In the waking adult, the amount of theta waves and organized theta rhythms is rather small, in children, however, they can be observed in states of drowsiness and sleep. (also cf: Chapter 11 "Maturation of the EEG: Development of Waking and Sleep Patterns" Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva) "Hedonic Theta Rhythms", associated for the first time by W. Grey Walter in 1959 with emotions, are believed to relate to the "relative of maturity of the mechanisms linking the cortex, the thalamus and the hypothalamus" (cf: Knott 1976b).

Theta responses encompassed in the range of 5 to 6Hz and with an origin at either the occipital region or the vertex have been observed by Nowak and Marczynski (1981) who stressed 24 subjects to put them into a state of maximum alertness. Rhythmical activity between 6 and 7Hz over the frontal region has been correlated with mental activity such as solving problems (Arellano and Schwab, 1950; Brazier and Casby, 1952; Ishihara and Yoshii, 1972; Mizuki 1982; 1987; Mizuki et al., 1980; 1983). The difficulty in correlating these rhythmic responses with mental activity has also been demonstrated by Mizuki as well as by Niedermeyer et al in 1989.

(cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva p. 183).

(cf: Brazier MAB and Casby JU: Crosscorrelation and autocorrelation. studies of electroencephalographic potentials. Electroencephalogr. Clin Neurophysiol 4:201-211)

(cf: Ishihara T, Yoshii N (1972) Multivariate analytic study of EEG and mental activity in juvenile delinquents. Electroencephalogr Clin Neurophysiol 33:71–80)

(cf: Knott., J.R. The theta rhythm. In Remond, A. et al. eds. Hand-. book of Electroencephalography Clin. Neurophysiol., Amsterdam,. Elsevier. 1976, 6A: 69-77)

(cf: Nowak, S. M., & Marczynski, T. J. (1981). Trait anxiety as reflected in EEG alpha response to stress. Electroencephalography & Clinical Neurophysiology, 52:175-191)

(cf: Arellano AP and Schwab RS. Scalp and basal electroencephalogram during mental activity. Int Congr Psychiat, Paris, 1950, p. 216-219)

(cf: Mizuki, Y., Tanaka, O. Isozaki, H., et. al . 1980. Periodic appearance of theta rhythm in the frontal midline during performance oa a mental task. Electroencephalog. Clin. Neuro physiol. 49:345-351)

(cf: Mizuki, Y. 1982. Frontal midline theta activity during performance of mental tasks. Electroencephalogr. Clin. Neurophysiol. 54:25P(abst.))

(cf: Mizuki, Y., Takii, O., Nishijiama H., et. al 1983. The relationship between the appearance of frontal midline theta activity (Fm theta) and memory function. Electroencephalogr. Clin. Neuro physiol. 56:56P(abst.))

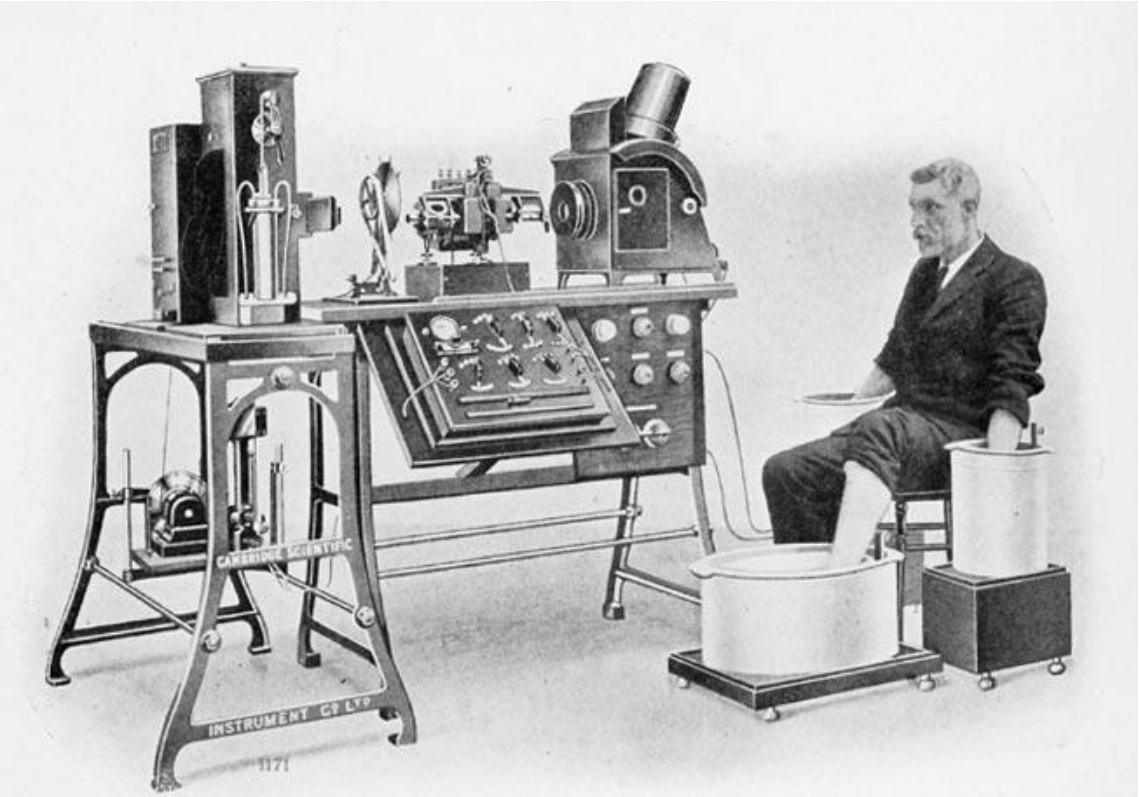
(cf: Mizuki, Y. 1987. Frontal lobe: Mental functions and EEG. Am. J. EEG Technol. 27:91-101)

(cf: Niedermeyer, E., Krauss, G.L., and Peyser, C.E. 1989. The electroencephalogram and mental activation. Electroencephalog. Clin. Neuro physiol. 20:215-226)

Delta waves

Delta activity occurs in the range of between 1 to 4Hz and reaches a maximum amplitude of up to 200 μV in the occipital area. In adults they are only present during deep sleep. Frequencies below 1Hz are called slow delta waves. The distinction is made because of the two origins and mechanisms of delta rhythms. The faster waves emerge from the thalamus, whereas the slow waves are believed to emerge from the cerebral cortex. This has been shown by performing a thalamectomy. The difference between the slow delta and faster delta waves during human sleep has been investigated by Achermann and Borbély in 1997. (cf: Electroencephalography Fifth Edition Niedermeyer, Lopes Da Silva p. 55, p. 167).

The History of ECG



PHOTOGRAPH OF A COMPLETE ELECTROCARDIOGRAPH, SHOWING THE MANNER IN WHICH THE ELECTRODES ARE ATTACHED TO THE PATIENT, IN THIS CASE THE HANDS AND ONE FOOT BEING IMMERSED IN JARS OF SALT SOLUTION

This is an early model of an ECG device. Note, that despite the saline solution instead of electrodes, the measurement is performed in the exact same way it is done today. The three buckets simply form Einthoven's Triangle. (http://en.wikipedia.org/wiki/Image:Willem_Einthoven_ECG.jpg)

In 1843 Carlo Mateucci performed animal experiments with pigeon hearts and observed that the heart works electrically. Rudolf von Koelliker and Heinrich Müller discovered for the first time in 1856 that the heart generated electricity. The first successful recording of electrical rhythm in the human heart was performed by Alexander Muirhead in 1869–1870 at St. Bartholomew's Hospital, London. The equipment he used was originally made to record signals passing through the transatlantic cable, which had been laid in 1866. In 1876 E. J. Marey was able to visualize his findings using a Lippmann capillary electrometer. The same device was then used by Augustus Waller at St. Mary's Hospital, Paddington, London in 1887 to record electrical reactions of the human heart. Waller published his work on cardiac electricity on the body's surface and called the recording a "cardiograph". He presented a paper titled "A preliminary survey of 2,000 electrocardiograms" to the Physiological Society of London in 1917. His contributions include the variability of the electrogram as well as the dipole concept. Einthoven, the „father of electrocardiography", attended the International Congress of Physiology in London and observed Wallers demonstration. In 1895, he detected

recognizable waves, which he labeled “P, Q, R, S, and T”. It is believed that he chose these letters because Descartes had used them to identify successive points on a curve. In 1903 Einthoven presented his invention, the “String galvanometer”. With the help of this new device he was able to overcome the previously used capillary electrometer to standardize the tracings. He formulated the concept of “Einthoven’s triangle” which mathematically relates three leads. Eventually in 1924, Einthoven received the Nobel Prize in Medicine. In 1934 Frank Wilson proposed a unipolar derivation scheme. By joining the wires from the left arm, right arm and the left foot with 5kOhm resistors he defined a so called *indifferent electrode* which was later referred to as the ‘Wilson Central Terminal’. The indifferent electrode acts as a GND and is attached to the negative terminal of the ECG. Any electrode connected to the positive terminal of the ECG machine is then *unipolar* and can therefore be placed anywhere on the body.

(cf: Wilson NF, Johnston FE, Macleod AG, Barker PS. Electrocardiograms that represent the potential variations of a single electrode. Am Heart J. 1934;9:447-458.)

(cf: http://www.ecglibrary.com/ecghist.html)
(cf: Klinische Kinderkardiologie, 4., überarbeitete und erweiterte Auflage Diagnostik und Therapie der angeborenen Herzfehler)

(cf:Cardiology’s 10 Greatest Discoveries of the 20th Century, Nirav J. Mehta, MD and Ijaz A. Khan, MD, FACC, Division of Cardiology, Creighton University School of Medicine, Omaha, Nebraska 68131, http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=124754)

(cf: Naming of the Waves in the ECG, With a Brief Account of Their Genesis; J. Willis Hurst, MD; Correspondence to J. Willis Hurst, MD, 1462 Clifton Rd NE, Suite 301, Atlanta, GA 30322.; *Circulation.* 1998;98:1937-1942.
© 1998 American Heart Association, Inc.)
(cf: *Journal of the History of Medicine and Allied Sciences* 1971, 26:181)
(Cf: Did Einthoven invent a string galvanometer? HB Burchell Heart 57:22, 190-193, 2/1987)

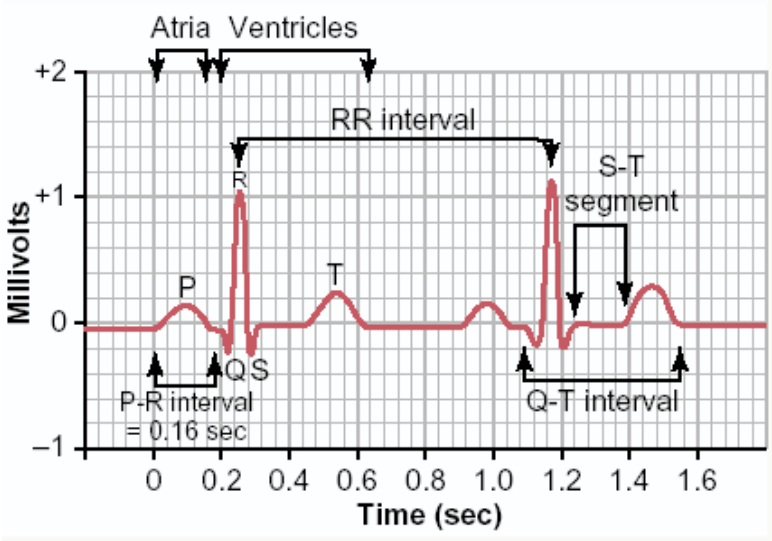
The Human ECG

The electrical impulses to cause rhythmical excitation of the heart muscle are generated by the heart itself. This excitatory and conductive system that controls normal cardiac contractions consists of the *sinus node* (sinoatrial or S-A node) which generates the normal rhythmical impulses and the atrioventricular node or A-V node which delays and conducts the impulses from the atria to the ventricles. This delay ensures full contraction of the atria before electrical stimulation of the ventricles takes place.

The sinus node is the pacemaker of the heart and is a small flattened, ellipsoid strip of specialized cardiac muscle. It is about 3 millimeters wide, 15 millimeters long and 1

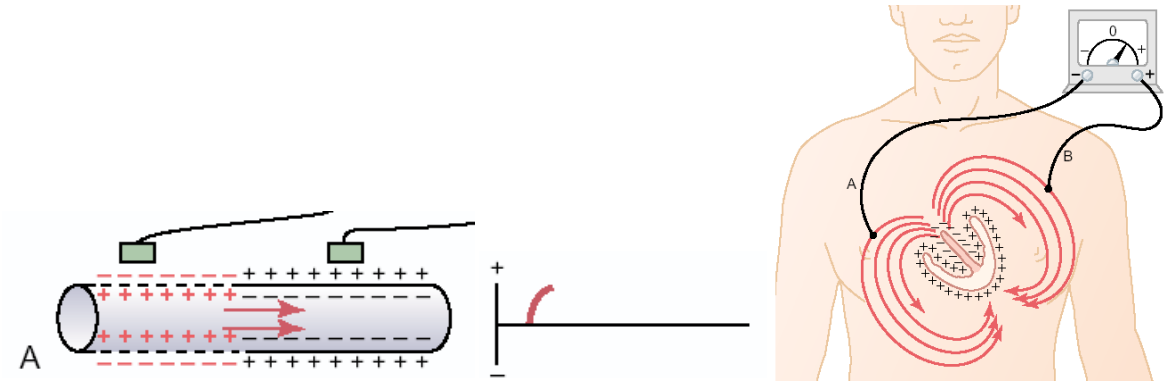
millimeter thick and is located in the upper part of the wall of the right atrium. The normal rate of the sinus node is about 70 to 80 Hz.

Cardiac impulses passing through the heart are causing electrical impulses to spread from the heart into adjacent tissues and all the way through the entire body. With electrodes located on the skin and at opposite positions of the heart we can record electrical potentials over time. This kind of recording is then called an electrocardiogram or ECG. The normal ECG consists of a P wave, a QRS complex and a T wave. The QRS complex consists of three mostly separate waves: the Q, the R and the S wave. The P wave occurs during atrial depolarization when an electrical impulse travels from the sinus node to the AV node and spreads from the right to the left atrium. The QRS complex originates from depolarization of the ventricles before contraction. The P, Q, R and S waves are therefore called depolarization waves. The T wave originates during recovery from the state of depolarization of the ventricles and is therefore called a repolarization wave. The resulting ECG wave pattern can be seen below.



Example of an ECG wave. The QRS complex is clearly visible.
(Textbook of Medical Physiology Eleventh Edition, Guyton & Hall p. 124)

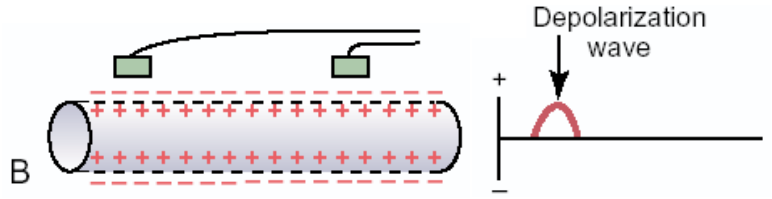
Depolarization and repolarization take place in four stages.



A muscle fibre during depolarization.

(Textbook of Medical Physiology Eleventh Edition, Guyton & Hall p. 124, 127)

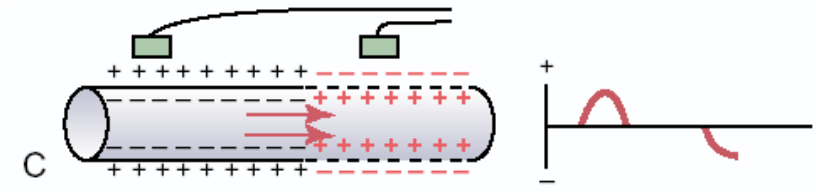
The polarity gradually changes as depolarization spreads through the muscle fibre. This causes the meter to record a positive wave.



Depolarization has completed and repolarization is about to begin.

(Textbook of Medical Physiology Eleventh Edition, Guyton & Hall p. 124)

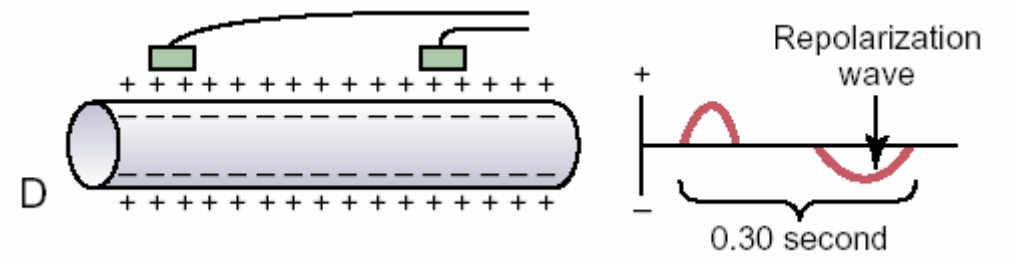
The process continues until the muscle fibre is completely depolarized and the reversal of the polarity is completed.



Repolarization is in progress

(Textbook of Medical Physiology Eleventh Edition, Guyton & Hall p. 124)

While the repolarization travels through the muscle fibre the polarity is gradually reverted to its original state. The meter records a negative wave.



The depolarization and repolarization cycle is completed.
(Textbook of Medical Physiology Eleventh Edition, Guyton & Hall p. 124)

The depolarization and repolarization cycle lasts for about 0.3 second.

Before any contraction can take place, the respective muscle fibres must be completely depolarized to start the (chemical) process of contraction. P waves therefore occur at the beginning of contraction of the atria and QRS waves at the beginning of the contraction of the ventricles. The contraction of the ventricles is released after full repolarization has occurred, that is, after the end of the T wave. Repolarization of the atria takes place about 0.15 to 0.20 seconds after the end of the P wave. The atrial repolarization wave, also called the atrial T wave, and the beginning of the QRS complex happen at the same time. Unlike the ventricular repolarization wave, the T wave, which marks the end of the QT interval, the atrial T is usually obscured by the much larger QRS complex. Ventricular repolarization starts between 0.20 and 0.35 after the beginning of the QRS complex. The process therefore lasts for up to 0.15 second making the T wave a prolonged wave with a considerably smaller amplitude that the QRS complex.

Voltages in the ECG

The amplitudes depend on how the electrodes are applied to the skin and how close to the heart. The voltage of the R wave reaches amplitudes of 3 or 4 mV when placing one electrode directly over the ventricles and a second one somewhere else on the body. However, action potentials with an amplitude of about 110 mV can be observed during recordings directly at the heart muscle. An amplitude of about 1.0 to 1.5 mV of the R wave can be observed during recordings at either the arms or one arm and one leg. The amplitude of the P wave is between 0.1 and 0.3 mV. The amplitude of the T wave is between 0.2 and 0.3 mV.

R wave amplitudes between slightly less than 1.0 mV and slightly above 1.0 mV can be observed when recording from the back of the hand or the wrist using an Ag/AgCl disc or Ag/AgCl ring electrode with the reference and GND electrode placed on the scalp. This

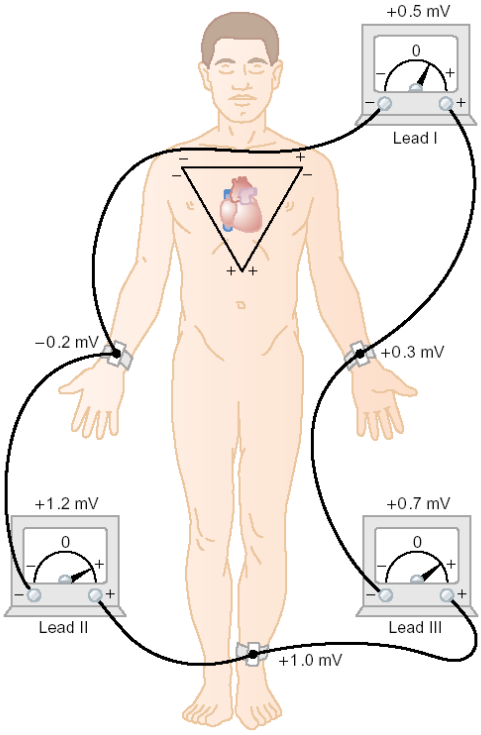
montage is commonly used during an EEG recording. It works somehow, but it is *not* a standard montage.

The time between the beginning of the P wave and the beginning of the QRS complex is called the P-Q interval. It is the time between the start of atrial excitation and the ventricular excitation and lasts for about 0.16 second. If the Q wave is absent, this interval is called the P-R interval. The Q-T interval is the duration of the ventricular contraction from the start of the Q wave (R wave, if the Q wave is absent) right to the end of the T wave. The Q-T interval is about 0.35 second.

Normal Heartbeat of the Adult

The heartbeat is the reciprocal time between two successive heartbeats or QRS complexes. In adults this time is about 0.83 second or about 72 beats per minute.

Electrocardiographic leads



A conventional arrangement of electrodes for recording a standard ECG. The triangle on the chest is called Einthoven's triangle. It is called a triangle because the left arm is connected via the upper part of the chest to the right arm. The upper part is connected to the lower part (leg) via the body.
(Textbook of Medical Physiology Eleventh Edition, Guyton & Hall p. 127)

For recording limb lead I, the negative terminal of the ECG machine is connected to the right arm and the positive terminal is connected to the left arm. Thus, the ECG records a positive

signal if the point where the right arm connects to the chest is electronegative with respect to the point where the left arm connects to the chest.

For recording limb lead II, the negative terminal of the ECG machine is connected to the right arm and the positive terminal is connected to the left leg. Thus, the ECG machine records positive if the right arm is negative with respect to the left leg.

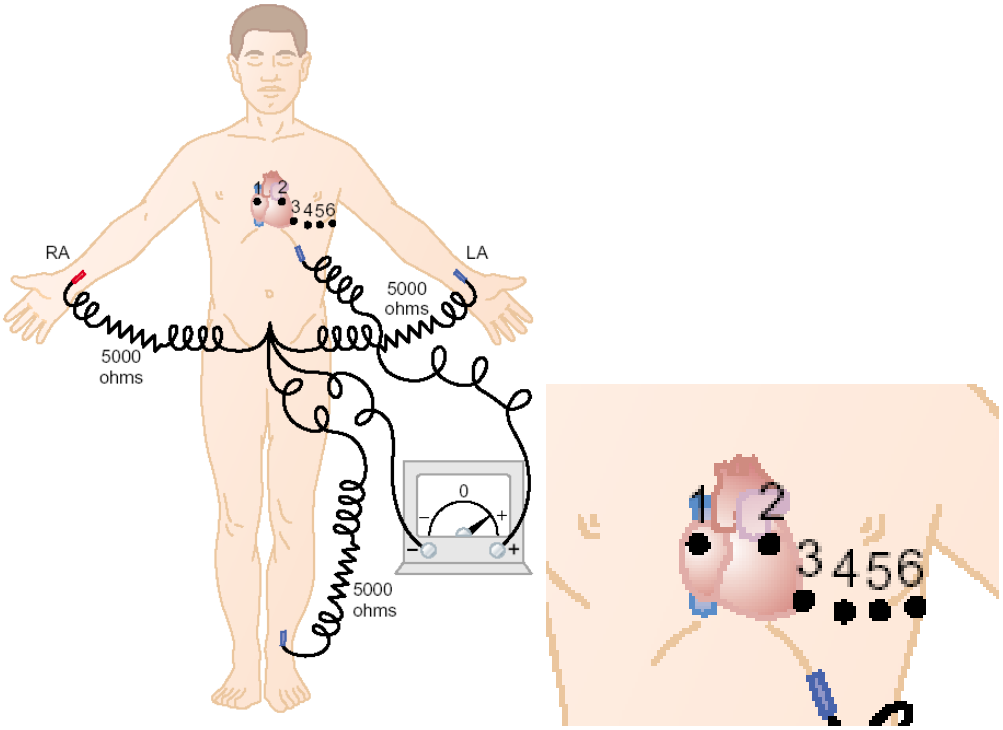
To record limb lead III, the negative terminal of the ECG machine is connected to the left arm and the positive terminal is connected to the left leg. Thus, the ECG machine records positive if the left arm is negative with respect to the left leg.

Einthoven's Law

If two of the electrical potentials out of the three bipolar limb leads are known, the third one can be computed by summing up the two known ones.

$$\text{Lead I} = \text{Lead II} - \text{Lead III}$$

Precordial Leads

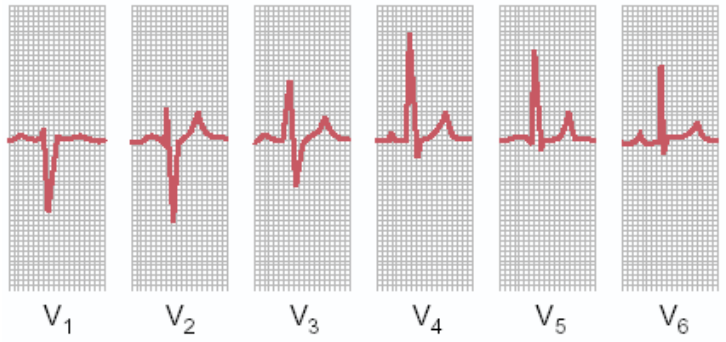


Derivation scheme proposed in 1934 by Frank Wilson. The six standard precordial chest leads are recorded from the anterior chest wall. Six recordings are performed with only one chest electrode per recording being placed at one of the six positions. These recordings are known as leads V_1, V_2, V_3, V_4, V_5 and V_6.

(cf: Textbook of Medical Physiology Eleventh Edition, Guyton & Hall p. 129)

Despite the more indirect three limb leads approach, an electrode can also be placed on the anterior surface of the chest, directly over the underlying heart at one of the six positions shown above. This electrode is then connected to the positive terminal of the ECG machine whereas the indifferent electrode is connected to the negative terminal. The derivation scheme shown above is the scheme proposed in 1934 by Frank Wilson. The six standard precordial chest leads are recorded from the anterior chest wall. Six recordings are performed with only one chest electrode per recording being placed at one of the six positions. These recordings are known as leads V_1, V_2, V_3, V_4, V_5 and V_6.

The QRS complex in V_1, V_2 are negative, because the chest electrode at these positions is closer to the base of the heart than to its apex. The heart's base is the direction of electronegativity during most of the ventricular depolarization process. In V_3, V_4, V_5 and V_6 the chest electrode is closer to the heart apex, the direction of electropositivity during depolarization. Therefore, the QRS complex is mainly positive.

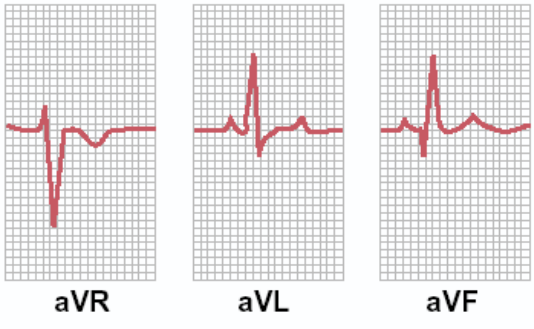


Normal ECG from the six standard chest leads

(cf: Textbook of Medical Physiology Eleventh Edition, Guyton & Hall p. 129)

Augmented Unipolar Limb Leads

The figure below shows the so called a*ugmented unipolar limb leads*. In this system, two limbs are connected through a resistor to the negative terminal of the ECG machine, whereas the third limb is connected to the positive terminal. If the positive terminal is connected to the right arm, the lead is called aVR lead. When it is connected to the left arm, it is called aVL lead and when it is connected to the left leg, it is called the aVF lead. This means that the *different electrode* of the single limb is measured against the common reference of the other two limbs, the *indifferent electrode*. The inversion for aVR is the result of the polarity connections to the ECG machine.



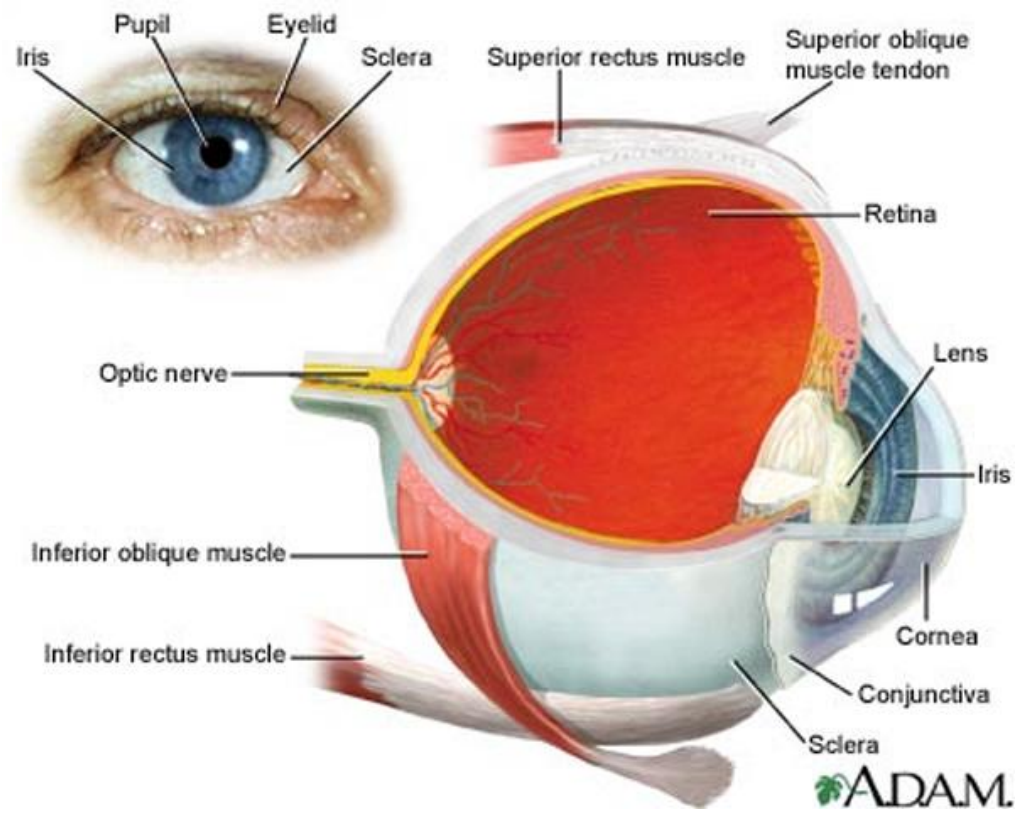
ECG from three augmented unipolar limb leads

(cf: Textbook of Medical Physiology Eleventh Edition, Guyton & Hall p. 129)

Vision and Perception

We take it for granted and it is hard to imagine not being able to see. Vision is one of the most complicated of all our senses. It is not surprising that it is also the most studied. With one fourth of our brain devoted to visual processing, it involves more brain matter than all the other senses.

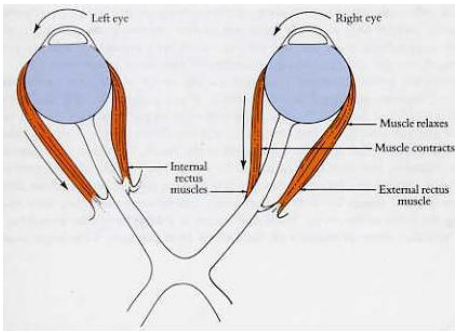
The anatomy of the eye



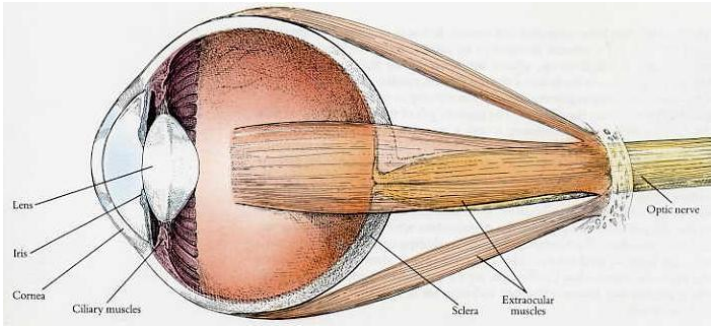
(http://www.nlm.nih.gov/medlineplus/ency/imagepages/1094.htm)

The image above shows the anatomy of the human eye. Each eyeball is being hold in its socket by six extraocular muscles. There are three pairs of muscles each working in opposition to take care of movements in one of the three perpendicular planes. To prevent seeing double, the tracking of objects has to be done with a precision of a few minutes of arc. The cornea is responsible for about two third of focusing. The lens provides the remaining third. Due to its jelly-like consistency, the shape can be altered to adjust the focus by contracting or relaxing a set of radial muscles called *ciliary muscles*. For near objects the shape is more spherical. For far objects it becomes flatter.

At about the age of 45 its ability to focus starts to degrade because the lens becomes harder. The cornea has got lots of nerves. Irritations caused by e.g. dust will set up a reflex to blink and produce tears to clean the cornea.



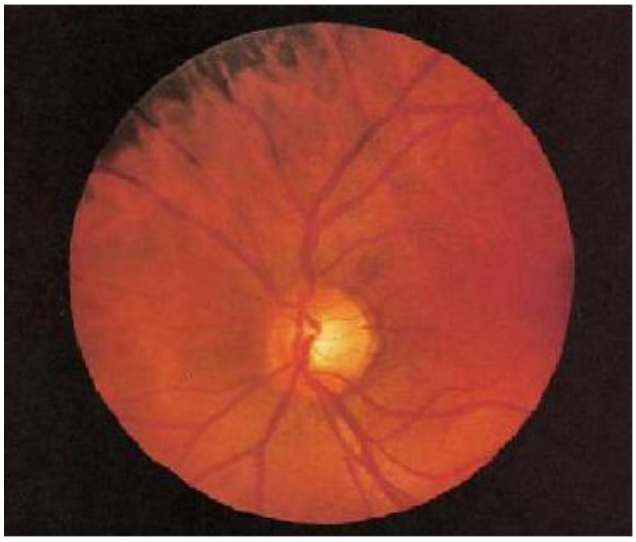
Schematic view of how the eyes are hold in their sockets. The optic nerves cross on their way to the visual cortex

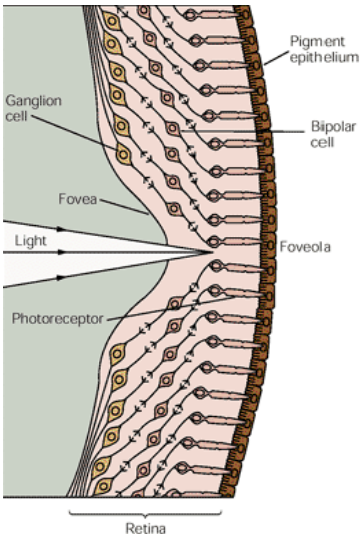


Anatomy of the eye. The ciliary muscles hold the lens to adjust the focus

(cf: Hubel D.H. Eye, Brain, and Vision [Scientific American Library-Vol.22] (Freeman, 1995) p. 28,32)

The Retina



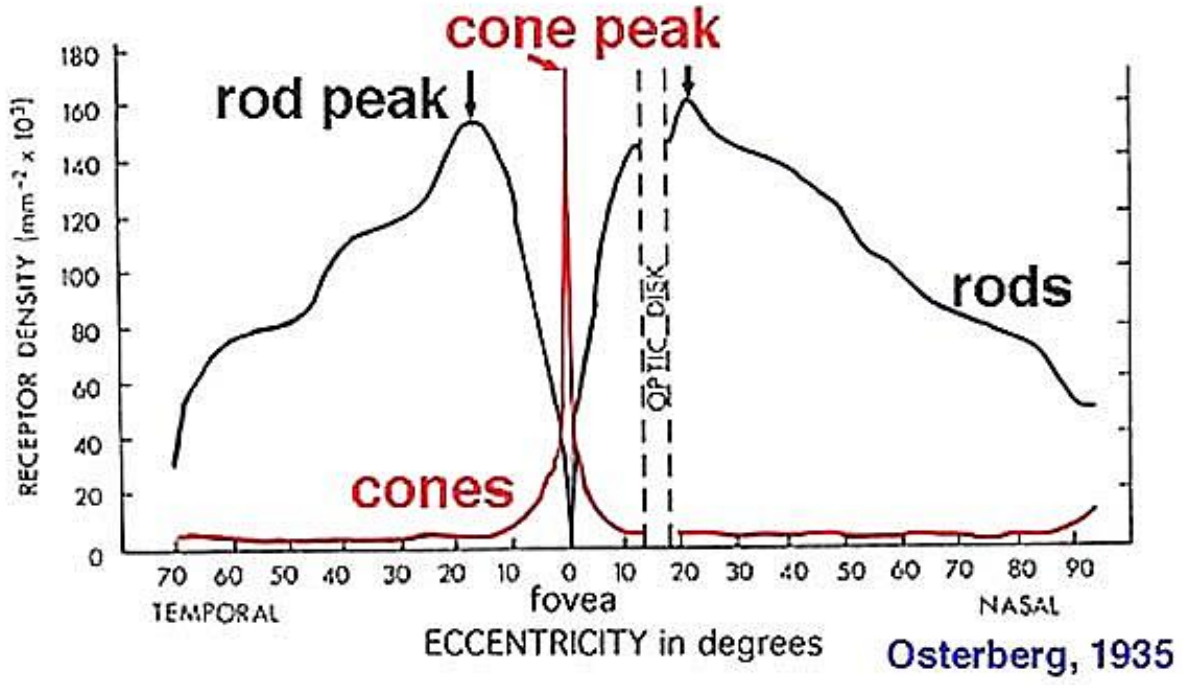


Left figure: The darker red pigmented area to the extreme right is the *macula*. The center of this region, not shown, is the fovea. The large pale area is the optic disc.
(cf: Hubel D.H. Eye, Brain, and Vision [Scientific American Library-Vol.22] (Freeman, 1995) chapter 3, p1)

(right figure cf: *Principles of neural science* (Fourth edition, P. 508). Connecticut: Appleton & Lange)

The retina is a part of the brain and is a sheet of photoreceptors that translate light into nerve signals which are then transmitted to the visual cortex via the optic nerve. The retina is about a quarter of a millimeter thick and consists of three layers of nerve cells: The rods and cones, which are the light receptors and a black layer behind these receptors. The black colour comes from the pigment melanin. Rods outnumber the cones and make us see in dim light (scotopic vision) whereas the cones are only sensitive to bright light. Cones are also responsible for

colour vision and the ability to distinguish fine details. Rods and cones are not equally distributed over the retina. In the centre, where vision for fine details and colour is best, only cones are found.



Distribution of rods and cones with respect to temporal and nasal eccentricity in degrees

The fovea is located 11.8° or 3.4 mm from the optical disc. The diameter of the foveola, a densely packed, rod free area is about 400 to 600 μm (cf: Polyak, 1941).

With respect to 50 x 50 μm the foveola contains about 147.000 cones/mm^2 (cf: Osterberg, 1935). The number of cones in the fovea is about 200.000 with 17,500 cones/$degree^2$. The rod's peak density is at 18° in a ring around the fovea with 160,000 rods/mm^2. The average number of rods is 80-100,000 rods/mm^2. The rod acuity peak is at about 5.3° with about 100.000 rods/mm^2 (cf: Mariani et al., 1984). The entire visual field corresponds to a 23400 square degree area. The diameter of the highest acuity circular region subtends 2° and the zone of high acuity extends to about 4° or 5°. Beyond 5°, acuity drops to 50%.

(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p.31-32)

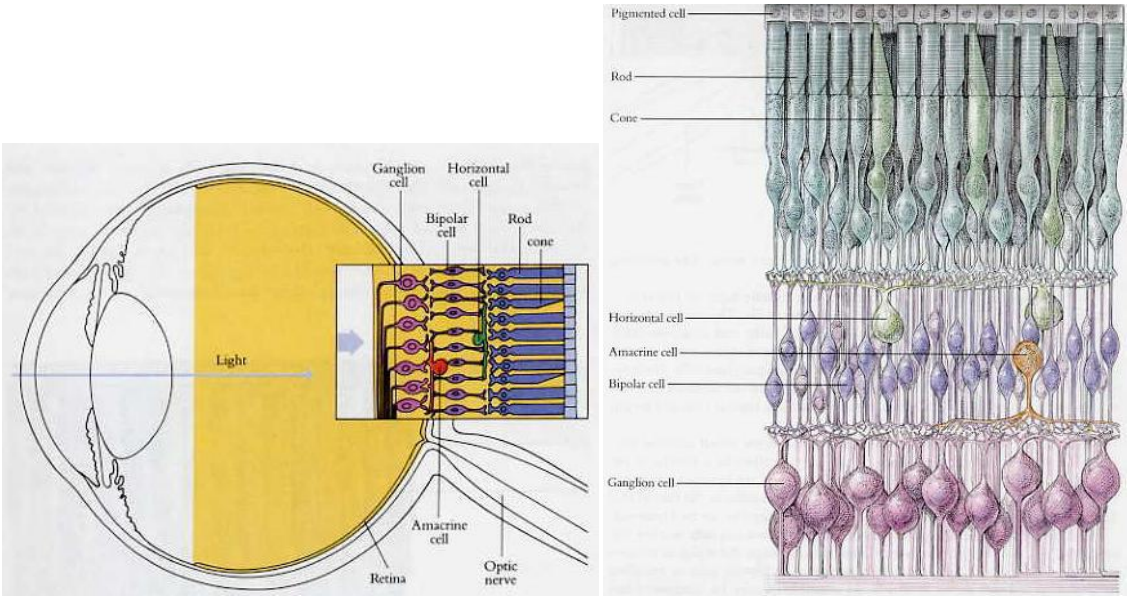
(cf: Hubel D.H. Eye, Brain, and Vision [Scientific American Library-Vol.22] (Freeman, 1995) p. 33)

(cf: Osterberg, G. (1935) Topography of the layer of rods and cones in the human retina. Acta Ophthal., suppl. **6**, 1-103)

(cf: Mariani, A.P., Kolb, H. and Nelson, R. (1984) Dopamine-containing amacrine cells of rhesus monkey parallel rods in spatial distribution. Brain Res. **322**, 1-7.)

(cf: Haber, R. N., & Hershenson, M. (1973). *The Psychology of Visual Perception.* New York: Holt, Rinehart, and Winston, Inc.)

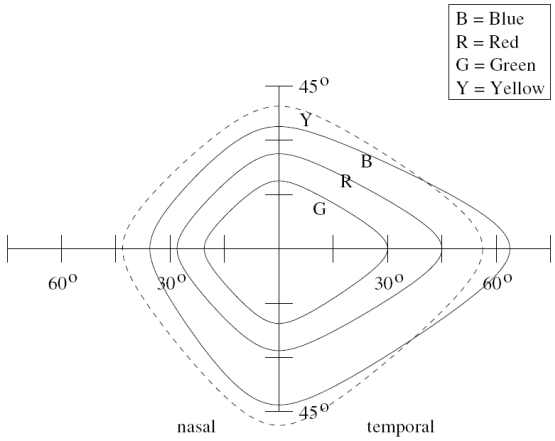
The melanin-packed cell layer is supposed to prevent light from being scattered around inside the eye and for chemically restoring light sensitive pigments inside the photo receptors.



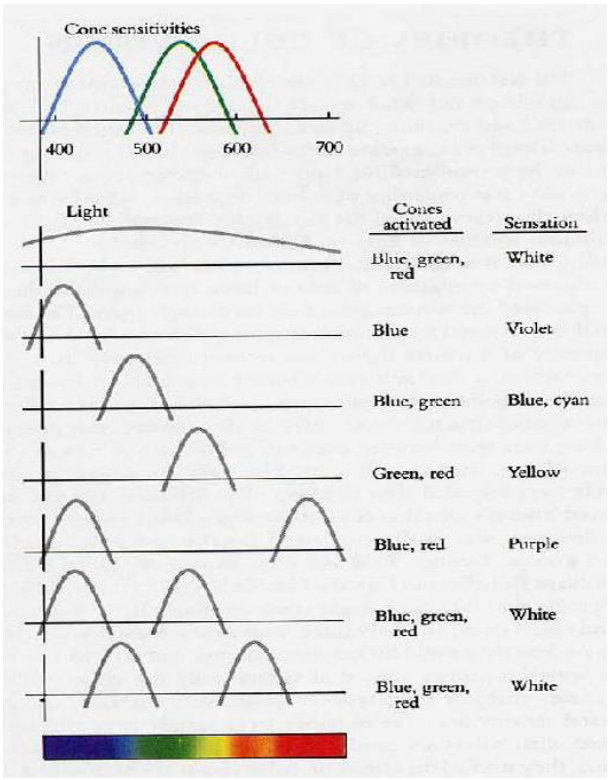
Cross section of the retina. The light travels thru several cell layers before hitting the rods and cones and the pigmented cells which absorbs light preventing it from scattering back.
(cf: Hubel D.H. Eye, Brain, and Vision [Scientific American Library-Vol.22] (Freeman,1995) chapter 3, p. 4,5)

The light has to travel through four cell layers that connect the rods and cones with the optic nerve. The light is somehow not distorted by these layers. Each eye has got about 125 million rods but only six million cones (cf: Osterberg, 1935). A rod cell is able to respond to even a single photon, making it about a hundred times more sensitive to light than a cone cell. Within the fovea each cone cell has its own exclusive ganglion cell and therefore a direct connection to the optic nerve. With greater distance to the fovea more and more receptors converge into interneurons. Rod cells respond times more slowly to light than cones. Stimuli are gathered for about 100 milliseconds making them considerably more sensitive to dim light than cones. As a consequence motion is best perceived within the fovea.

Color vision



Visual fields for monocular color vision
(right eye).
(cf: Boff, K. R., & Lincoln, J. E. (Eds.). (1988))



Cone sensitivities and their activation with respect to mixtures of coloured light.

Blue and yellow fields are larger than the red and green fields. Sensitivity drops off gradually and irregularly over a range of 15–30° visual angle (Boff & Lincoln, 1988). Peripheral color discrimination approximates foveal discrimination for relatively small field sizes e.g. 2° at 10° eccentricity, and less than 4° at 25°. Doyal (1991)

For colour discrimination at eccentricities of up to about 80° visual angle, the colour of a peripherally located 1.3° circle displayed on a CRT display is correctly identified 95% of the time if the colour is blue, 37% if the colour is red and 38% if the colour is green. Blue cannot be seen farther than 83.1°, red not farther than 76.3° and green not farther than 74.3° off the fovea along the x-axis. (cf: Ancman, C. E. G. 1991)

(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p.35-38)

(cf: Hubel D.H. Eye, Brain, and Vision [Scientific American Library-Vol.22] (Freeman, 1995) chapter 8, p.8)

(cf: Boff, K. R., & Lincoln, J. E. (Eds.). (1988). *Engineering Data Compendium: Human Perception and Performance.* Wright-Patterson AFB, OH: USAF Harry G. Armstrong Aerospace Medical Research Laboratory (AAMRL))

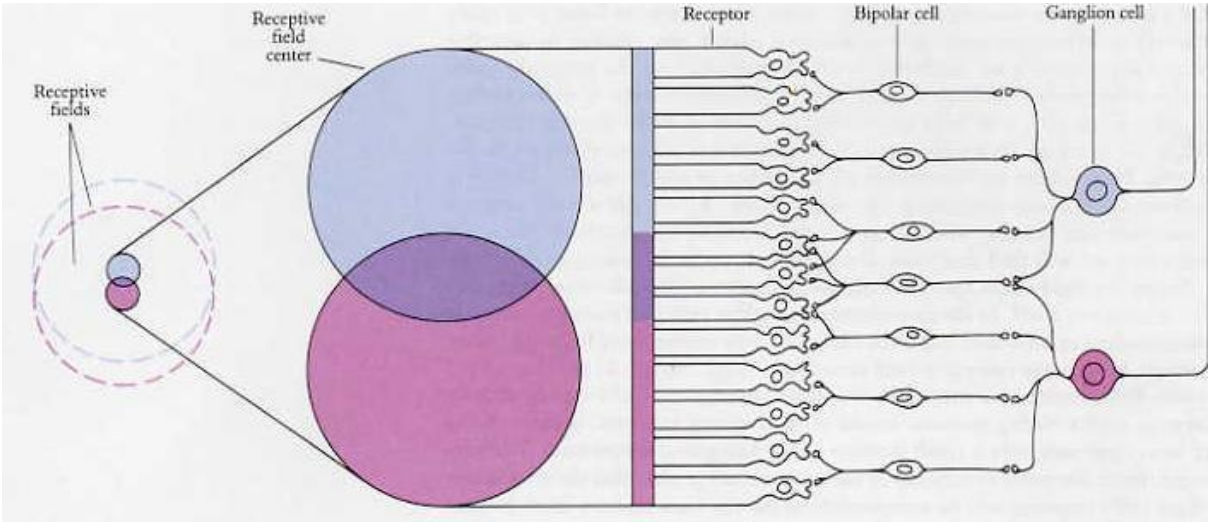
(cf: Doyal, J. A. (1991). *Spatial Summation for Color in the Peripheral Retina.* Unpublished master's thesis, Wright State University)

(cf: Ancman, C. E. G. (1991). Peripherally Located CRTs: Color Perception Limitations.
In *National Aerospace and Electronics Conference (NAECON)* (pp. 960–965).
Dayton, OH)

(cf: Osterberg, G. (1935) Topography of the layer of rods and cones in the human retina. Acta Ophthal., suppl. **6**, 1-103)

Receptive fields

The cluster in the back of the retina where receptors feed one ganglion cell in the front layer directly, or indirectly, is called the receptive field of the ganglion cell and is an area which can influence the ganglion cell's firing by light stimulation. There are two types of ganglion cells, the *on-centre cells* and the *off-centre cells*. An *on-centre cell* fires when the centre of its receptive field is exposed to light. It does not fire when the area around the centre, the surround, is exposed to light. *Off-centre cells* then show the opposite behaviour. Except for spontaneous firing, both cells do not fire when no stimulation occurs. If, however, the whole receptive field is stimulated by light, the cells only fire at a low frequency due to mutual inhibition of the centre and the surround.

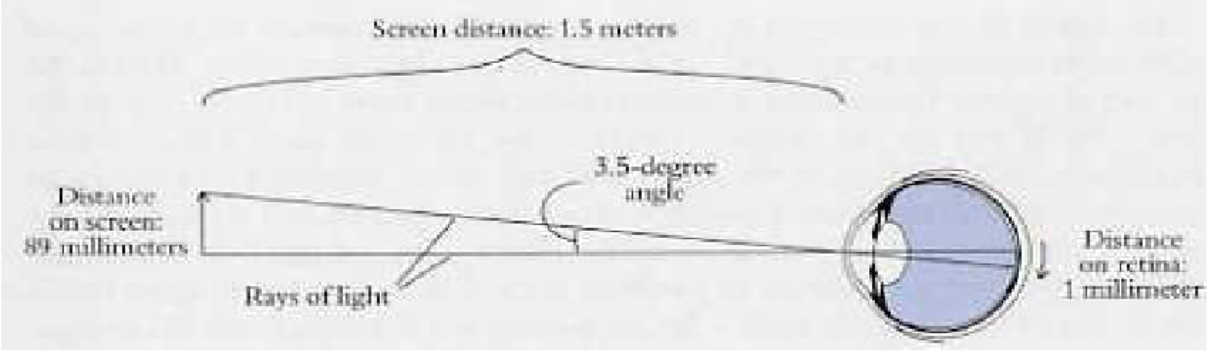


Receptive fields of neighbouring retinal ganglions overlap and therefore light is likely to influence hundreds of ganglion cells, some off and some on centre.

(cf: Hubel D.H. Eye, Brain, and Vision [Scientific American Library-Vol.22] (Freeman,1995) chapter 3, p. 10)

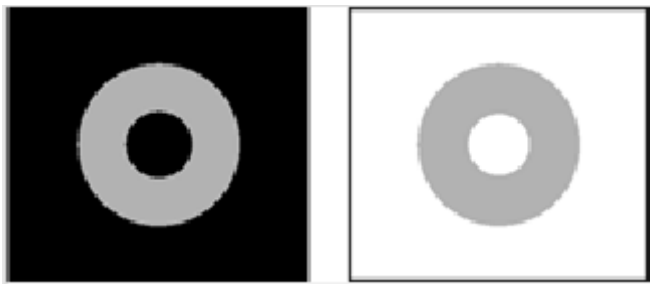
The receptive fields of neighbouring retinal ganglions overlap and therefore light is likely to influence hundreds of ganglion cells, some off and some on centre.

DIMENSIONS OF RECEPTIVE FIELDS



Relation between an objects' size and the affected retinal area. It describes the size of the receptive field as the angle subtended by the receptive field on a screen at the eye. For the human eye a distance of one millimeter corresponds to an angle of about 3.5°.
(cf: Hubel D.H. Eye, Brain, and Vision [Scientific American Library-Vol.22] (Freeman,1995) p. 41)

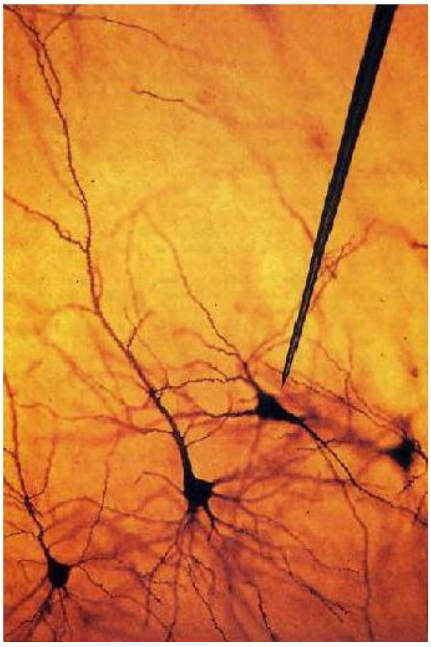
The image above shows the relation between an objects' size and the affected retinal area. It describes the size of the receptive field as the angle subtended by the receptive field on a screen towards the eye. For the human eye, a distance of one millimeter corresponds to an angle of about 3.5°. Taking in account the design of the retina mentioned earlier, it is obvious that receptive fields differ in size. Within the fovea, the centres of the receptive fields are smallest and as a result the visual acuity, the ability to distinguish fine details, is best. Farther to the periphery of the retina, the centres become bigger while the acuity deteriorates progressively. In the periphery of the retina, field centres consist of thousands of receptors with diameters of one degree or more. This implies that the receptive fields' centre is determined by the direct pathway (from receptors to bipolars to ganglion cells) and the receptive fields' surround by indirect pathways (involving interneurons).



Ganglions respond best to differential illumination of the receptive field centre and the surround. Hence, an object's appearance depends on the contrast between the object and its background
(cf: *Principles of neural science* (Fourth edition, P. 517). Connecticut: Appleton & Lange)

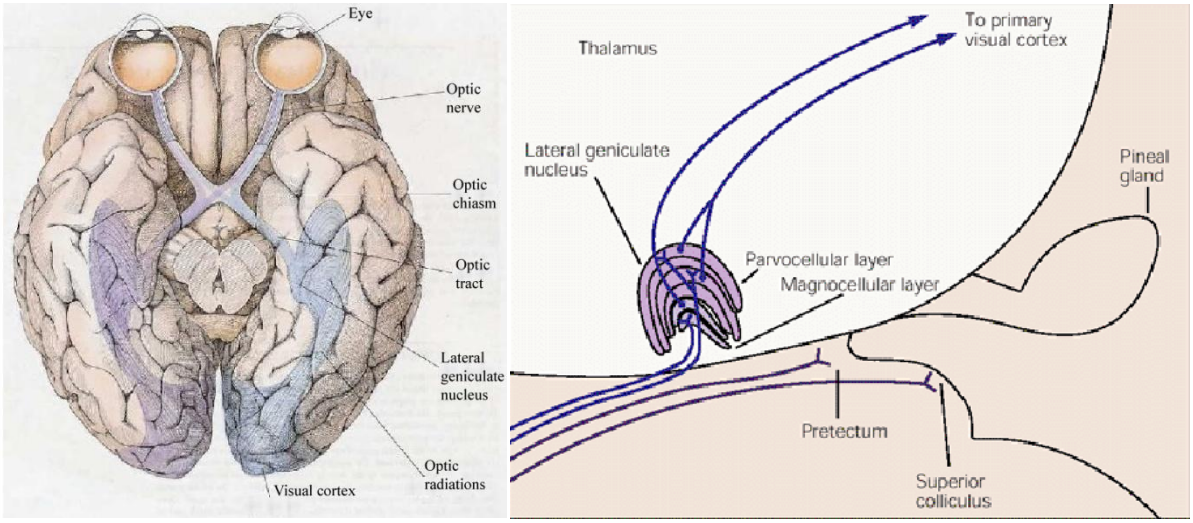
Ganglions respond best to differential illumination of the receptive field centre and the surround. Hence, an object's appearance depends on the contrast between the object and its background (see image above). We usually perceive an object by its shape and its colour. The shape is determined by the edges which are only visible if the contrast in brightness and/or colour between the object and its background is high enough and it is the specialized ganglion cells with their centre surround receptive fields that make this possible. Thus we are able to detect meaningful information in a visual scene. However, our perception of subtle changes in illumination is not good, because the ganglion cells' response to uniform illumination is rather weak. The detection of contrast starts in the retina, because transmitting the information from the photoreceptors through several relays to the visual cortex distorts the signal A way to mitigate the transmission errors is to let the retinal ganglions do the measuring of the differences. The firing rate of the ganglion cell corresponds to the intensity of light illuminating centre and surround. The intensity differences are then transmitted to the higher cortical centres. The visual system's performance is further improved by parallel on-centre and off-centre pathways, because each type of the ganglion cell is responding best to rapid decreases or increases in illumination. The retina has even more to offer. For different aspects of the visual image, there are different types of specialized cells. M (*magnocellular*) cells have large receptive fields and respond particularly well to large objects. M cells are able to follow rapid changes in the stimulus. M cells therefore appear to be concerned with the analysis of the gross features of a stimulus and its movements. So called P (*parvocellular*) cells then have smaller receptive fields and appear to be concerned with the analysis of fine details.

The Visual Cortex and the Central Visual Pathways



The visual cortex in a monkey, stained by the Golgi method, shows a few pyramidal cells—a tiny fraction of the total number in such a section. The entire height of the photograph represents about 1 millimeter. A tungsten microelectrode, typical of what is used for extra cellular recordings, has been superimposed, to the same

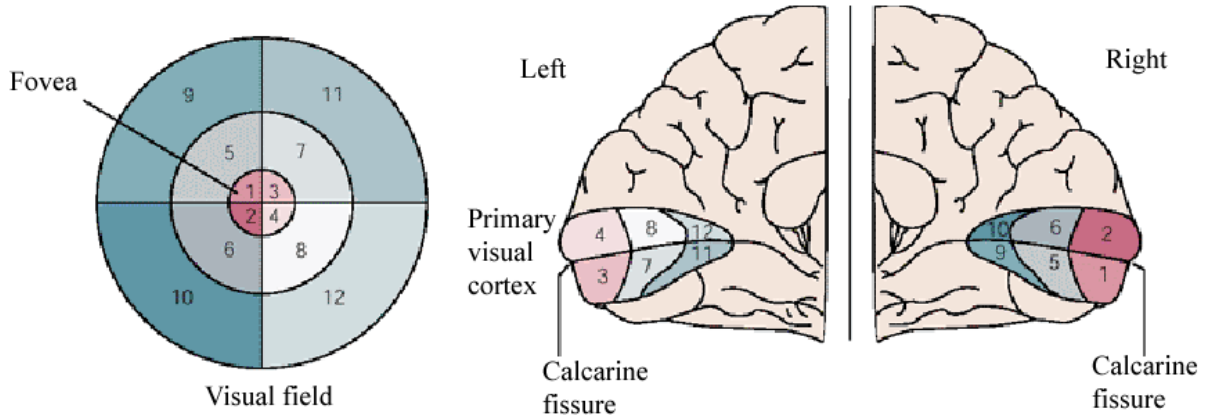
(Image and description cf: Hubel D.H. Eye, Brain, and Vision [Scientific American Library-Vol.22] (Freeman,1995) p. 53)



Visual information arrives to the two purple-coloured halves of the retinas. Half of the left visual field ends up in the right half of the brain and vice versa, because about half the nerve fibres cross at the chiasm. The other half stays uncrossed. Each hemisphere gets input from both eyes and each hemisphere gets information from the opposite half. The information is then projected to three subcortical targets, the pretectum, the superior colliculus and the lateral geniculate nucleus, a structure of alternating gray cellular and white axonal layers on the roof of the midbrain. Within the superficial layers cells project through the pulvinar nucleus of the thalamus to an area of the cerebral cortex to establish an indirect pathway from the retina to the cerebral cortex.

(cf: Hubel D.H. Eye, Brain, and Vision [Scientific American Library-Vol.22] (Freeman,1995) chapter 4, p. 2)
(cf: *Principles of neural science* (Fourth edition, P. 527). Connecticut: Appleton & Lange)

Visual information arrives to the two purple-coloured halves of the retinas. Half of the left visual field ends up in the right half of the brain and vice versa, because about half the nerve fibres cross at the chiasm. The other half stays uncrossed. Each hemisphere gets input from both eyes and each hemisphere gets information from the opposite half. The information is then projected to three subcortical targets, the pretectum, the superior colliculus and the lateral geniculate nucleus, a structure of alternating grey cellular and white axonal layers on the roof of the midbrain. Retinal ganglions project directly to the superficial layers to form a map of the contralateral visual field. Within the superficial layers, cells then project through the pulvinar nucleus of the thalamus to an area of the cerebral cortex to establish an indirect pathway from the retina to the cerebral cortex. Deeper layers have the same map of the visual field found in the superficial layers and receive projections other areas of the cerebral cortex like e.g. the auditory cortex. Thus, neurons that respond to a car moving within the contralateral visual field also will respond to its motor noise when it is in that same part of the field. The auditory and somatosensory inputs are adjusted to fit with the visual map in situations where the maps of these other modalities might diverge, e.g. when the eyes are directed to one side while the head is directed straight ahead with respect to the body. A car located where we are looking will be in the centre of the visual field but its engine noise will locate it to one side of the auditory field. Cells within the deeper layers of the colliculus also discharge before the onset of saccadic eye movements, forming a movement map (in register with the visual map) in the intermediate layers of the colliculus. The superior colliculus receives direct retinal input. However, control of saccadic eye movements is assumed to be determined by inputs from the cerebral cortex that reach the intermediate layers.



Relation between the visual field and the visual cortex.

(cf: *Principles of neural science* (Fourth edition, P. 531). Connecticut: Appleton & Lange)

The Lateral Geniculate Nucleus

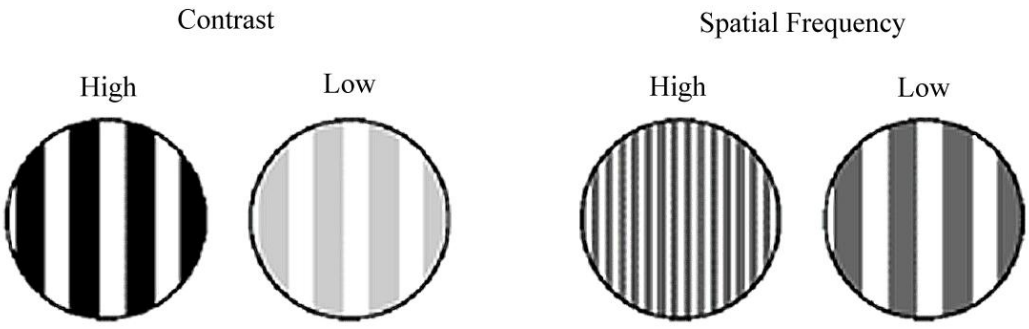
Cells responding to stimuli in the right visual field will discharge before a rightward saccade and vice versa. Ninety percent of the retinal axons terminate in the lateral geniculate nucleus, making it the main terminus for input to the visual cortex. Losing this pathway results in an almost complete loss of visual perception, although movement toward objects in the visual field and very limited stimulus detection still remains. This residual vision is called blindsight. As mentioned earlier, there are considerable differences between cells in the M and P pathways. The most important difference between the cells in the lateral geniculate nucleus is their sensitivity to *colour contrast.* While the P cells respond to red/green and blue/yellow changes regardless of the relative brightness of the colours, M cells do not respond much to changes of colour when the brightness of the colour is matched.

Stimulus feature	M cells	P cells
Colour contrast	No	Yes
Luminance contrast	Higher	Lower
Spatial frequency	Lower	Higher
Temporal frequency	Higher	Lower

(cf: *Principles of neural science* (Fourth edition, P. 530). Connecticut: Appleton & Lange)

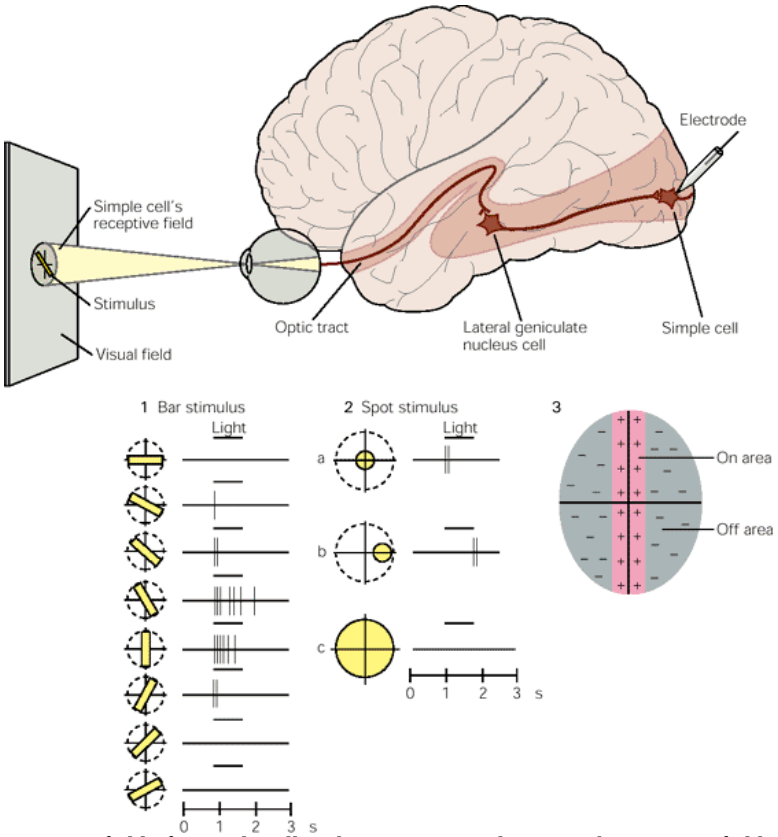
Luminance contrast is the difference between the brightest and the darkest parts of the stimulus. M-cells respond to contrasts as low as only 2%. P cells only respond to contrasts as low as 10%. Spatial frequency is the number of times a pattern is repeated per degree of visual angle, whereas the temporal frequency is how rapidly a pattern changes over time.

Grating stimuli



(cf: *Principles of neural science* (Fourth edition, P. 530). Connecticut: Appleton & Lange)

In the visual cortex, the retinal input is organized into building blocks of visual images. Specialized cells within the visual cortex decompose the image into short line segments. Individual cells respond best to lines or bars of light with a specific orientation. Hence, a cell responding best to a vertical bar will therefore respond weakly to a horizontal bar.

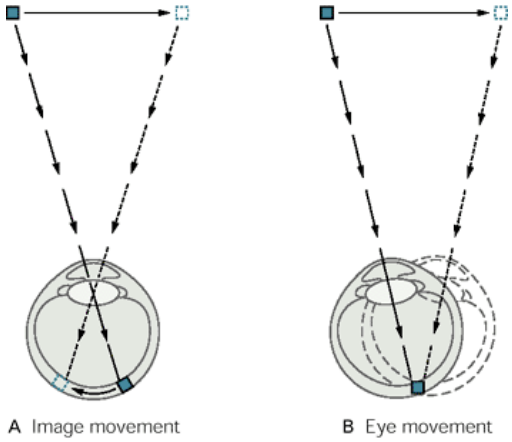


Receptive field of a simple cell in the primary visual cortex. The receptive field of a cell in the visual system is determined by recording activity in the cell while spots and bars of light are projected onto the visual field at an appropriate distance from the fovea. The records shown here are for a single cell. Duration of illumination is indicated by a line above each record of action potentials. (Adapted from Hubel and Wiesel 1959 and Zeki 1993.)
1. The cell's response to a bar of light is strongest if the bar of light is vertically oriented in the centre of its receptive field.
2. Spots of light consistently elicit weak responses or no response. A small spot in the excitatory centre of the field elicits only a weak excitatory response. A small spot in the inhibitory area elicits a weak inhibitory response. Diffuse light produces no response.
3. By using spots of light, the excitatory or "on" areas (+) and inhibitory or "off" areas (-) can be mapped. The map of the responses reveals an elongated "on" area and a surrounding "off" area, consistent with the optimal response of the cell to a vertical bar of light.
(cf: *Principles of neural science* (Fourth edition, P. 533). Connecticut: Appleton & Lange)

Consequently, the interior of a form does not excite cells in the brain. Instead, we become aware of an object because cells in our brain are sensitive to borders.

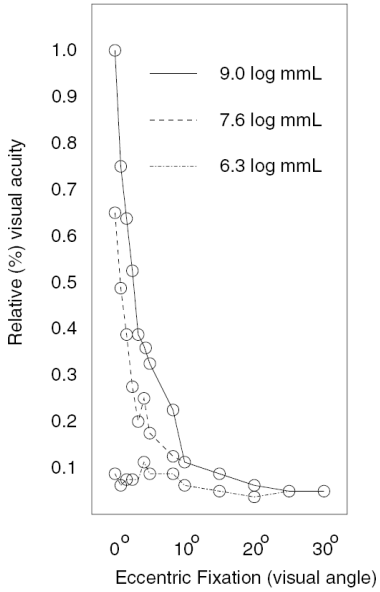
The area in the brain involved in processing information about form and colour is the infra-temporal-cortex.

Motion



(cf: *Principles of neural science* (Fourth edition, P. 551). Connecticut: Appleton & Lange)

Motion is analyzed primarily in the dorsal pathway to the parietal cortex. To perceive motion, there are basically two ways to do it. A: An object in motion produces a moving image on our retina resulting in sequential firing of receptors of the retina. B: An object in motion tracked by moving the eyes and or head produces an image that remains (more or less) stationary on the retina. In this case the sensation of motion comes primarily from moving the eyes and or the head.



Visual acuity at various eccentricities and light levels.

(cf: Davson, H. (1980). *Physiology of the Eye* (4th ed.). New York: Academic Press.)

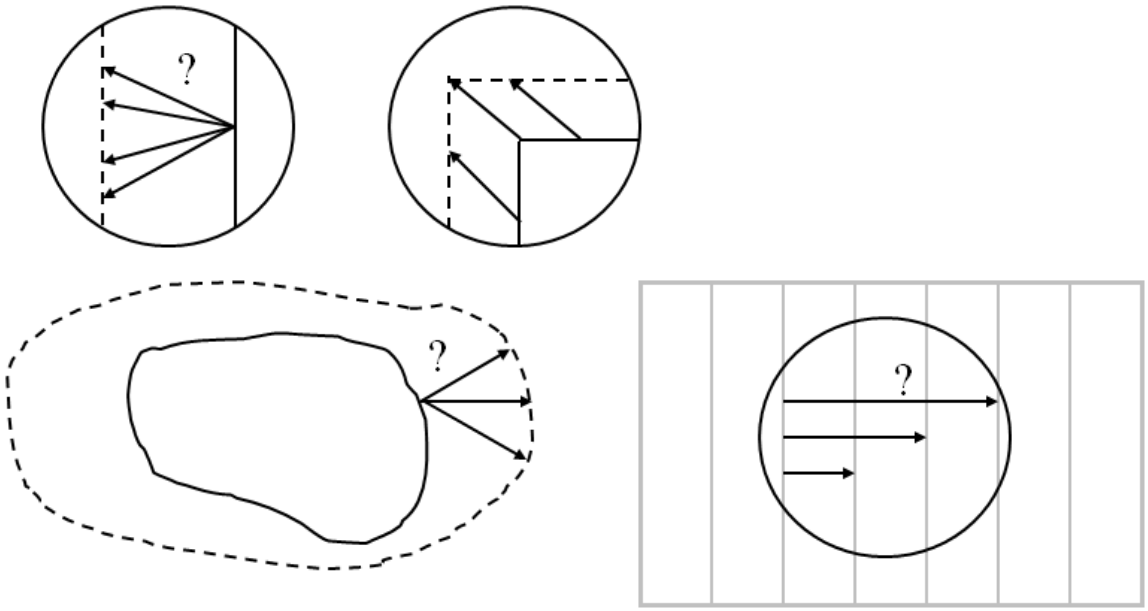
(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p.34)

Motion within the visual field is detected by comparing two successive images. In a strict sense, the eye is not a camera but sends *continuous* visual information to the visual cortex. Most cells in the visual system are sensitive to retinal position and are able to resolve changes at up to 100 Hz. Motion that is independent of retinal position is called *apparent motion* and

is the reason why things like TV, motion pictures and even “moving” neon signs work for us. It also suggests that there are specialized neural subsystems for each type of motion.

While analyzed in the parietal cortex, representation takes place in the Middle Temporal Area (MT). This is the place where the *Aperture Problem* is solved. It is also the place where *direction sensitive neurons* determine the direction of lines, entire objects in the visual field and patterns (*pattern-direction-sensitive neurons*).

Aperture Problem



The grating is larger than the aperture. Therefore it is impossible to unambiguously determine the displacement vector, which describes the motion of the grating. E.g. it is impossible to detect motion in the vertical direction.

(cf: Jähne 1997 Jähne, Bernd: Digitale Bildverarbeitung: Springer – ISBN 3-540-61379-X)

Motion in the visual periphery

With respect to visual attention, the temporal response of the human visual system is not homogeneous across the field of vision. With respect to motion responsiveness, the periphery is more receptive to faster motion than the fovea. Sensitivity to target motion decreases monotonically with retinal eccentricity for slow and very slow motion with 1 cycle/deg, meaning that the velocity of moving targets appear to be slower in the periphery than in the fovea. Therefore, a higher rate of motion is needed in the periphery to match apparent stimulus velocity in the fovea. Because movement is more salient in the periphery than it is in the fovea, it is easier to peripherally detect a moving target than a stationary one. This and the low sensitivity for fine details make early motion detection the periphery's major task.

(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p.35, 36)

(cf: Boff, K. R., & Lincoln, J. E. (Eds.). (1988). *Engineering Data Compendium: Human Perception and Performance.* Wright-Patterson AFB, OH: USAF Harry G. Armstrong Aerospace Medical Research Laboratory (AAMRL))

Donders and Listing's Law

For any gaze direction, the eyes *could* rotate around the gaze axis (torsional movement). In fact they don't. For each gaze direction, the eye has got a unique position. No matter what movements had to be made to achieve the gaze. This is known as Donders' Law. When including a specification of that particular position adopted, we get Listing's Law. A consequence of Listing's Law is that eye movements can be specified in terms of the horizontal and vertical components of rotations. Other degrees of freedom can therefore safely be ignored. Apart from voluntary movements, one major task of the extraocular muscles is achieving visual stabilization during head and body movements.

Vestibulo-ocular and Optokinetic reflexes

The two systems involved are the *vestibulo-ocular* and *optokinetic* reflexes (VOR and OKR) and they are completely involuntary and automatic. The stabilization information for the *vestibulo-ocular* reflex is derived from the vestibular organs of the inner ear; the *optokinetic* reflexes derive their information from patterns of coherent optic flow on the retina.

Saccadic, Pursuit and Vergence Systems

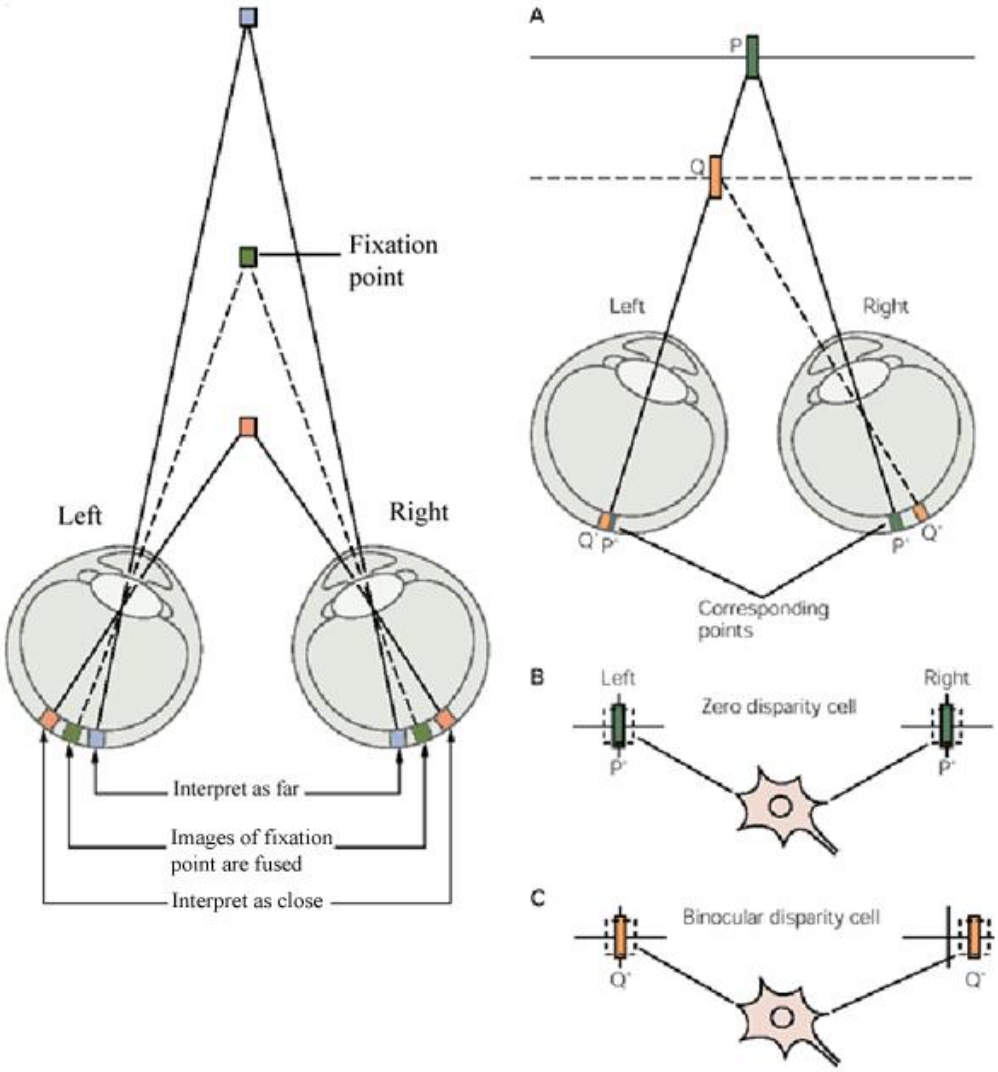
The systems involved in target selection and bringing or keeping a target onto the *fovea*, the area where vision is best, are the *saccadic*, *pursuit* and *vergence* systems. The saccadic system is rotating the eyes to bring a selected target onto the fovea. The pursuit system makes it possible to smoothly follow an object in motion. The vergence system maintains both eyes on

a target in motion and makes adjustments with respect to its distance to the eye. This is essential for proper depth perception. While the saccadic movements are jump-like movements and rotations for brief periods at several hundred degrees per second, pursuit movements are continued movements at speeds well below 100 degrees per second. Both systems are conjugate with both eyes rotating equally. Vergence movements are continuous movements in which the eyes move in opposite directions at a speed below 20 degrees per second. Pursuit and vergence system together maintain the eyes aligned onto an object in motion and make fast target acquisition possible.

Depth perception

Psychophysical studies suggest that three-dimensional perception relies on monocular clues for depth and stereoscopic cues for binocular disparity. At fairly great distances of about 30 meters, where both retinal images are about the same, we essentially have monocular vision. To determine distance we rely on monocular cues like *familiar size*, *occlusion*, *linear perspective parallel lines*, *size perspective* (size difference between similar familiar objects), *distribution of shadows* (brighter shades of colours appear as nearer) and most importantly, *parallax* (monocular motion).

The predominant cue for depth perception at distances of less than 30 meters is stereoscopic vision. This is because the two eyes are horizontally separated by about 6 cm (in humans). Fixating on a point causes identical images on each retina. Depth cues are given by points proximal or distal to the fixation point. Binocular disparity is created because these points stimulate slightly different areas of each eye's retina. The information from both eyes are processed in the primary visual cortex to accomplish stereopsis. For that the inputs from both eyes have to be different. There must be enough horizontal disparity in the two retinal images, otherwise points are fused into a single three dimensional spot. Cells which are actually selective for horizontal disparity were found in 1968 by Horace Barlow, Colin Blakemore, Peter Bishop, and Jack Pettigrew.



(cf: *Principles of neural science* (Fourth edition, P. 560,561). Connecticut: Appleton & Lange)

In example A we have zero binocular disparity for point *P* since the retinal images overlap perfectly. For point *Q* we get some lateral displacement mostly in the right eye and therefore binocular disparity. In image B we see that the *zero disparity cell* is maximally activated because the inputs of both eyes have zero disparity for retinal image P'. In image C the *binocular disparity cell* is maximally activated because the inputs of both eyes are spatially disparate with respect to cortical image Q'. As for motion, the area involved in depth perception is also the MT.

Visual Attention

Everyone knows what attention is. It is the taking possession by the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought. Focalization, concentration, of consciousness are of its essence. It implies withdrawal from some things in order to deal effectively with others...
(cf: Andrew T. Duchowski Eye Tracking Methodology:Theory & Practice, p.4)

When the things are apprehended by the *senses*, the number of them that can be attended to at once is small, *'Pluribus intentus, minor est ad singula sensus.'*
(cf: William James (1890) The Principles of Psychology p.257)

The Latin term translates into *"Many filtered into few for perception"* and implies that human beings are not capable of attending to many things at the same time. The objects we pay attention to receive more processing and more representation in perception. Attention is also linked to some kind of interaction or at least the intention to do so. As a result, visual attention is about selection of objects in the visual field and the link between selection and (inter)-action.
(cf: Allport, 1993, Active Vision The Psychology of Looking and Seeing - John M. Findlay p.41).

This important problem of *selective attention* is then further carried out in the essay "The Stream of Thought"

"We see that the mind is at every stage a theatre of simultaneous possibilities. Consciousness consists in the comparison of these with each other, the selection of some, and the suppression of the rest by the reinforcing and inhibiting agency of attention. The highest and most elaborated mental products are filtered from the data chosen by the faculty next beneath, out of the mass offered by the faculty below that, which mass in turn was sifted from a still larger amount of yet simpler material, and so on. The mind, in short, works on the data it receives very much as a sculptor works on his block of stone."
(cf: William James (1890) The Principles of Psychology p.174)

Attention, in one sense, is seen as a "selective filter" responsible for regulating sensory information to sensory channels of limited capacity.
In 1958 Donald E. Broadbent concluded that information enters in parallel but is then selectively filtered to sensory channels.
(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p.6).

Deutsch and Deutsch (1963) rejected this idea and the assumption of a limited capacity system theory of attention, since a filter would have to be at least as complex as the limited capacity system itself. Instead they argued that it is not attention but weightings of importance that have causal role in attention.

Covert and overt attention

With some effort we are able to consciously fixate on an object and to some part of the periphery of vision at the same time without moving our eyes. This observation was made by Helmholtz in 1866 and is strongly supported by the design of the retina and the distribution of rods and cones. This ability is called *covert attention*.

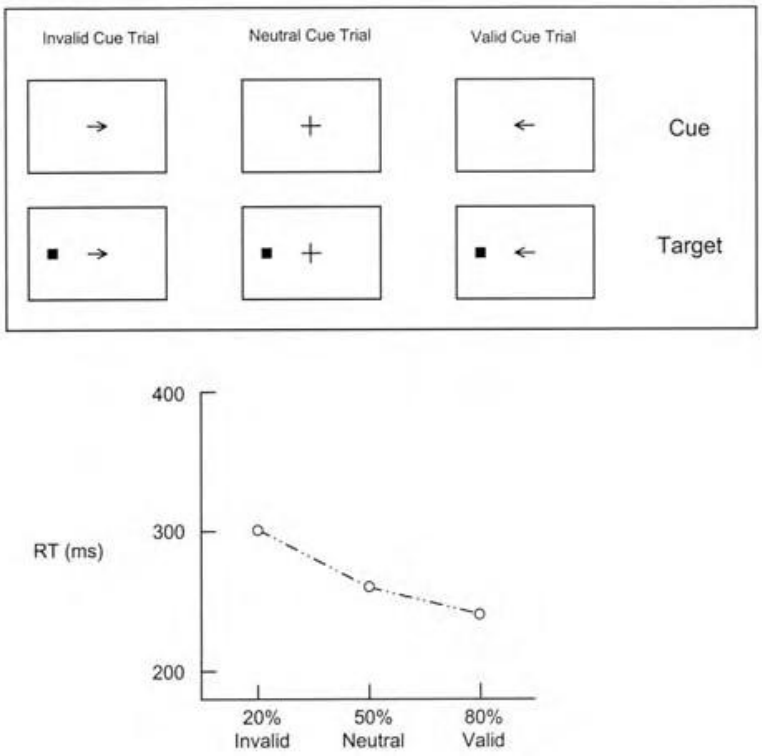
Moving the eyes to saccade and foveate parts of the visual field requires a lot less effort and is called *overt attention*.

Covert spatial attention

Michael Posner (1978, 1980; Posner *et al.*1978; Posner *et al.* 1980) showed that reaction times are faster for previously cued locations. In these experiments, subjects were asked to maintain fixation on a central fixation point and to not move their eyes. After some time, they were shown an arrow cue at the fixation point, indicating the possible location of a subsequent target. Most of the time, the target appeared at the correct position: so called *valid trials*. Sometimes, the target appeared in the opposite direction to the cue: so called *invalid trials*. Additionally, there were so called *neutral trials* when no target appeared at all, because no target location was given. Comparing the reaction times for the three conditions gives the costs and benefits of the spatial cues. Compared to the neutral condition the valid condition led to faster and the invalid condition to slower reaction times. Thus, even when the eyes do not move, correct information about the possible location of a target leads to better reaction times and incorrect information prolongs reaction time. This is observation is called *The Posner cueing Paradigm*. The paradigm remains valid even when the cue consists of a peripheral event such as a flash.

A peripheral cue gives the largest advantage when it occurs 100 ms before the target is presented. The central cue gives a maximum benefit about 300 ms before the target. If the cue is uninformative and indicates the target position on only 50% of the trials, the costs and benefits are still present for peripheral cue, but not for the central cue. As a result, peripheral cueing is called *automatic* and central cueing *voluntary*.

For much longer intervals between the cue and the target, a cost is associated with the cue indicating the target location (Posner and Cohen, 1984). The effect is called *inhibition of return* (IOR) and is supposed to reflect a bias against attention returning to the cued location. IOR might also be mechanism to structure scanpaths and to prevent rechecking in visual search.



Posner cueing paradigm. The test subject is shown a cue and then a target. Reaction time varies on the cue. Reaction time is faster for previously cued locations.

(cf: The Posner cueing paradigm, Active Vision The Psychology of Looking and Seeing - John M. Findlay p.42).

Visual Attention: Different Opinions

In over 150 years of research, different people have explored the topic from different perspectives; not surprisingly, there are quite different opinions to consider.

In Helmholtz's *Treatise on Physiological Optics* he notes "We let our eyes roam continually over the visual field, because that is the only way we can see as distinctly as possible all the individual parts of the field in turn."

(cf: Third Volume, 1925, Electronic edition (2001): p.63)

He therefore observed visual attention's limitation to small regions of space and the mind's tendency to wander to new things. This type of attention is referred to as covert attention.

Helmholtz's main concern, however, was *overt attention* or the "where". Here, conscious eye movements to spatial locations reflect the subjects' will to inspect and / or to interact with an object.

William James' idea about how visual attention works was quite different to Helmholtz's idea. James's focus was on *covert attention*, an internal mechanism akin in general to thought. Attention in terms of the "what" therefore then relates to *what* an objects means to us or *what* expect we from looking at it.

Anatomically, the concepts of "what" and "where" correspond to the infra-temporal-cortex (IT) and the mid-temporal-cortex (MT) whereas *foveal* (James) and *parafoveal* aspects (Helmholtz) roughly represent the psychological view on visual attention.

Given an image with various areas of interest, an area in the periphery might then draw the attention to further inspect this area foveally. Before we can identify "what" we see, we first have to decide "where" to look next.

The dichotomy between both points of view provides a meaningful background to a dual "where" and "what" bottom-up feature driven explanation of visual attention and allows the creation of simple computational models of visual attention. However, better and more complete models have to include intentional and pre-conceived cognitive aspects that drive attention as well.

In the 1940s, James Jerome Gibson proposed another aspect of visual attention. For Gibson, it was important "how" a subject reacted to a stimulus, because "what to do" or "how" to do something largely depended on the subjects' preconceptions or attitude. Therefore, in an experiment, a subject's perceptual expectation of a stimulus could be influenced by the instructions given at the beginning.

Deutsch and Deutsch's idea of central structures with preset importance weightings and changes in attention as a result of importance corresponds pretty well to William James' "what", whereas Broadbent's "where" corresponds to Helmholtz's "where".

Eventually in the 1960s, Anne Treisman resolved this dichotomy by incorporating both opposing ideas into her unified theory of attention. To bring the theories of Deutsch and Deutsch and Broadbent together, Treisman introduced two components of attention: an attenuation filter and central structures called "dictionary units". Her attenuation filter is a modified and extended version of Broadbent's version; instead of completely blocking unwanted messages, it attenuates them to a certain degree before all messages are processed by "dictionary units".

(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p.7).

The relationship between covert and overt attention

Naturally, our eyes are not stationary and fixed to a single point all the time. Thus, both overt and covert attention can co-occur, because they are driven by similar visual input. Although independent, they are closely coupled, with saccades directed by the location of covert attention.

(cf: Active Vision The Psychology of Looking and Seeing - John M. Findlay p.46)
(cf: Klein's Independence Account, Active Vision The Psychology of Looking and Seeing - John M. Findlay p.47)
(cf: Klein, R. (1980). Does oculomotor readiness mediate cognitive control of visual attention?
In Attention and Performance VIII, (ed. R. S. Nickerson) pp. 259-276, Lawrence Erlbaum Associates, Hillsdale N J.)

Sequential Attention Model

Eyetracking recordings performed by Yarbus (1967) and Noton and Stark ("scanpaths") in 1971 suggest that a coherent picture in the visual field is not perceived at once but assembled from small serially viewed regions of interest. James' "what" then corresponds to regions of interest selectively filtered by foveal vision for detailed inspection.

(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p.8, 9).

In an experiment, Henderson et al. (1989) examined the degree to which participants are able to use parafoveal preview of objects while foveally viewing displays with four objects on the corners of an imaginary square around the central fixation location. Participants had to perform saccades to the objects in a specified order to determine the object's identity. It was found that fixation on a specific object was shorter when it had been visible during the previous fixation in comparison to when it was masked during the previous fixation. The preview of objects that would be fixated two saccades later did not further reduce fixation duration on that object. Henderson et al. argued that during fixation a reduction in fixation time is caused by shifting the attention to the object that would be fixated next, allowing pre-processing of the object at that location. The indication that attention was only allocated to the next saccade location is considered evidence for the sequential attention model. Saccades and attention are both directed to the same location to facilitate the processing of objects. Therefore, in reading and in fact every time the eyes are moving from one object to another object of interest, allocating the next saccade to that next object makes sense.

(cf: Parallel Allocation of Attention Prior to the Execution of Saccade Sequences, Journal of Experimental Psychology Human Perception and Performance 2003, Vol. 29, No. 5, p.883)

Later in 1992 Henderson developed a *sequential attention* model to investigate a close link between covert and overt attention. His assumptions are:

1. Attention is allocated to the stimuli at the centre of fixation at the beginning of each fixation.

2. Attention is allocated to a new stimulus when the fixated stimulus is understood or identified.

3. Reallocation of attention is coincident with the beginning of saccade programming to the new location that becomes the target for the next saccade.

4. Allocation of attention to the new location involves higher level processing at the new location.

(cf: Active Vision The Psychology of Looking and Seeing - John M. Findlay p.47)

Parallel Attention Model

While sequential models assume that attention is only allocated to the next single saccade goal, Parallel attention models assume that attention can be allocated to multiple saccade goals or locations.

(cf: Parallel Allocation of Attention Prior to the Execution of Saccade Sequences, Journal of Experimental Psychology Human Perception and Performance 2003, Vol. 29, No. 5, p.883)

Feature Integration Theory

*"A purple giraffe with wings would look surprising but it would not be invisible"(*Treisman, 1999, p. 92). And so *can* we see purple pigs with wings…

*The binding problem in perception deals with the question of how we achieve the experience of a coherent world of integrated objects, and avoid seeing a world of disembodied or wrongly combined shapes, colours, motions, sizes and distances (*Anne Treisman (1998).

According to Treisman and Gelade (1980), basic visual features across the visual scene are extracted in parallel if search targets are defined by unique basic features (e.g. colour and orientation) relative to the background. This search is therefore often referred as *feature search*, whereas search targets (*unitary objects*) defined by a conjunction of features lead to serial search. This search is therefore often referred as *conjunction search*. Treisman and Gelade (1980) argued that complex features can only be detected because attention provides the glue which integrates separated features in a particular location, making perception of objects as a unified whole e.g. the purple pig with wings possible. Conjunction search assumes an attentional spotlight moving serially across the complex object for detection. This serial scanning process is assumed to consume some additional time in serial search.

Treisman's *Feature Integration Theory* links *preattentive* and *attentive* processing. The idea is that *spatially selective attention* (cf: Posner et al.) might play an important role in binding (Treisman, A. (1999) the "what?" and the "where". Treisman's *Feature Integration Theory* therefore adds to earlier studies which show that visual processing is modularized to some degree and provides one explanation of how features are bound.

(cf: Active Vision The Psychology of Looking and Seeing - John M. Findlay p.107)

(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p.11)

(cf: André Melzer (2002). Von roten Autos und blauenTöpfen: Farbinformationen in impliziten und expliziten Gedächtnistests, Universität Trier, p.82)

(cf: Treisman, A. (1999). Feature binding, attention and object perception. In G. W. Humphreys & J. Duncan & A. Treisman (Eds.), *Attention, Space and Action. Studies in Cognitive Neuroscience* (pp. 91-111). Oxford: University Press.)

(cf: Treisman, A., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology, 12*, 97-136.)

(cf: Anne Treisman (1998) Brain Mechanisms of Selective Perception and Action Feature Binding, Attention and Object Perception, *Philosophical Transactions: Biological Sciences*, Vol. 353, No. 1373, pp. 1295-1306)

(cf: Principles of neural science (Fourth edition, p. 493). Connecticut: Appleton & Lange)

Visual Attention and Eye Movements

In order to qualify for an adequate model of attention, eye movements must be taken into consideration. Klein and Farrell (1989) e.g. found that when search was serial, restricting eye movements (preventing saccades) lead to a relatively small but significant error increase from 9.3% to 18.8%. This effect is even present for small display sizes and gets even more prominent with an increase in the number of items and bigger display sizes.

Taking eye movements into account, in terms of a dual "where" and "what" bottom-up feature driven model of attention (cf: William James), we have a cyclical process composed of at least the following steps:

1. An entire scene is first seen mostly in parallel through peripheral vision at low resolution. Eventually, interesting features may "pop out" and draw attention to their location for further detailed inspection.
2. Attention is disengaged from the foveal location and the eyes move to the first attractive region.
3. Attention is reengaged and the region of interest is inspected foveally at high resolution.

This bottom up model is at least compatible with natural human vision. However, it does not address the types of features (cf: chapter "The Lateral Geniculate Nucleus"), nor does it explain the need for voluntary eye movements (cf: Yarbus and Gibson). It does also not cover the link between attention and eye movements, a classical question in eye tracking technology, because attention addresses both low and high level cognitive functions – "voluntary and involuntary attention" (cf: Posner et al.).

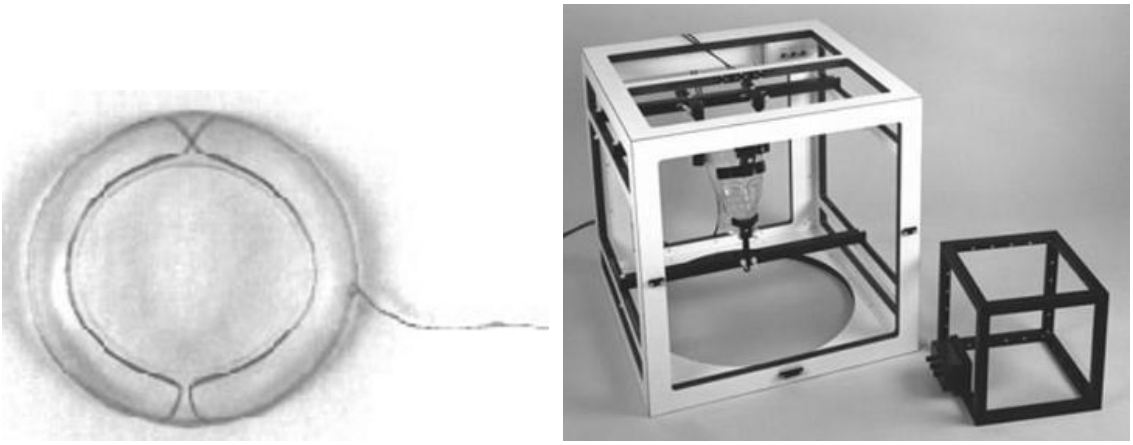
(cf: Active Vision The Psychology of Looking and Seeing - John M. Findlay p. 11-12, 110-112)

(cf: Klein, R. M. and Farrell, M. (1989). Search performance without eye-movements.)

s(cf: Perception and Psychophysics, **46**, 476-482.)

Eye Tracking Techniques

Scleral Contact Lens/Search Coil

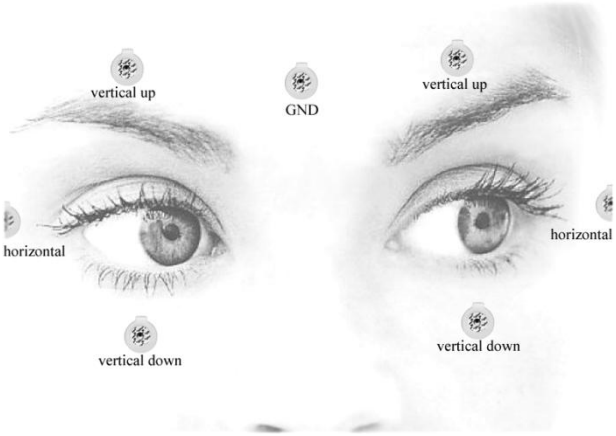


Search coil embedded in contact lens electromagnetic field frames for search coil eye movement measurement (cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p.53)

This very precise eye tracking method requires the subject to wear contact lenses with an embedded coil and therefore creates considerable discomfort for the subject. The way it works is that there are alternating magnetic fields which are different for each coordinate axis. The position of the eyes can be determined by measuring the changes in voltage induced in the search coils as the scleral contact lenses move through the magnetic fields. This method is by far the most precise, 5 to 10 arc-seconds over a range of about 5°. Insertion of the lens requires a lot of care and practice to prevent injuries of the eye like corneal abrasion and or infections. Therefore, the lenses can only be worn for a relatively short time. The method is also not sufficient for measuring the point of regard, because the head position is not taken into account.

(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice, p. 53)

Electro-Oculography (EOG)



Electrode positions used in EOG

Electro-Oculography measures potentials generated by eye movements. These potentials do ***not*** come from oculomotor activity. They are in fact standing *corneoretinal potentials* (potential differences between the cornea and the retina) changing as the eyeball rotates. Other *corneoretinal potentials* originate from changes in illumination – so called illumination potentials.

1. In vertebrates the cornea is positive relative to the retina. In invertebrates, however, the anterior of the eye is negative. This corresponds to the difference in orientation of the retinas of subphylum vertebrata and the retinas of the invertebrates, the visual cells pointing outward in the former and inward in the latter.
2. The polarity of the standing potential is increased by light (the illumination potential) whether the animal has a positive cornea (vertebrate) or not (invertebrate).
3. There is a sudden discontinuity in potential at the ora serrata, which is found as one measures the topographic distribution of voltage.
4. Under certain influences, such as ionic changes or mechanical insults, there are independent changes of the standing and the illumination potential.
5. One finds a similar alteration in both the standing and the illumination potential under other influences, such as changes in carbon dioxide and oxygen tensions, temperature and electrical stimuli.

(cf: Marg, E (1951), p 4, Kohlrausch, A 1931)

The potentials generated by eye movements are in the range between 15 and 200 μV. The nominal sensitivity is about 20 $\mu V/deg$ of eye movement. Without additional head tracking or head fixation, EOG is not suitable for point of regard measurements.

If a general purpose bio-signal amplifier is already available, EOG is an inexpensive way of measuring eye movements. During sleep EEG recordings, saccadic EOG measurements are an easy way to analyze eye movements and a valuable tool for detecting and removing eye movement related artefacts in regular EEG signals.

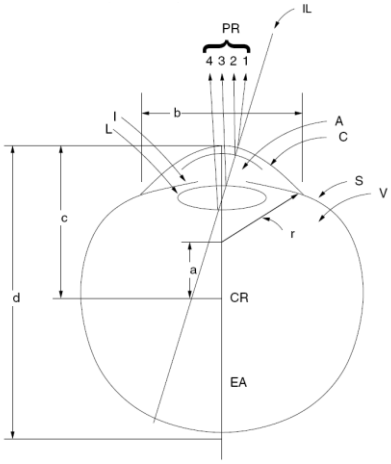
Although electrodes have to be placed around the eyes, this method is significantly less obtrusive than the Scleral Contact Lens/Search Coil method**.**

(cf: Marg, E (1951) Development of electro-oculography. AMA Arch. Ophthal. 45: 169.)
(cf: Kohlrausch, A 1931. : Elektrische Erscheinungen der Augen, in Bethe, A. ; von Bergmann, *G.;* Embden, G., and Ellinger, A,: Handbuch der normalen und pathologischen Physiologie, Berlin, Springer-Verlag, 1931, vol. 12, pt. 2, pp. 1393-1496.)

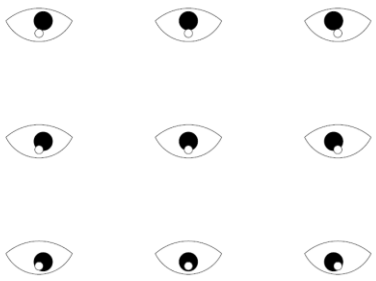
Video-OculoGraphy (VOG)

By utilizing inexpensive cameras and image processing hard and/or software, Video-OculoGraphy facilitates an elegant way to overcome the drawbacks found in EOG and search coil methods. Measuring the point of regard is considerably easier and much more comfortable even if the subject's head has to remain fixed all the time or if the subject is required to wear a head-mounted device. Non-invasive systems are completely unobtrusive and allow measurements over very long periods without any discomfort and health risks for the subject whatsoever. All video based systems take high speed videos of the moving eyes to measure their position. The ocular features used by VOG systems are the position of the pupil and/or the limbus. Additionally, corneal infra-red reflections, so called Purkinje images, are measured relative to the location of the pupil centre. Devices using this method are called Purkinje trackers. The Blink frequency e.g. to compute the degree of alertness is determined by measuring the distance between the upper and lower eyelid.

Example of limbus (left) and pupil tracking (right)



Purkinje images
1: front surface of the cornea
2: rear surface of the cornea
3: front surface of the lens
4: rear surface of the lens – almost same size and formed in the same place as the first image. Due to the refraction index of the lens, the intensity is less than 1% of of the first image.
IL, incoming light; A, aqueous humor; C, cornea; S, sclera; V, vitreous humor; I, iris; L, lens; CR, center of rotation; EA, eye axis; a≈6 mm; b≈12.5 mm; c≈13.5 mm; d≈24 mm; r≈ 7.8 mm
(Crane, H. D. 1994)
(cf: Andrew T. Duchowski p.57)



Relative positions of pupil and first Purkinje images as seen by the tracker's camera as the left eye rotates to fixate nine calibration points. The Purkinje reflection is shown as a small white dot in close proximity to the pupil.
The Purkinje remains stable, because the IR light remains fixed whereas the eye ball and therefore the pupil rotates freely in its orbit.
(cf: Andrew T. Duchowski p.58)

Two reference points on the eye are needed to separate eye movements from (minor) head movements. The positional difference between the pupil centre and corneal reflection remains relatively constant with minor head movement and mostly changes with pure eye rotation. To completely compensate head and body motion some kind of head tracking must be performed otherwise the head has to be fixed in its position.

(cf: Andrew T. Duchowski Eye Tracking Methodology: Theory & Practice)

(cf: Crane, H. D. (1994). The Purkinje Image Eyetracker, Image Stabilization, and Related Forms of Stimulus Manipulation. In D. H. Kelly (Ed.), *Visual Science and Engineering: Models and Applications* (pp. 13–89). New York: Marcel Dekker.)

JMFEL Core Classes

The Java Media Frame Work version 1.0 was developed by Sun Microsystems, Silicon Graphics and Intel in 1997, followed by version 2.0 developed by Sun and IBM in 1999. Its most recent version is 2.1.1e, released in 2003. It supports a wide variety of codecs, where additional codecs are available through ffmpeg-support. While the JMF has not been updated for many years, and despite its lack of support for more recent codecs (when used without FFMPEG), it is still the most complete media API for the Java language.

The following list briefly explains most of JMFEL's core and supporting classes. Some less important classes are not mentioned. These classes can be uses as simple building blocks to quickly develop sophisticated JMF applications that would otherwise require very detailed knowledge of the Java Media Framework.

Generic custom data sources

- *GenericCustomCaptureDevice*

  Used for data acquisition from a nonstandard source or device.

  A *CustomCaptureDevice* must subclass the *GenericCustomCaptureDevice* class.
- *GenericDataSource*

  A *subclassed GenericCustomCaptureDevice*
- GenericDeviceSourceStream

  A *GenericCustomCaptureDevice*

Data Sources

- *AudioDataSource*

  Used for accessing the systems' default audio input device
- *VideoDataSource*

  Used for Accessing the systems' default video capture device
- *LiveStreamDataSource*

  Used for Accessing any video device including simulation devices
- *SignalDataSource*

  Used for Accessing any signal device (no video!)

Cloning, merging and splitting

- *CloneableDataSource*

  Used for cloning data sources

- *DataSourceMerger*

  Used for merging several sources into one single source

- *SplitDataSource*

  Used for splitting a single source containing *N* several media streams (*SplitStream*) into *N* data sources each containing a single media stream

Media transcoding and editing

- *MediaTranscoder*

  Used for trancoding media into various formats

- Cut, *CuttingDataSource* and *CuttingStream* (these classes are taken directly from the JMF examples website http://java.sun.com/products/java-media/jmf/2.1.1/solutions). These classes perform media editing by cutting parts out of a given media stream.

Media presentation and processing

- *SimplePlayer*

  A simple Player for playing media.
- *SimpleProcessor*

  Used for processing media using codecs and plugins
- *ProcessorPlayer*

  A player for playing *and* processing media using codecs and plugins. When playing a *MergedDatasource* (cf: *DataSourceMerger*), each individual source can have its own individual renderer.
- *BasicAudioCodec*

  Used for accessing and processing chunks of data

  The key methods are processInData() and processOutData() to work with audio data.
- *BasicDataAccessor*

  Being a subclass of *BasicAudioCodec*, this class overwrites the processInData() method to write the data from the media buffer into a *DoubleDataBuffer* – a high performance ring buffer (see the utility class section for more). There is one *DoubleDataBuffer* per channel.
- *BasicAudioRenderer*

  Used for accessing, processing *and* rendering chunks of data.
- NullAudioDevice and RealNullAudioDevice

  These classes can be used with a custom audio renderer to mute or completely discard audio information, respectively. These devices will also eliminate delays caused by the audio hardware buffers during playback or recording.
- *NullAudioRenderer*

  This class is a direct subclass of *BasicAudioRenderer*, does nothing is therefore the most basic audio renderer. This renderer is usually used with a *NullAudioDevice* or *RealNullAudioDevice* to eliminate delays.

RTP Real-Time Transport Protocol (cf: RFC 3550)

- *RTPMediaTransmitter*

  Used for transmitting RTP media streams
- *RTPMediaReceiver*

Datasink

- *BasicCustomDataSink*

  This is a basic custom data sink

- *CustomDatabaseDataSink*

  This class is a subclass of the *BasicCustomDataSink* and implements a data sink that writes to a data base. This class needs a *Source Handler* to be able to handle the data from a source.

- *CustomFileDataSink*

  This class is a subclass of the *BasicCustomDataSink* and implements a data sink that writes to a file. **Important:** This is not to be confused with the JMF data sink. This data sink does not know how to handle the data that is coming from the data source by itself. Hence it always needs a *Source Handler*.

- *AbstractFileDataSourceHandler*

  The base class for all file related data source handlers.

- *AbstractDelimitedFileDataSourceHandler*

  A subclass of the *AbstractFileDataSourceHandler* to write into a CSV (Character Delimited Values) file. Popular examples are Comma Separated or Tab Delimited files.

- *AbstractDatabaseSourceHandler*

  The base class for all data base related data source handlers.

- *SimpleDataSink*

  A simple data sink. Features are: saving, trancoding, start, stop resume. Both, CaptureController and SimpleCaptureController use this class internally.

- *SimpleCaptureController*

  This convenience class uses the *SimpleDataSink* to easily write data to a file. Standard media containers are Quicktime and AVI.

- *CaptureController*

  More or less the same as the *SimpleCaptureController* but with an additional Monitor/Preview feature.

Custom Controller

- *CustomController*

  You might never need to use this class yourself unless you want to write new players.

- *DatasinkController*

  You might never need to use this class yourself unless you want to write completely new data sinks.

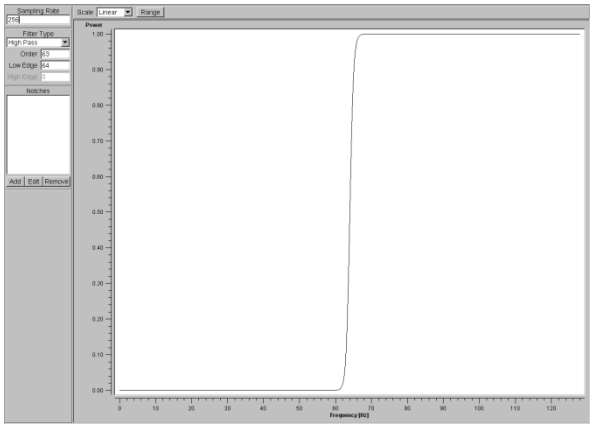
Support Classes

Signal Processing

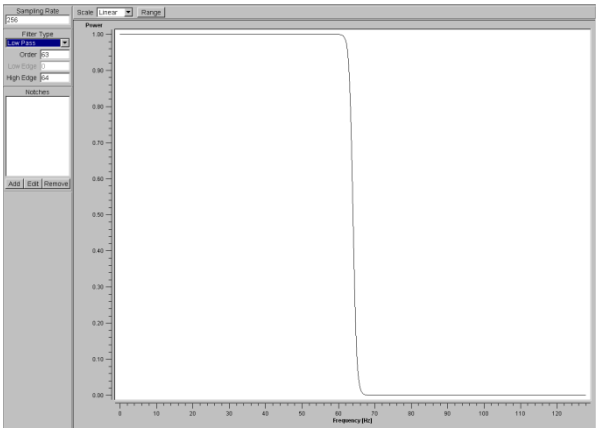
The signal processing class uses a JNI wrapper for Kevin Dolan's VecMat math library version 1.5. It only uses VecMat's signal processing, but not it's vector and matrix functions. Other signal processing libraries can be used by implementing a new JNI wrapper for that particular native library. This includes signal processing hardware. *The signal processing methods can be used independently from the JMFEL.*

Features: FFT, inverse FFT, IIR (Butterworth) filtering (low, high, band-pass and notch filtering), power spectrum, cross spectrum, Hilbert transformation, spectra (first and second power spectra, real and imaginary cross spectra), coherence, signal mean and variance.

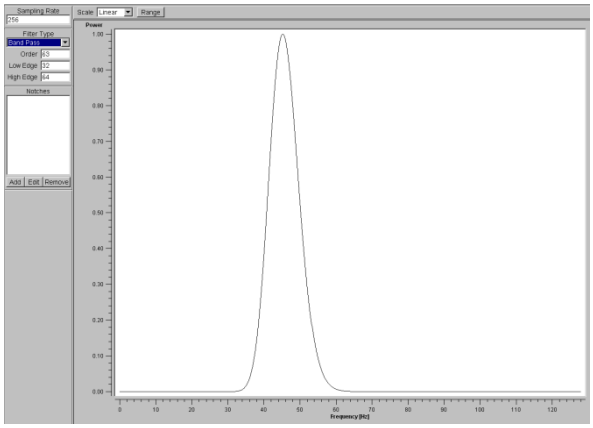
VecMat uses the FFTW, the industry standard for performing Fourier transforms and filtering. Please see the website (fftw.org) for a detailed performance analysis. Intel's Math Kernel Library (MKL) can be used instead of the FFTW. This might increase the performance.



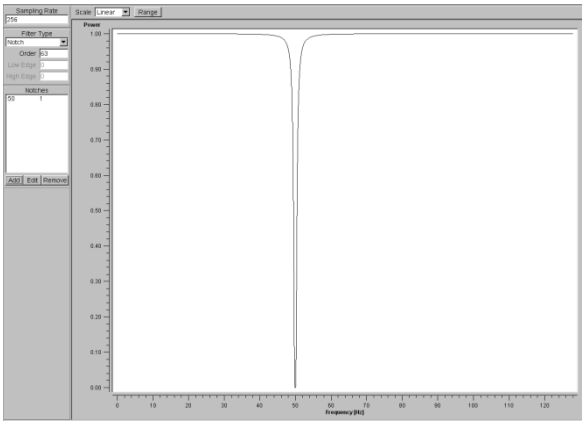
High pass filter; order 63; sample rate 256 Hz



Low pass filter; order 63; sample rate 256 Hz



Band pass filter; order 63; sample rate 256 Hz



50 Hz notch filter; width 1; order 63; sample rate 256 Hz

All figures were created with VecMat's filter editor. The scaling is linear, the sample rate is 250Hz and the order is 63.

All filters are IIR Fourier domain amplitude filters, hence there is, by construction, no phase shift. Any physical implementation of an IIR filter is going to have a nonlinear phase response. Fourier mode filtering is acausal, as it operates on the entire signal at all times. It is therefore possible to produce any phase response, including a flat phase.

That is, the Fourier transform of the function you convolve the signal with is a real, positive, symmetric function. Consequently, the cross-spectrum of the original signal and the signal filtered with an IIR Fourier domain amplitude filter is real.

Classes

There are two classes to perform signal processing, the *SignalProcessor* accesses the VecMat library through JNI and the *EEGSignalProcessor* class uses *the SignalProcessor* class to provide ready-to-use filters for EEG related signal processing.

- *SignalProcessor*

  This provides access to the signal processing methods.

- *EEGSignalProcessor*

  This class uses the *SignalProcessor* and provides the EEG band pass filters. This class is implemented as a singleton, hence there can be only one instance.

  Filter construction example:

  // design the filter for the delta band 0 - 3 Hz
  SignalProcessor processor = new SignalProcessor();
  Int order = 10;
  processor.newFilter(DELTA_LOW); **// set the filter ID to** DELTA_LOW
  processor.setFilterType(SignalProcessor.LOW_PASS, DELTA_LOW); **// set type to LOW_PASS**
  processor.setSampleRate((double) sampleRate, BAND8_10_LOW); **// set sample rate**
  processor.setOrder(order, BAND8_10_LOW); **//set order**
  processor.setHigh(3, BAND8_10_LOW); **//set high frequency**

  usage:

  double[] filtered_Signal = filter(original_signal, DELTA_LOW);

(cf: VecMat; http://kdolan1973.mine.nu/vecmat/software.htm)
(cf: Fastest Fourier Transform in the West FFTW, http://www.fftw.org/)
(cf: BLAS Basic Linear Algebra Subprograms http://www.netlib.org/blas/)
(cf: LAPACK The Linear Algebra PACKage http://www.netlib.org/lapack/)
(cf: Math Kernel Library (MKL), http://www.intel.com/cd/software/products/asmo-na/eng/307757.htm)
(cf: A.V. Oppenheim and R.W. Schafer, Digital Signal Processing (Prentice Hall, Englewood Cliffs, N.J. 1975))
(cf: P.D. Welch, in Modern Spectrum Analysis, edited by D.G. Childers, (IEEE Pres, New York, 1978))

Eyetracking

- *InsightDatagramPacketDecoder*

  Eye tracking data sent by the *Insight Eye Tracking System* is encoded in UDP packets and represented as byte arrays. This decoder class decodes the byte array and makes all values available through their respective methods. E.g. to get the opening of the left eye at the given time you simply call *getEyeOpeningLeft()*

  Design issue: The COGAIN project (Communication by Gaze Interaction) (cf: http://www.cogain.org) aims to develop and provide standardized software interfaces for commercial and non-commercial eye tracking systems. However, with only a small number of supported systems at this time, most eye tracking systems, such as the Insight Eye tracking System, still require a vendor specific decoder class.

  Right now classes directly call the decoder class. This is acceptable as long as we only support one specific eye tracker.
  Supporting various types of trackers requires some kind of standard or least common denominator. Without COGAIN support, one possible solution would be defining an interface type and implementing all decoders according to it. This would avoid refactoring and recompiling of all the classes depending on this standard decoder. But it is not the most elegant solution and unsafe since it is still possible to use vendor specific code by casting the decoder from its interface to its actual type. A far better and safer solution is to build up a facade to define the least common denominator, thereby prohibiting the application programmer from using vendor specific features.

This approach guarantees that all eye tracking applications will work with any eye tracking hardware according to the standard defined by the facade. Furthermore, the strategy pattern can be applied for easy switching between vendor specific decoders. Incompatibilities e.g. in terms of method names can be easily fixed by adapters.

Thus, resulting debug code and fixing problems is a lot easier and safer, since no user is using vendor specific decoders directly.

The price for this safety and compatibility is that new features will only be available with new versions of the facade, but this is still better than uncontrolled growth.

- *EyeTrackerCalibrationServer*

  This application is running on the host machine and displays calibration points for each area of interest. The application assumes up to 42 uniformly distributed area of interest or calibration points. Active areas of interest are defined in the file scene.csv. The server reads its configuration from the file *server.ini*.

  Sample configuration: IP=192.168.1.215, PORT=10001
- *EyeTrackerCalibrationClient*

  This application is running on the client machine, reads the coordinates from the aoi_positions.csv file and displays or hides the area of interest indicated by the calibration server on the client machine.

  Sample configuration: IP=192.168.1.215, PORT=10001, FULLSCREEN=true, WIDTH=1024, HEIGHT=768
- media/ protocol/ eye/ *DataSource*

  subclass of GenericDataSource
- media/ protocol/ eye/ *CustomCaptureDevice*

  subclass of GenericCustomCaptureDevice
- media/ protocol/ eye/ *DeviceSourceStream*

  subclass of *GenericDeviceSourceStream*
- media/ protocol/ eye/ config.ini

  Sample configuration: UDP_PORT=10000, SAMPLERATE=60, BITS_PER_SAMPLE=8, IP_ADDRESS=192.168.1.174, HORIZONTAL_MAXIMUM=350, HORIZONTAL_MINIMUM=-350, VERTICAL_MAXIMUM=510, VERTICAL_MINIMUM=-50

- *EyeTrackerBlinkPanel*

  Class used for drawing the eye tracking blink signals
- *EyetrackerInfoPanel*

  Class used for displaying information about head position and movement
- *GazePanel*

  Class to display the gaze using the decoder classes' methods:

  getCombinedGazeVectorFocalPlaneIntersectionX()

  getCombinedGazeVectorFocalPlaneIntersectionX()

  Conversion of tracker coordinates to screen coordinates is done by the *CoordinateConverter* class
- *GazeVideoPlugin*

  This plugin takes the gaze position from the eye tracker and draws a crosshair showing the current gaze position on top of a video stream
- resources/ eyetracker/graphics

  Graphics used by eye tracking related classes
- *EyeTrackerApplication*

  Application showing the blink signals of both eyes
- EyeTrackerGazeApplication

  Application showing the gaze signal
- EyetrackerDataSourceHandler

  source handler for used for writing eye tracker data to a data base

EEG

- media/ protocol/gtec/
  this package contains the java API and tools to work with the g.USBamp EEG amplifier
- media/ protocol/ usbAmpA/ *DataSource*
  subclass of GenericDataSource
- media/ protocol/ usbAmpA / *CustomCaptureDevice*
  subclass of GenericCustomCaptureDevice
- media/ protocol/ usbAmpA / *DeviceSourceStream*
  subclass of *GenericDeviceSourceStream*
- media/ protocol/ usbAmpA /configuration.ini
  configuration written by the *CalibrationWizard*
- media/ protocol/ usbAmpA / scaling.ini
  configuration written by the ScalingWizard
- media/ protocol/ usbAmpA / serial.ini
  each amplifier is opened using its serial number
  Note:*,../ usbAmpB /, ../ usbAmpC /, ../ usbAmpD / contain data sources for amplifiers B, D and C.*
- *BasicEEGDataModel*
  Holds EEG data and calls all its observers every time the data is updated
- *EEGDataModel*
  Performs signal processing on EEG data and calls all its observers every time the data is updated and filtered
- *ElectrodeAssignment*
  This list is based on a (near) equidistant 76-channel electrode arrangement
  Usually you will use this list to read and write a vertext to assign the actual channels to the electrodes. E.g. CZ could be assigned to channel 5.
  This class uses java.util.Properties to perform the mapping. The *ElectrodeAssignment* class has methods to read and write mappings from / to disk.
- AlphaBetaPanel
  Displays the alpha and beta power EEG channels C3 and C4. It uses percentage bars to indicate the band power for each 2 Hz sub band and it uses the *ElectrodeAssignment* class to map C3 and C4 to their respective channel. The channels C3 and C4 electrodes are connected to the amplifier. This mapping is defined in resources\ eeg\

channels_to_electrodes_mapping.ini. For visualization purposes e.g. surface laplacian channels need to be mapped to vertices. For that you would use resources\ eeg\ channels_to_vertex_mapping.ini. **These are just default mapping files and locations. You can place mappings wherever you want.**

- *ChannelNumberPanel*

  This panel displays a channel number.
- *EEGEnergyPanel*

  This panel displays the energy of common EEG bands as percentage bars
- *EEGPowerPanel*

  This panel displays the power of common EEG bands as percentage bars
- *EightChannelPanel*

  This panel displays eight channels at once
- *MultiChannelEEGPanel*

  This panel displays multiple (16) channels at once
- *MultiBandEEGPanel*

  This panel displays classic EEG sub bands of a single channel
- *SingleChannelEEGPanel*

  This panel displays a single channel at once.
- *SpectrumPanel*

  This panel displays the signal spectrum of a signal. It uses the *EEGDataDodel* and the *EEGSignalProcessor* class to compute the spectrum
- *EEGDataPlugin*

  being a subclass of the *BasicDataAccessor*, it provides access to the chunks of data of the EEG signal.
- *EEGMotorCortexLaplacePlugin*

  This plugin takes the average of the electrodes that surround C4 and C3, the center electrodes and subtracts it from C3 and C4 respectively. This is another application of the *ElectrodeAssignment* class. Using the electrode's standard names instead of nameless channel numbers results in much more readable and maintainable code.
- *EEGEnergyApplication*

  Uses the *EEGEnergyPanel* to display the energy in EEG sub bands
- *EEGPowerApplication*

  Uses the *EEGPowerPanel* to display the power in EEG sub bands

- *EEGMultiBandApplication*

  Uses the *MultiBandEEGPanel* to display classic EEG sub bands of a single channel
- *MultichannelEEGApplication*

  Uses the *MultiChannelEEGPanel* to display classic EEG sub bands of a single channel
- SingleEEGChannelContainer

  Uses the *SingleChannelEEGPanel* to display the signal of single channel
- *SpectrumAnalyzerApplication*

  Uses the *SpectrumPanel* to display the signal spectrum of a single channel

ECG

- *ECGPanel*

  This panel displays the ECG signal, an estimated heart rate and highlighted R parts. The algorithm to calculate the heart rate is quite possibly the simplest adaptive thresholding algorithm ever. Yet, it's very effective and reliably extracts the R parts (peaks) of QRS complexes for hours without missing a beat. The algorithm works on a 10 second ring buffer and counts R parts (peaks) to estimate the heart rate per minute. The adaptive thresholding assumes the shape of *healthy* looking QRS complexes.

  This is a proof of concept – an example – *not* a clinical application

```
// This is the complete algorithm to extract and count R parts within a 10 second ECG signal
double maximum = 0.0;

    for (int i = 0; i < bufferSize; i++) {
       maximum = Math.max(maximum,signalArray[i]);
    }

    for (int i = 0; i < bufferSize; i++) {
       double value = signalArray[i]-maximum/2.0;
       if(value>0){
          signalArray[i] =
getDoubleDataBufferContainer().getDataBuffer(getECGChannel()).getMaxVoltage();
          withinPeak = true;
       }else if((value<=0)){
          signalArray[i] = 0.0;
          withinPeak = false;
       }
       if(lastPeakState&&!withinPeak){
          peakCounter++;
       }
       lastPeakState = withinPeak;
    }
    heartRate = peakCounter*6;
```

- *ECGDataPlugin*

  This plugin provides access to chunks of ECG signals
- *ECGApplication*

  Uses the ECGPanel to display an ECG signal

Video

- *VideoDataAccessor*

  The key methods are processInData() and processOutData() to work with video data. In principle this class uses a novel approach that is quite different from SUN's method. The video data is copied back and forth between a compatible Java2D *BufferedImage* and then back to the JMF videobuffer. This makes it possible to use Java2D methods on JMF video data. Because System.arraycopy() is used, it is as fast possible. The JMF provides two classes *BufferToImage* and *ImageToBuffer* for the very same purpose. The JMF classes work sufficiently well on today's machines. On older machines, they show their great inefficiency.

  The *System.arraycopy()* approach has originally been designed several years ago for a 1 Ghz AMD Athlon system. Back then, *BufferToImage* and *ImageToBuffer* achieved about 10 fps, because the CPU usage was always at 100%. Using *System.arraycopy(),* the very same computer was able to achieve 30fps and a CPU usage of only 3 to 5%.
- *VideoDate*

  A video plugin based on *VideoDataAccessor* which paints the current date and time into video frames

Utility classes

- *ArrayUtility*

  This class provides methods to convert a matrix (n-dimensional array) into a one-dimensional array and back into a matrx. This is particularly convenient for using BLAS and LAPACK matrix routines on multichannel signals.

  These methods are e.g. used in the *EEGMotorCortexLaplacePlugin*.

- *CSVReader*

  Utility class to read CSV files. The separator is the comma

- *CSVWriter*

  Utility class to write CSV files. The separator is the comma

- *PropertiesReader*

  Class to read java.util.Properties from disk

- *PropertiesWriter*

  Class to write java.util.Properties to disk

- *XMLDataProvider*

  Utility class for working with XML files. **org.apache.xerces is required!**

- *ByteUtility*

  Utility class to join two bytes into a 16bit value and to split a 16bit value into two bytes. The endianess can be configured e.g. to convert between network (big endian) and x86 (little endian) format

- *DoubleDataBuffer*

  Very high speed array based and fixed sized ring buffer. Each ring buffer features two separate callback mechanisms. *One* buffer can call exactly *one* registered object. For that, the registered object has to implement the *Callable* interface. Each buffer also employs the Observer Pattern by extending Java's built-in Observable class.

- *DoubleDataBufferContainer*

  Container that holds *N* instances of ring buffers (*DoubleDataBuffer*)

- *SignalPanel*

  Used to draw signals. Data is provided read from DoubleDataBuffer or from primitive array

- *PercentageBar*

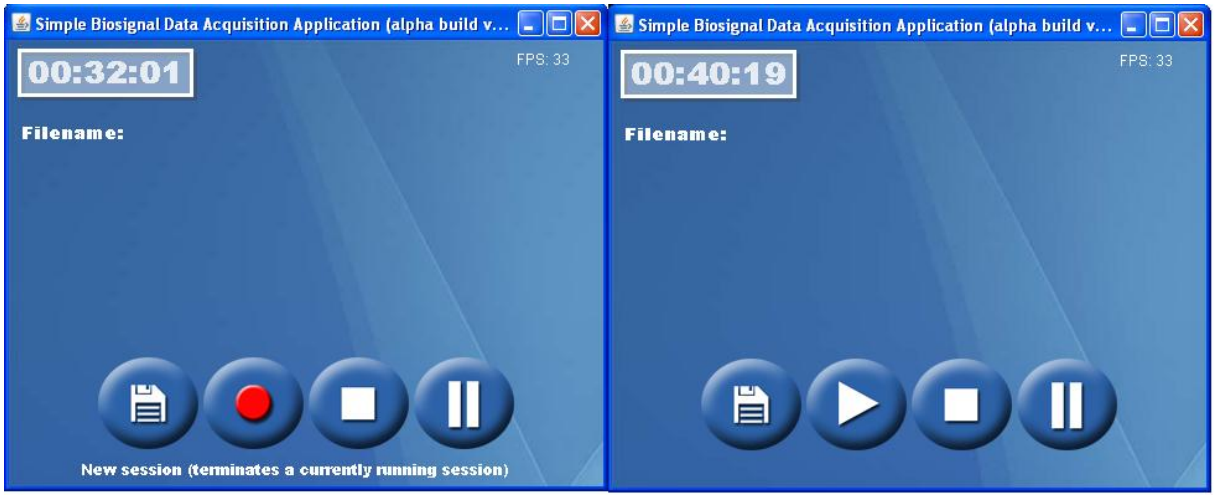
  Used to draw horizontal and vertical percentage bars

- ClockPanel

  Panel to draw a clock on any drawing surface

Reusable GUI

The *CaptureStateApplication* can be used as a simple GUI for various capture and play back applications. The applications can be run in recording or playback mode.



The *CaptureStateApplication* is a simple simple reusable GUI. It can be run in either recording (left figure) or play back mode (right figure). The tool tips shown at the bottom can be defined in a small configuration file. The tool tips depend on the where mouse pointer is located.

Classification

- *Classifier*

  Every Classifier must implement this interface.

- *ALNClassifier*

  This class implements the *Classifier* Interface. The classifier used by this class is the commercial version of the Dendronic Learning Engine which uses Adaptive Logic Networks (ALN). The actual classifier is a decision tree that was generated by ALNFit Pro, which automatically generates ALN classifiers after appropriate statistical analysis. The file is loaded from disk by the *ALNClassifier* class during initialization.

  Performance:

  Test data: The well known Wisconsin Breast Cancer dataset

  Size: 570 samples and 30 features per sample

  Misclassifications during validation: 1 out of 56 test samples

  Runtime performance: 13680 classifications per second

  The tests were performed on a standard PC with a Core2Duo 6700 CPU at 2.67 GHz. The code was not optimized for this specific CPU.

  (cf: Open source version of the Dendronic Learning Engine; http://www.dendronic.com)

  (cf: W. Armstrong and M. Thomas. Adaptive Logic Networks, *Handbook of Neural Computation*, Emile Fiesler and Russell Beale, editors, Oxford University Press, 1996, pages C1.8:1 - 14.)

  (cf: A. Kostov, W. Armstrong and M. Thomas. Adaptive Logic Networks in Rehabilitation of Persons with Incomplete Spinal Cord Injury, *Handbook of Neural Computation*, Emile Fiesler and Russell Beale, editors, Oxford University Press, 1996, pages G5.1:1 - 8.)

  More papers and publications: http://www.dendronic.com/articles.htm

Graphics Framework

Normally, to draw something with Java2D, you need to extend a JPanel and overload its paint method. This approach works. But in terms of reusability, interoperability and coupling it is a very bad thing, since the drawing code is very closely tied to the panel. Thus, you have to always use the panel in order to paint its content on the screen or – even worse – copy its drawing code! If you want to change something you *might* get in trouble. But if you want the same graphics to be painted in an application that uses active rendering or OpenGL (using the Java2D/JOGL interoperability bridge) you *will* get in trouble, which means you have to rewrite the whole drawing code. Redundant copies of code leads to double maintenance and eventually to broken code. To overcome these problems, the drawing code and the drawing surface have been decoupled as much as possible. The result is a small framework that facilitates Java2D drawing code, which is independent of its drawing surface.

- *AbstractByteImage*
  Provides direct access to the byte array of a BuffererImage
- *ByteImage*
  sub class of *AbstractByteImage*
- *BasicImageUtility*
  Utility class to load and manipulate images
- *ColorSpaceConverter*
  Utility class to convert between different color spaces
- *CoordinateConverter*
  Utility class to convert coordinates one 2D coordinate system into another. This class is e.g. used to convert eye tracker coordinates to screen coordinates
- *PixelColor*
  Utility class to store RGB pixel colors
- *StringHelper*
  This class calculates the width and height of strings with respect to the font. The values are given in number of pixels

- *Vertexhelper*

  Utility to extract the coordinates of red pixels in an input image. This was used to extract the coordinates of the 42 areas of interest from an image. Instead of writing a vertex editor you can just use any graphics program for that pupose. The *Vertexhelper* writes a list of coordinates into a text file. The coordinates/vertices are within 0.0f and 1.0f and ordered from left to right and top to bottom.

- *GraphicsSurface*

  Interface that any graphics surface has to implement

- *AbstractPainter*

  All so called panel components that are mentioned above are *not* AWT or Swing JPanels. Instead, these panels contain Java2D drawing code which is called by their respective painter classes (sub classes of AbstractPainter). All painters execute the panels' drawing code with respect to a *GraphicsSurface.*

- *PassiveRenderingPanel*; *ActiveRenderingPanel*

  This panels are both abstract classes and derived from JPanel. The passive rendering panel is for passive rendering and the active panel is for active rendering. (cf: http://java.sun.com/docs/books/tutorial/extra/fullscreen/rendering.html)

- *NullRepaintManager*

  This repaint manager consumes all system events e.g. redrawing events. Therefore this manager is only active in full-screen exclusive mode.

- *ApplicationContainer (active rendering)*

  This is either a subclass of *AbstractActiveRenderingFrame* which itself is a subclass of JFrame – for active rendering – or in the case of passive rendering, a direct subclass of JFrame. An *ApplicationContainer* literally contains the application. A container contains the surface and takes care of repainting all panel components at the desired frame rate.

- *Surfac*e

  A surface is a subclass of its respective rendering panel. That is, a surface used for active rendering extends the *ActiveRenderingPanel* and a surface used for passive rendering extends the *PassiveRenderingPanel*. All surfaces implement the *GraphicsSurface* interface to assure full interoperability with rendering code - the panel components. Since all Surfaces implement the *GraphicsSurface* any rendering code will work with any Surface implementation.

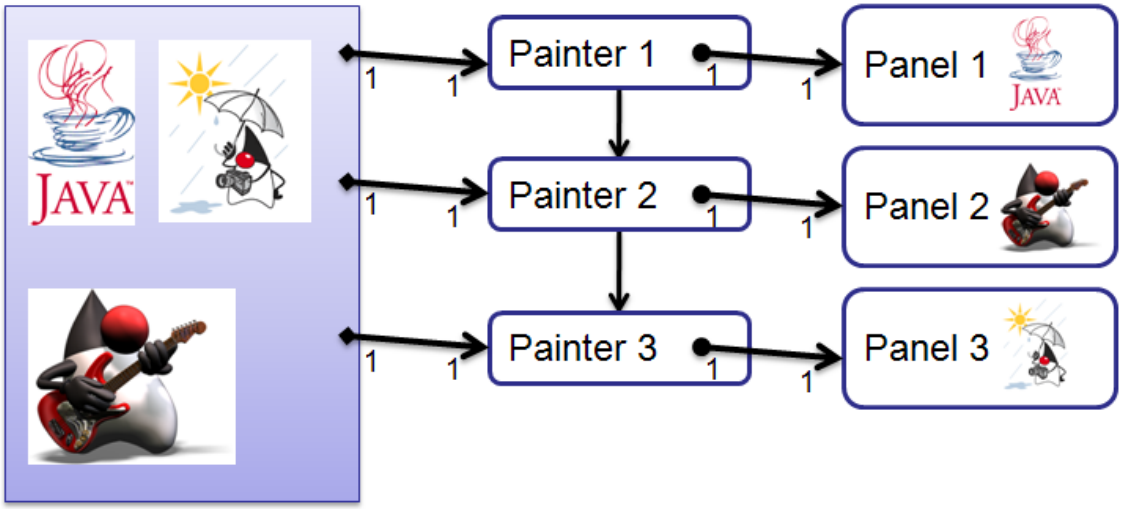
Implementing new Surface

Right now only active and passive rendering is supported. In order to use OpenGL rendering while at the same time maintaining full interoperability with existing rendering code (panel components) an OpenGL Surface – actually the OpenGL rendering panel – would have to use the Java2D/JOGL interoperability bridge. At least with up to three different rendering types, a factory – surface factory – should be considered to instantiate drawing surfaces. This virtually eliminates any refactoring and recompiling of application code and makes choosing or switching between different drawing surfaces at runtime possible.

Although the *ApplicationContainer* used for active rendering *could* be used directly with an OpenGL rendering surface, it is probably a good idea to subclass this container and put it into a separate package called e.g. **opengl_rendering**. This might add a slight amount of overhead, but it does make code more readable, which in this case is much more important.

How drawing on a Surface works

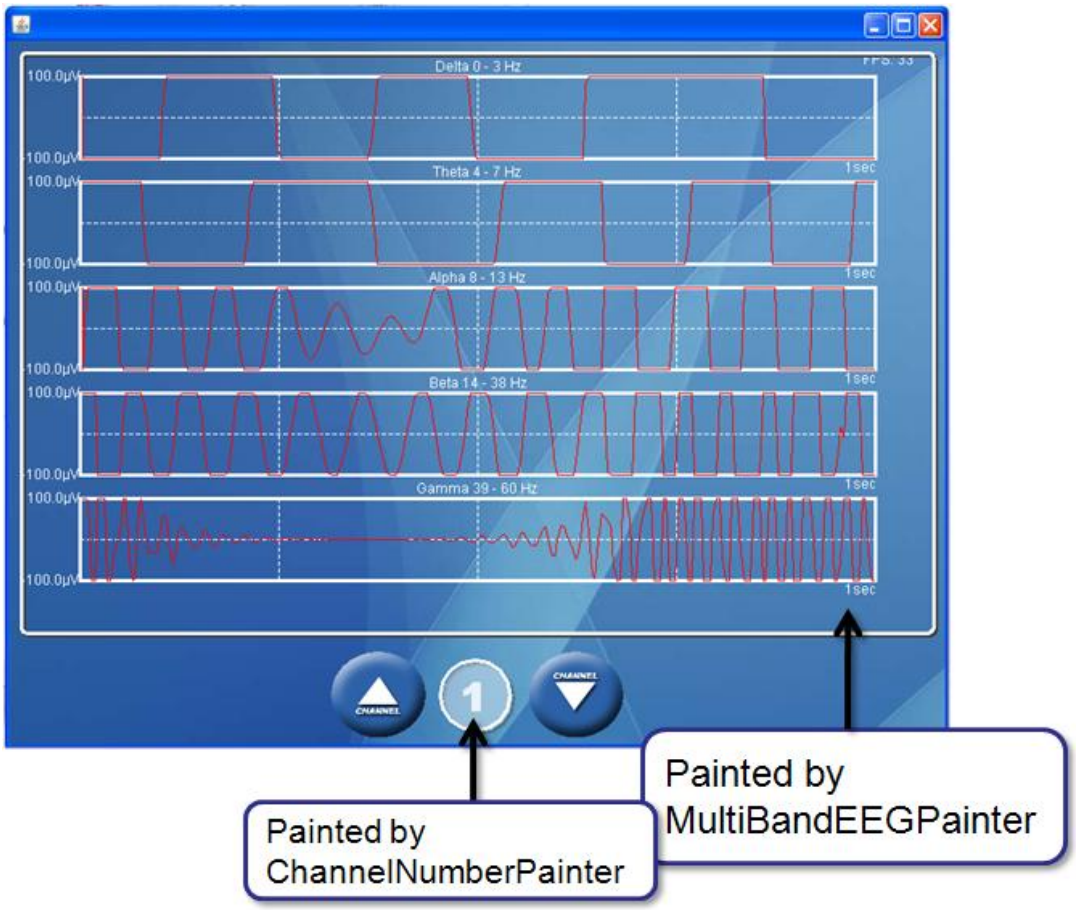
To draw on a surface, a panel component has to register itself at a surface through its individual painter (subclass of *AbstractPainter*). That is, any surface employs its own "rendering queue" or callback mechanism to execute every panel component's drawing code. Painter objects referencing panel components are registered to surfaces and ***not*** the panel components themselves. This indirection is needed to decouple the drawing code from the actual surface type. In terms of the observer pattern, the surface is made observable and painter objects observe the surface for changes – refresh cycles or messages.



Example: Three *Painter* are registered at the drawing *Surface*. One *Painter* references exactly one *Panel* and its surface independent drawing code. For each frame the drawing *Surface* calls each of the three *Painter* back to paint its *Panel*. This works, because for every frame each *Painter* passes the surfaces' graphics context to its *Panel*. The panel's position is only known to its individual painter, *not* to the panel and *not* to the drawing surface.

 http://www.go-java.com/images/duke_winter.gif  official Java logo (1996-2003)  java.sun.com

Duke and the Java logo are registered trademarks of Sun Microsystems, Inc.

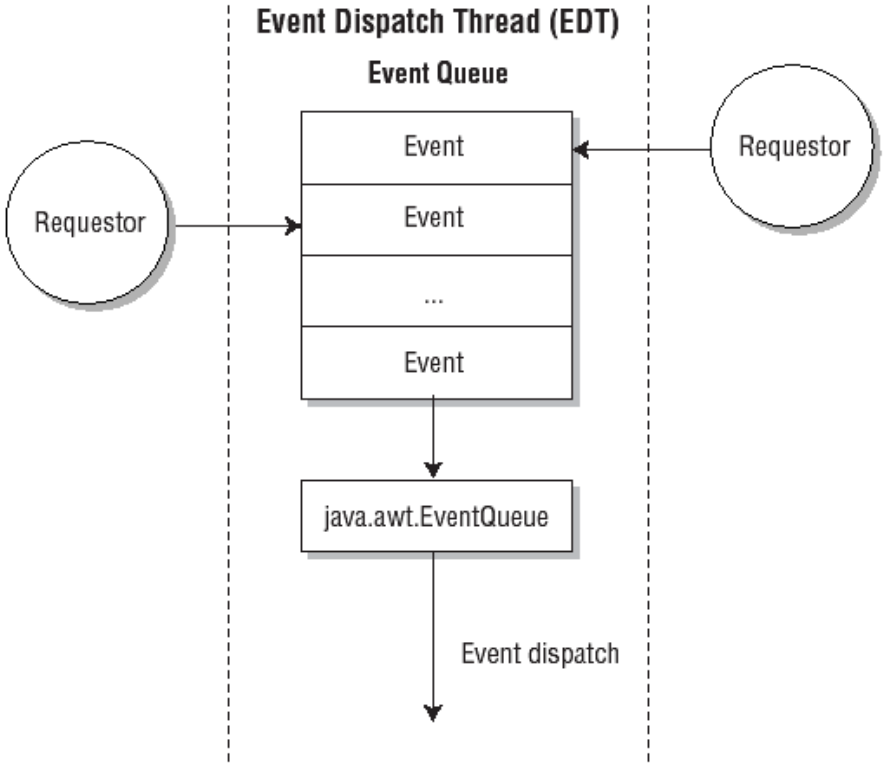


This screenshot of the MultiBandEEGPanel shows one second long EEG sub bands of a continuous test signal (rect signal). The background image, the buttons and the frame counter (top right) are directly painted by the graphics surface. The channel number between the buttons is painted by the *ChannelNumberPainter* and the signals are painted by the *MultiBandEEGPainter* which gets its filtered data from the *EEGDataModel*. The MultiBand-EEGPanel is just a composite of several reusable components. None of the components are specifically made just for this panel.

The background image is adapted from Apple's OS X Aqua user interface and can be changed by the user. OS X and Aqua are registered trademarks of Apple Inc.

Using Swing Timers to improve rendering performance

It's a common misconception that Java2D is slow. This is probably because all official examples use AWT threads for triggering repaints. Using AWT threads for rendering graphics can cause a high system load, because they block the Event Dispatch Thread (EDT) which also handles key and mouse events and makes the cursor blink. The EDT wraps requests in an event and posts it onto the event queue (java.awt.EventQueue). This is the only correct place to handle repaint requests. Consequently, all drawing surfaces implemented in the graphics framework use Swing timers and the EDT to achieve high drawing performance and constant frame rates.



Event posting and dispatching in Swing happens via the EventQueue, which runs on the Event Dispatch Thread (EDT)

(cf: Filthy Rich Clients, Chat Haase, RomainGuy, Prentice Hall (2007), Chapter 2, p.16)

Synchronous instead of asynchronous rendering

All surfaces use synchronous instead of asynchronous rendering.

In asynchronous rendering, repaint requests get “coalesced” or combined and all repaint requests for a specific component are ignored unless all prior requests have been processed by the EDT. This is ok for small, simple and not very complex rendering tasks which do not happen very often. Rendering signals or complex animations accurately at constant and high frame rates requires synchronous rendering. This is implemented by paintImmediately() which executes all paint requests, well, immediately. This method tells a component that a specified area must be updated; Swing calls paint() internally on all of the component’s subcomponents (it the component is a composite) to make this happen.

(cf: Filthy Rich Clients, Chat Haase, Romain Guy, Prentice Hall (2007), Chapter 2, p.16, 17)

Synchronization

JMF media synchronization can be achieved in different ways.

Multiple Players

Players can be synchronized by associating then with the same time base.

```
player1.setTimeBase(player2.getTimeBase());
```

These players are then started by synchStart().

Synchronizing players by associating them with the same time base requires managing the control of each Player individually. That is, each player must control each player's state individually, listening for events and calling control methods on them as appropriate.

Players can also take control over other players by using their controller.

```
player2.addController(player1);
```

Players can also be controlled directly

- call setMediaTime() to all of the players
- prefetch all players to prepare them to start
- stop all players when the second media time request is received
- call setMediaTime() with the new media time to all of the players
- restart prefetching
- When all players are prefetched, call syncStart() on all players

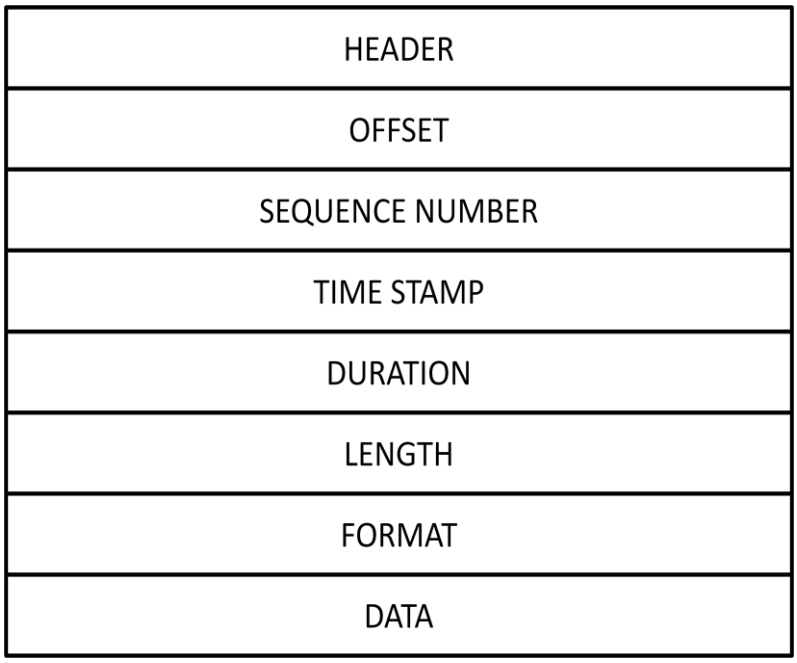
Synchronizing multiple player is very difficult and the accuracy is usually disappointing.

(cf: Java Media Framework API guide (1999), p.58-61)

Single Player

The JMF uses mechanisms described in section 5 and 6.5.1 of the RTP protocol specification to synchronize merged media streams, perform jitter calculations and maintain a constant playback rate. JMF uses frame-based media. That is, each media stream is a sequence of consecutive frames or media buffers.

javax.media.buffer



Properties of javax.media.buffer. Each media stream is a sequence of media buffers

The JMF uses a master clock to generate timestamps as described in RFC 3550. The clock increments monotonically and linearly in time and has nanosecond precision, but *not* necessarily nanosecond accuracy. Java's nanosecond timer uses the most accurate system timer. The accuracy depends on the Operating System and on the CPU. Each media buffer has a unique sequence number. The timestamp is generated by the master clock. The buffer length can vary between two consecutive buffers (variable buffer length). This does not affect the overall playback rate given by the sample rate. However, the buffer length must be large enough to guarantee a continuous stream of samples without any interruptions (see formula below). The duration can be unknown; a live stream probably has no predefined length.

The buffer size depends on the sample rate and bit size; the higher the sample rate and bit size, the larger the size of the buffer.

*scanSize = sampleRate * lengthInSeconds * 0.001*

*bufferSize = scanSize * numChannels * sizeOf(float) + HEADER SIZE*

This is formula computes the maximum buffer size according to the g.USBAmp software manual. Formulae will vary for different devices.

JMFEL's *DataSourceMerger* automatically enables synchronization by altering the "cname" (Canonical end-point identifier SDES item, cf: RFC 3550 Section 6.5.1) entries for each media stream. Consequently, the *RawSyncBufferMux* multiplexer will be used instead of the default *RawBufferMux* multiplexer.

The *RawSyncBufferMux* multiplexer uses the track, which has the highest sample rate, as a master track and synchronizes all other media tracks with it. Therefore, a video track will always be synchronized with its audio track, but the audio track will never be synchronized with its video track.

Applied to movie making in the real world, this would be like getting the voice actors to record the audio track first, and then start shooting the actual movie accordingly.

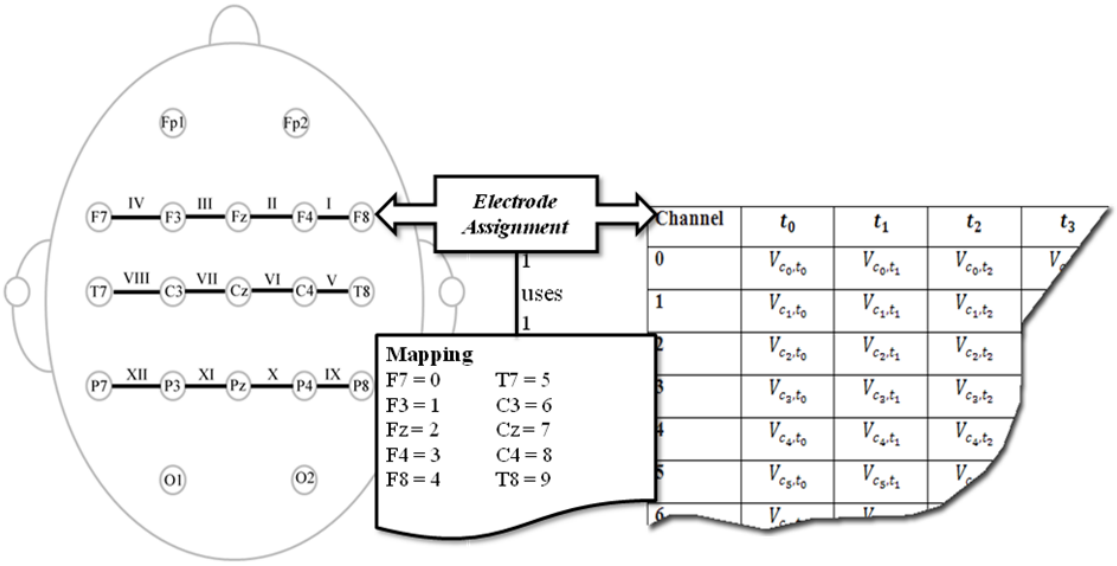
This mechanism is JMF's most precise synchronization method and can be used for all merged media streams. Since all media streams are played and processed by a single player, there is no need for player synchronization.

From now on media synchronizations is generally referred to this method.

Datasink

Like the data source merger, JMF's *DataSink* facilitates synchronization by multiplexing several data streams into one. Internally, the JMF uses the very same synchronization algorithm as for RTP or any other merged media streams.

Electrode Mapping



10-20 Electrode Montage JMF Media Buffer

Bipolar derivation scheme. Each roman numeral represents one bipolar derivation. E.g. IV represents the bipolar derivation between F7 and F3. The *ElectrodeAssignment* class facilitates a mapping between electrode symbols and channel numbers. The media buffer on the right side has been converted into a matrix using *ArrayUtility.arrayToMatrix()*. The channel number of e.g. electrode Fz is determined by electrodeAssignment.getFz(). Each column of the matrix represents a value in time. Thus, V_{c_2,t_0} represents the value of Cz at buffer position t_0.

The *ElectrodeAssignment* class facilitates a mapping between up to 76 standard electrode symbols and numerical values. Plug-ins e.g. the *EEGMotorCortexLaplacePlugin* use this class to access channels by their name instead of hard-coded and meaningless numerical values. The content of a JMF media buffer should be converted into matrix (two-dimensional array) for easy access of channel's sample values with a simple for loop. This can be achieved by *ArrayUtility.arrayToMatrix()* and *ArrayUtility.matrixToArray ()*. The computational overhead for this conversion is the same for every buffer and insignificant even for many channels and high sample rates.

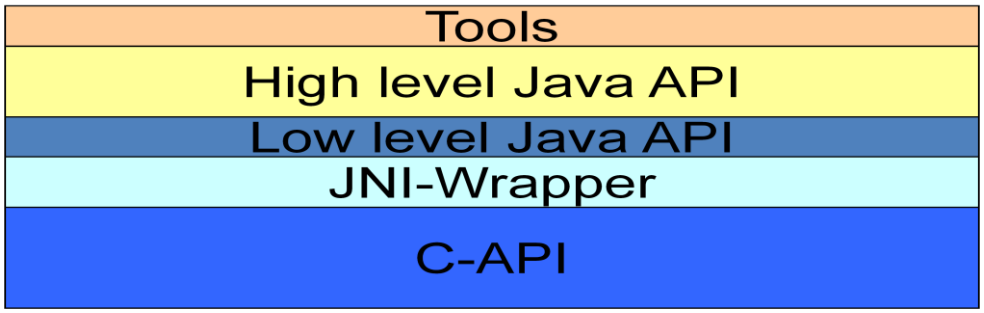
```
ElectrodeAssignment electrodeToChannelAssignment;
double[][] matrix = arrayToMatrix(getInData());
for (int p = 0; t < getTransferBufferSize(); t++) {
    bipolar_I  =  matrix [electrodeToChannelAssignment.getF4()][t] - matrix [electrodeToChannelAssignment.getF8()][t];
    bipolar_II  =  matrix [electrodeToChannelAssignment.getFz()][t] - matrix [electrodeToChannelAssignment.getF4()][t];
    bipolar_II  =  matrix [electrodeToChannelAssignment.getF3()][t] - matrix [electrodeToChannelAssignment.getFz()][t];
    bipolar_IV =  matrix [electrodeToChannelAssignment.getF7()][t] - matrix [electrodeToChannelAssignment.getF3()][t];
}
```

This example shows how to perform the bipolar derivations I, II, III and IV as shown in the bipolar derivation scheme.

The calculations are performed for buffer positions t_0, t_1, t_2 … t_n with n = buffer size. **Important:** This approach is not limited to simple calculations. An ICA algorithm could be applied in the very same way to each media buffer and effectively to the whole signal.

g-tec Java API

This API has been developed to access the g.USBAmp EEG amplifier from Java. It consists of three layers and basic tools. The API has also been used to develop the JMFEL data sources.

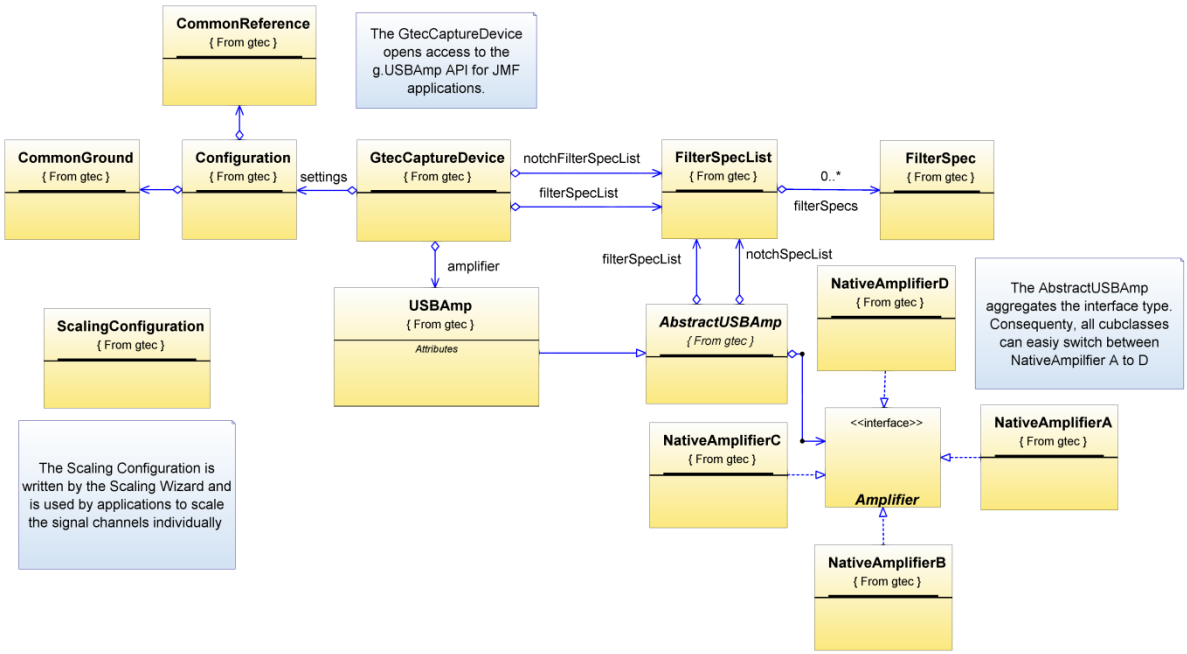


The layered structure of the *g-tec* Java API

The JNI-Wrapper is the link between Java and the C/C++ API for the g.USBAmp. The low-level API is not object oriented and does not facilitate any exception handling. Errors are therefore indicated by numerical values that are returned by the C/C++ API. Based on the low-level API, the high-level API is object oriented and does provide exception handling.

The tools are stand alone GUI-applications and intentionally resemble the look and feel of the respective Matlab tools, which are part of the g.USBAmp Matlab API.

The design of the *g-tec* Java API



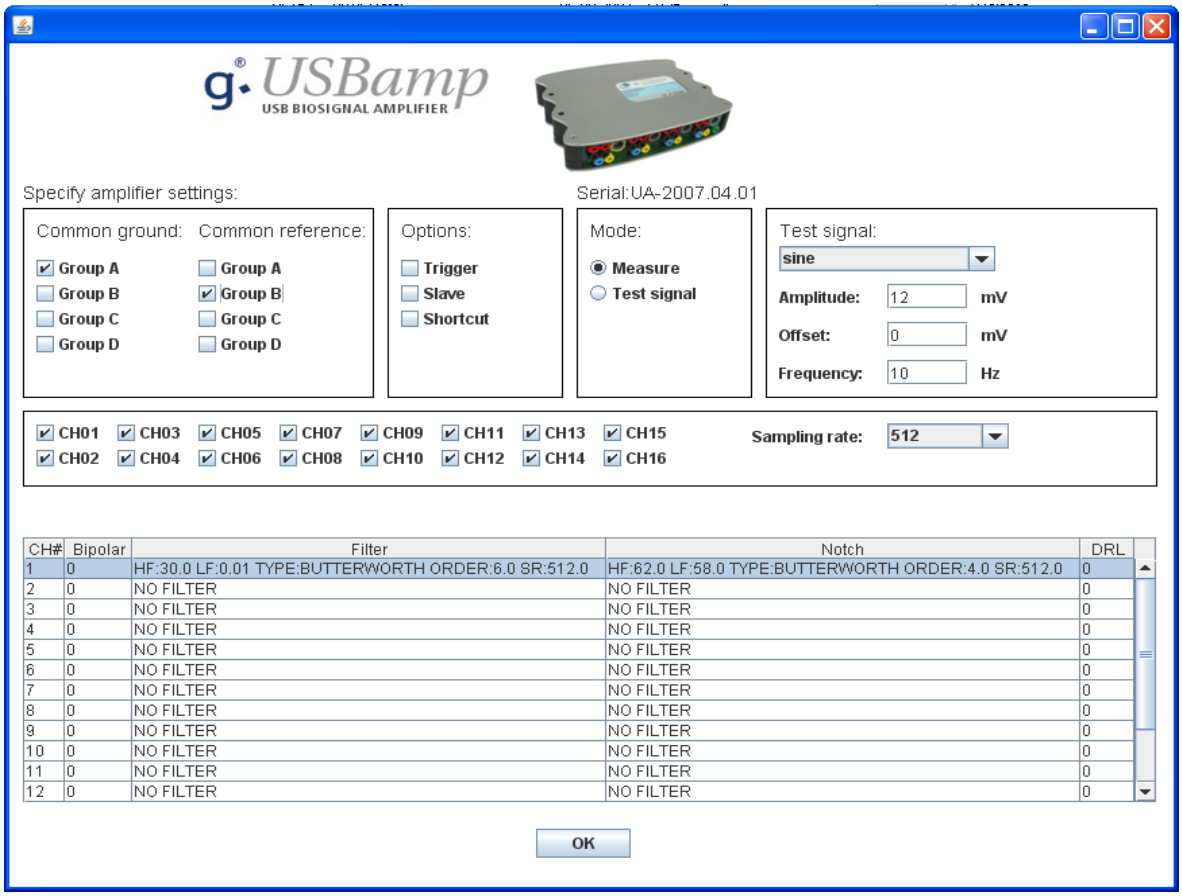
The Java API opens easy access to the g.USBAmp C/C++ API. The GtecCaptureDevice integrates into the JMFEL. This is covered in detail in the JMEFL Framework Design chapter.

Tools

All tools use the Java API to access the g.USBAmp amplifier. They intentionally resemble the look and feel of the g.USBAmp Matlab tools. All tools accept command line parameters and except the one for checking the impedances modify the configuration of the respective amplifiers.

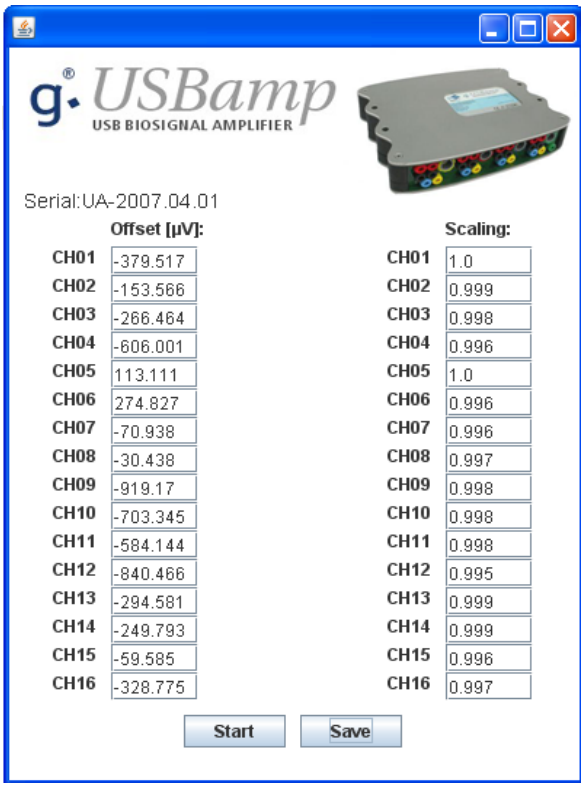
(cf: g.USBAmp Matlab API documentation)
(cf: Appendix B; How to use the Java tools for the g.USB Amp)

Configuration



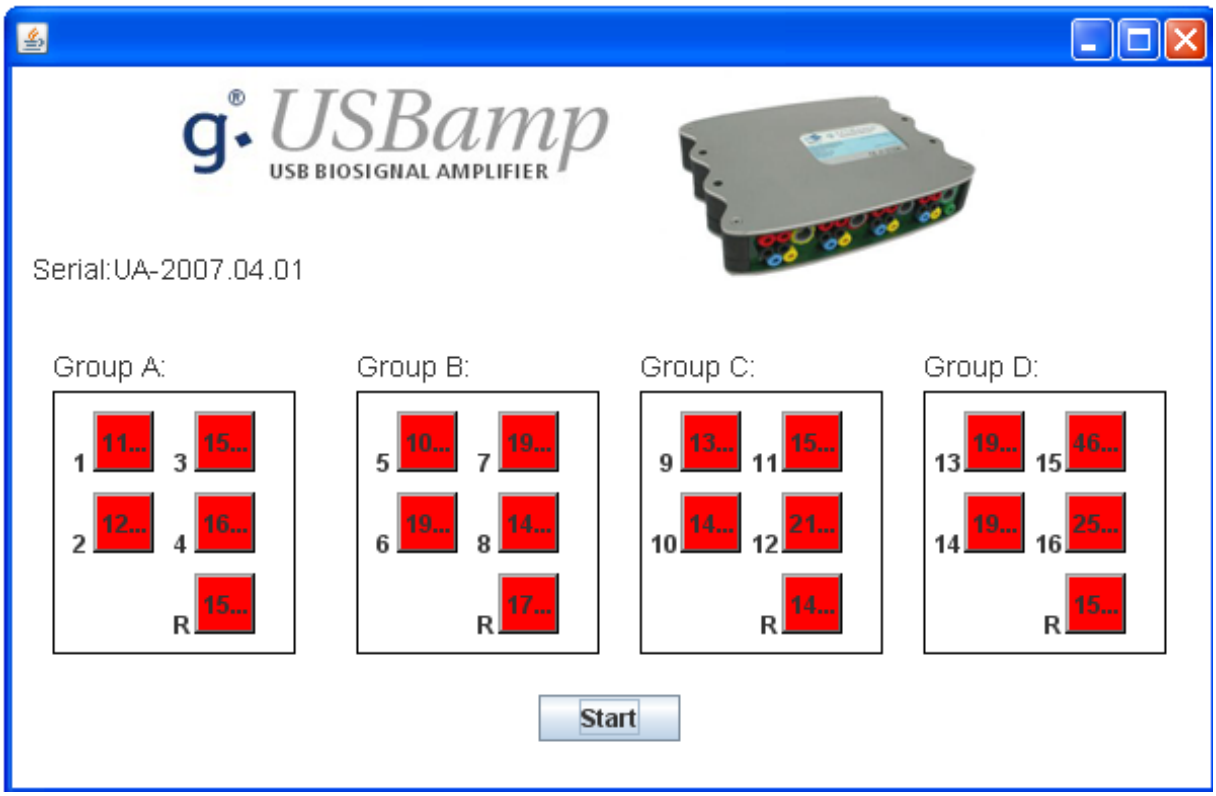
Use this tool to configure all settings the amplifier should use during recording. The set of filters has been implemented by the amplifiers manufacturer. *The cannot be changed by the customer*.

Calibration



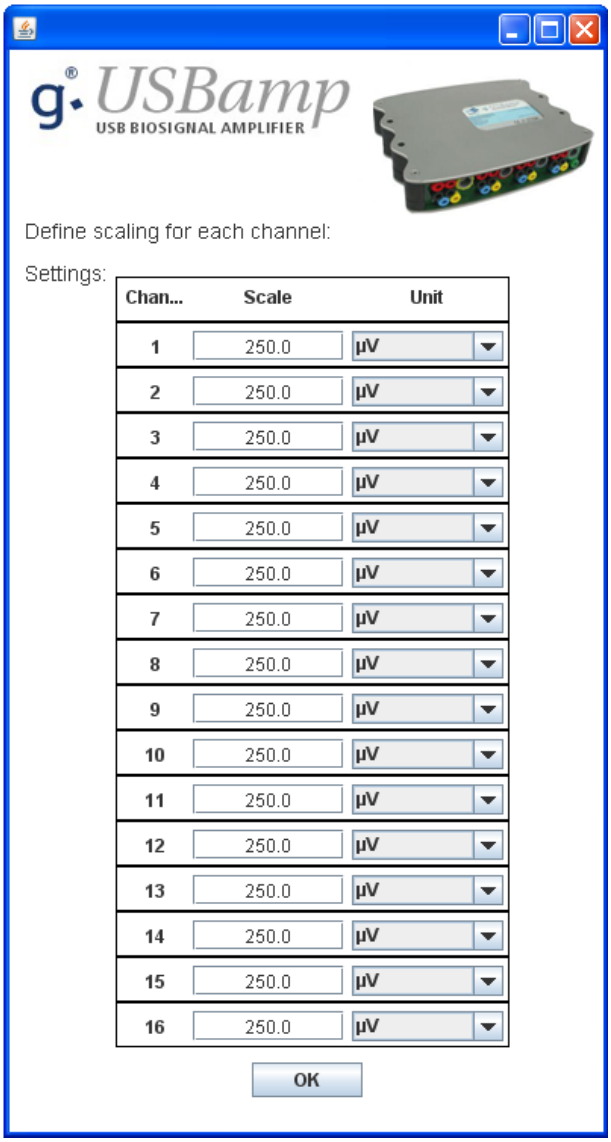
Use the calibration tool to ensure a zero-mean signal. The GtecCaptureDevice also re-calibrates the amplifier before capturing. A scaling ensures that the original signal's amplitude is not changed.

Impedance Measurement



Use this tool to check the electrode's impedances. Green indicates very good, yellow indicates good and red indicates rechecking the electrode montage. Keep in mind that impedances usually improve over time.

Scaling



Use this tool to configure the scaling of each channel

JMFEL Framework Design

The main design goal for the JMFEL was to greatly simplify the usage of the JMF. Hence, there are a few core classes – simple building blocks like players, processors, codecs renderers and data sinks – making it possible to quickly build sophisticated multimedia applications without struggling with complicated details.

The design concept is shown by simple JMFEL applications. They are simple, because they are all made entirely from existing and reusable JMFEL components. The UML diagrams presented below show the classes used in the examples. They also show the class hierarchy to demonstrate how little the end user has to develop and how much has already been done.

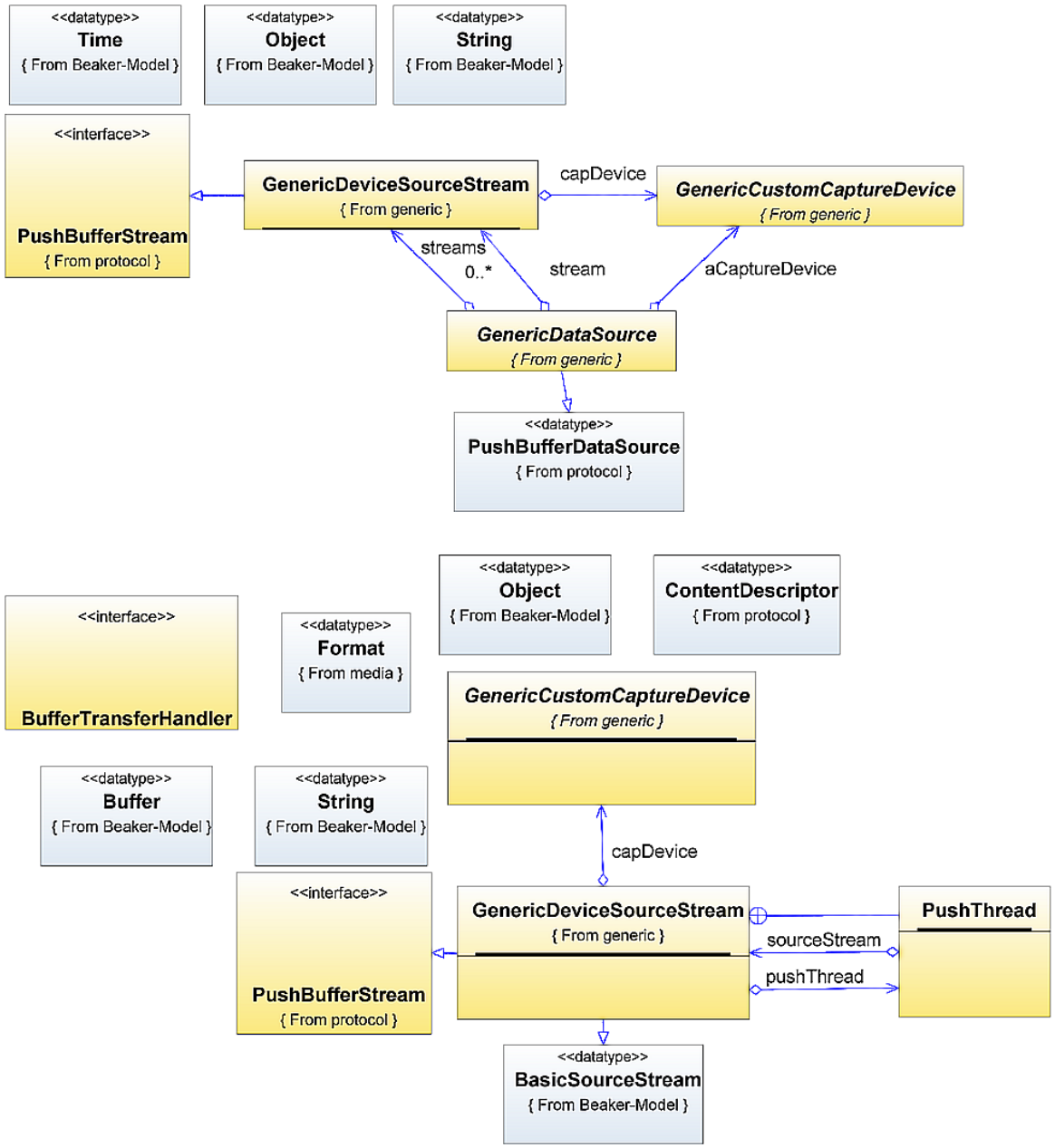
EEG capture application

The EEG signal is acquired by an EEG signal data source, which is cloned once; one data stream is fed to a processor player and the other one to an SQL database custom data sink. The data stream fed to the custom data sink is unaltered while the second stream is processed by the *EEGMotorCortexLaplacePlugin* and the *EEGDataPlugin*. The EEG signals are displayed by the *SingleEEGChannelContainer* application, which gets the data from the *EEGDataPlugin.* Normally, the processor player would send the EEG signal to the speakers, thus a *NullAudioDevice* is used to keep this from happening. Alternatively, the player could be muted.

Generic custom data source

The JMF provides DMA-like I/O – push data sources in JMF terminology – to access data from devices. Push data sources are *pushing* the data though the signal graph while pull data sources continuously poll a device for new data. Pull data sources have to be polled for new data by the player, while push data sources tell the player that new data is available. A video camera is a good example for a push data source and a video file is a good example for a pull data source.

The JMF does not directly support EEG devices, thus a custom EEG push data source has to be implemented. Unfortunately, this is a complicated task and requires solid experience in parallel programming. Hence, the JMFEL provides a ready to use implementation as a set of abstract generic classes.



All JMF data sources / software devices consist of a *DataSource* and *SourceStream*. The JMF supports *Pull Data Sources* and *Push Data Sources*. Pull data sources implement polling and push data sources implement DMA like I/O. Implementing custom push data sources is a complicated task and requires solid experience in parallel programming.

The JMFEL provides abstract push data sources to speed up the development of new custom push data sources. Right now, there is no additional support for pull data sources.

The abstract classes *GenericDataSource* and *GenericDeviceSourceStream* implement the DMA like I/O. The abstract class *GenericCustomCaptureDevice* is used to access a physical or software data source.

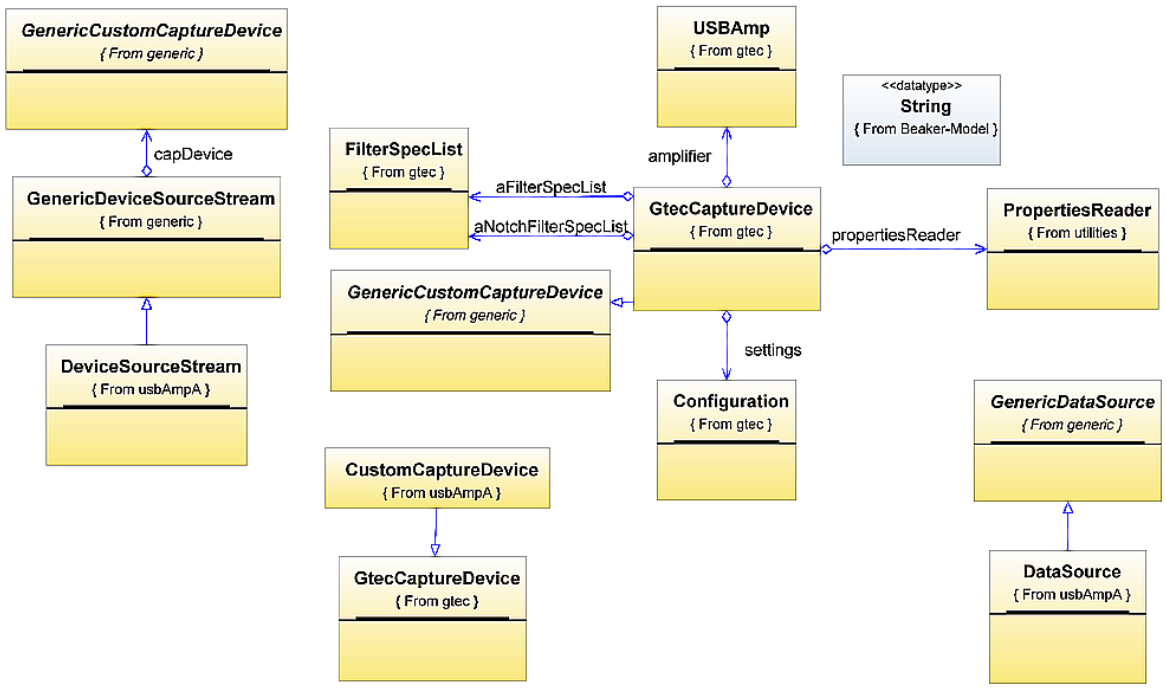
Note: *Beaker is the project's internal code name. It's also the assistant of Dr. Bunsen Honeydew. Both are characters from The Muppet Show, copyright © The Jim Henson Company.*

EEG custom data source

A custom EEG data source is implemented by sub classing the generic custom data source classes.

The only thing left to do is getting the EEG data from the physical device. This is done in the *getAvailableData()* method of the *GtecCaptureDevice* class, which uses the *USBAmp* class from the *g-tec* Java API – a JNI wrapper for the g-tec C++ API – to get access to the g-tec USBAmp EEG device.

A JMFEL protocol push data source requires the three classes *DataSource*, *SourceStream* and *CustomCaptureDevice.* Hence for the EEG push data source *DataSource extends GenericDataSource, SourceStream extends GenericDeviceSourceStream* and *CustomCaptureDevice extends GtecCaptureDevice,* which is a *GenericCustomCaptureDevice.* The two classes *DataSource*, *SourceStream* are required by the JMF and are instantiated by javax.media.Manager. createDataSource(), which is nothing else but a factory method for creating JMF data sources.



Every concrete JMF protocol data source has to implement two classes called *DeviceSourceStream* and *DataSource*. For JMFEL related implementations this simply means extending the respective abstract classes *GenericDataSource* and *GenericDeviceSourceStream*.

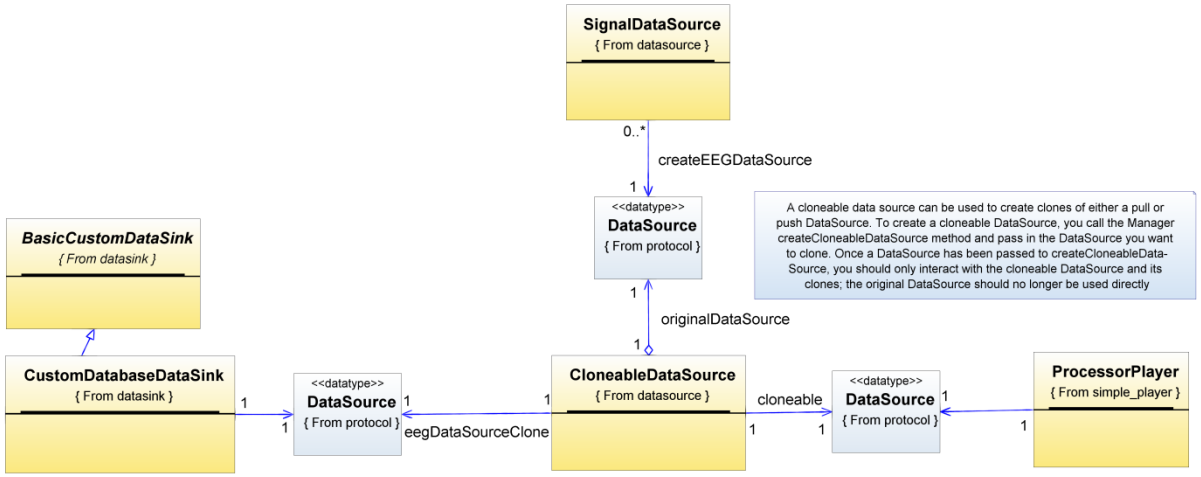
The actual work – getting the data from the device – is done in the *CustomCaptureDevice* classes' *getAvailableData()* method. Consequently, the CustomCaptureDevice extends the *GtecCaptureDevice* which extends the *GenericCustomCaptureDevice*. Within the *getAvailableData()* method the *GtecCaptureDevice* uses the *USBAmp* class from the *g-tec* Java API – a JNI wrapper for the g-tec C++ API – to get access to the g-tec USBAmp EEG device.

Cloning

The JMFEL facilitates the JMF feature to clone data sources, through the *ClonableDataSource* class. A cloned data source provides the same data as the original data source and can be used independently from the original source.

But, clones do not necessarily have the same properties as the cloneable data source used to create them or the original DataSource. For example, a cloneable data source created for a capture device might function as a master data source for its clones in this case, unless the cloneable data source is used, the clones will not produce any data. If you hook up both the cloneable data source and one or more clones, the clones will produce data at the same rate as the master.

(cf: Java Media Framework API guide (1999), p.18)

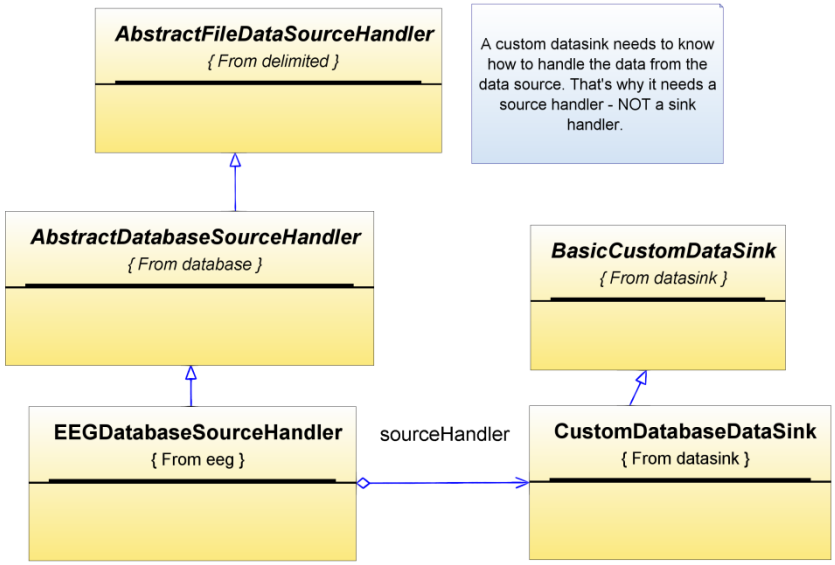


The *SignalDataSource* class uses *javax.media.Manager. createDataSource(media locator)* to instantiate the EEG data source.

The media locator for the "EEG device A" is set by *source.setMediaLocator(new MediaLocator("usbAmpA://"))* According to the JMF specification, this original data source must *not* be used after cloning. The clone is connected to the custom database data sink and the *CloneableDataSource* – the new master data source – is connected to a processor player.

Custom data sinks

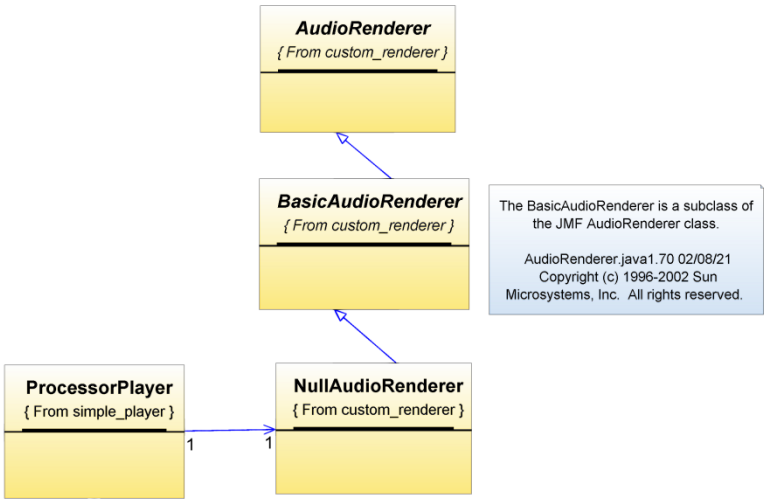
Like the standard JMF data sources, JMF data sinks "only" support common media formats. Consequently, the JMFEL provides custom data sinks for delimited files and databases. Those data sinks are generic and always need a custom source handler. It is called a source handler, *not a sink handler*, because the data sink needs to know how to handle the data from the data source. The EEG capture application uses an EEG custom database sink and an EEG data source handler. For a CSV file, the application would use an EEG custom delimited file database and an appropriate source handler. Both types of custom data sinks and source handlers – database and file related – share the same design. New file formats and new databases require a new custom source handler. *Only custom data sinks that don't match the file or database type must also subclass the BasicCustomDataSink.*



The cloned EEG data source stream is handled by the *CustomDataBaseSink*. This data sink by itself does not know how to handle the data from the data source. Hence it needs a source handler, which knows how to handle EEG data and how and where to store it in an SQL database. A similar design exists for file based custom data sinks e.g. CSV file data sinks.

Custom Renderer

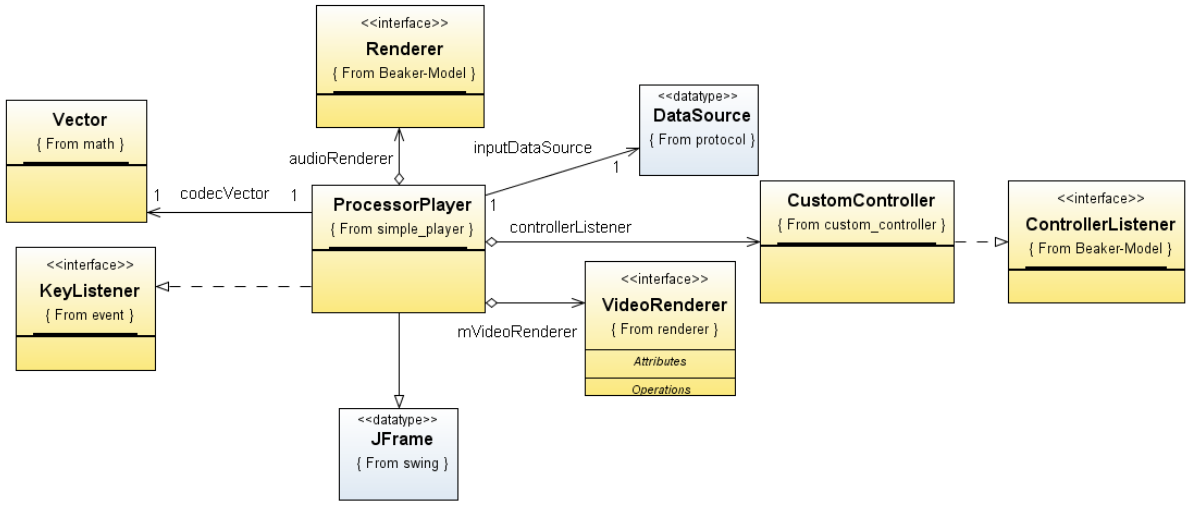
The JMF provides standard renderers for its standard media formats. Audio is rendered by the audio renderer and the audio device and videos are rendered to the screen by some audio renderer and the graphics card. Since there is no direct JMF support for EEG signals, EEG signals are also rendered by the audio device, which is very cumbersome. There are two ways to get rid of noise. The first approach is to set the player or processor player to mute. The second approach is to use a custom null audio renderer combined with a null audio device instead of the standard audio renderer. Thus, no data is sent to the audio hardware and consequently no delay is introduced. Audio delays are well known and annoying problem of the standard JMF audio implementation.



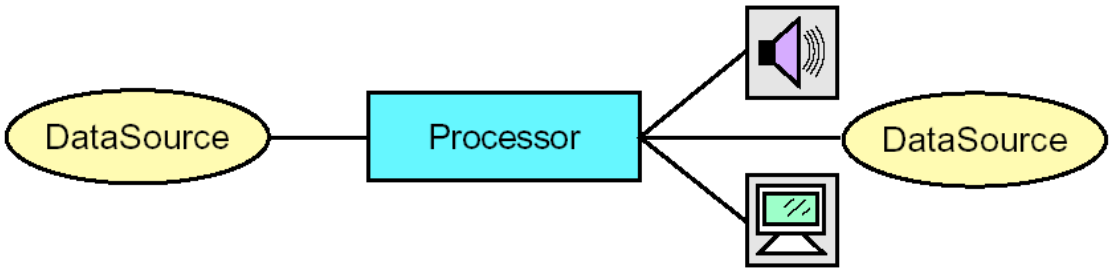
The BasicAudioRenderer enables the developer to write custom audio renderer. The *NullAudioRenderer* is probably the easiest example since it just discards any audio data. Consequently, no audio data is sent to the audio device and no delay is introduced.

ProcessorPlayer

The processor player is one of the key JMFEL building blocks. The main difference to the simple player class, which can only play media files, is that the *ProcessorPlayer* uses a JMF processor configured as a player, making it possible to use plug-ins or codecs.



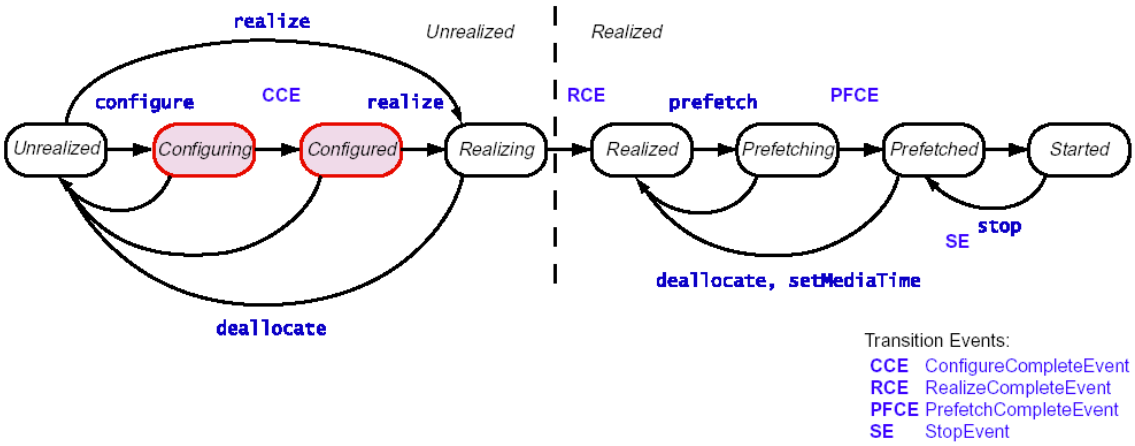
The ProcessorPlayer is an easy to use media player that supports custom renderers and codecs. It is based on the original JMF processor class and other classes, which are quite complicated to use.



Processors can also be used to present media data. A Processor is just a specialized type of Player that provides control over what processing is performed on the input media stream. A Processor supports all of the same presentation controls as a Player.

(cf: Java Media Framework API guide (1999), p.29)

Because the JMF is a *low level API*, the developer has to deal with complicated things like internal processor or player states. Even playing a simple media file with standard JMF classes is surprisingly difficult. The *ProcessorPlayer* class was one of the very first reusable classes developed for the JMFEL.



(cf: Java Media Framework API guide (1999), p.32)

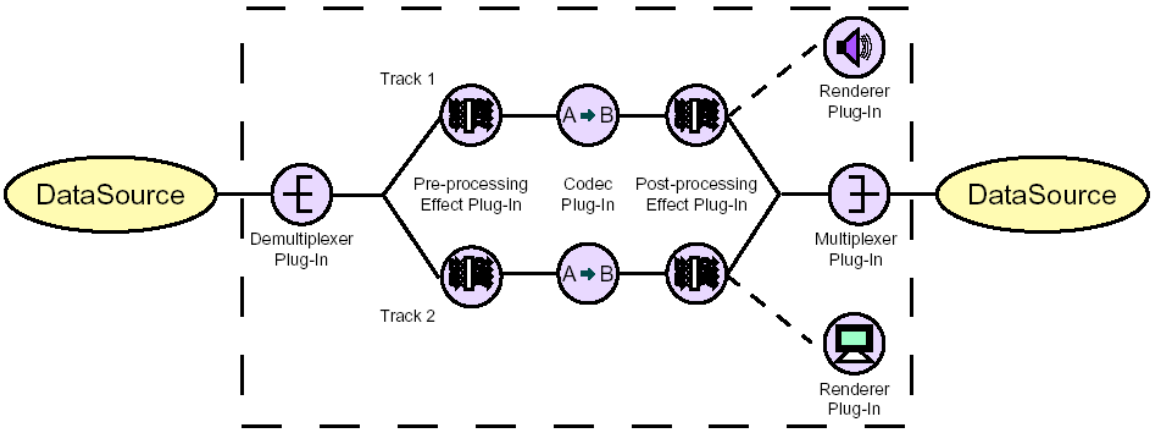
A developer trying to play media with standard player or processor JMF components has to keep these stages in mind, which makes playing a simple media stream quite complicated. The *ProcessorPlayer*, *SimpleProcessor* and *SimplePlayer* classes use the JMFEL *CustomController* class to take care of these transitions.

The *ProcessorPlayer*, its *CustomController* and the *SignalDataSource* are implemented with about 310 lines of non-trivial JMF code. Fortunately, the JMFEL code below is only 14 lines. Not only does this mean a tremendous speedup in development time, it also means a significant increase in stability.

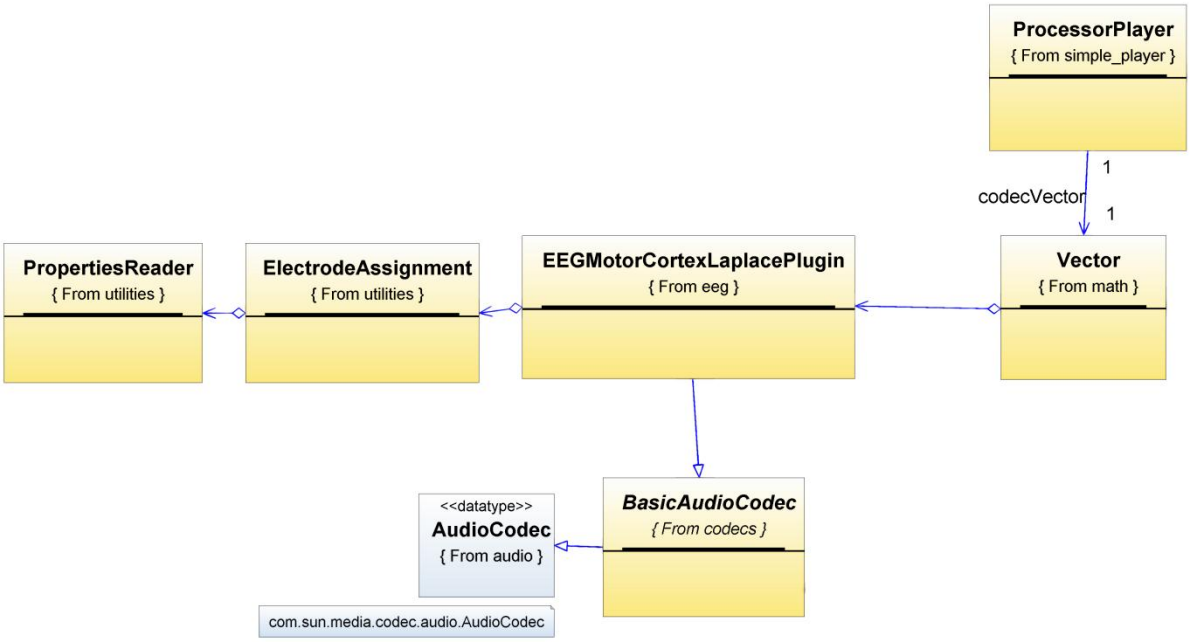
```
public static void main(String [] args) {
        String file_separator = System.getProperty("file.separator");
        DataSource dsFile  = null;
        SignalDataSource aFileDataSource;
        ProcessorPlayer aPlayer;
        aFileDataSource = new SignalDataSource();
        aFileDataSource.setMediaLocator(new
MediaLocator("file:\\"+System.getProperty("user.dir")+file_separator+"examples"+file_separator+"basic"+fil
e_separator+"audio_and_video"+file_separator+"testaudio.wav"));
        aFileDataSource.init();
        dsFile = aFileDataSource.getDataSource();
        aPlayer = new ProcessorPlayer(dsFile);
        aPlayer.setVisible(true);
        aPlayer.start();
  }
```

Codecs

The EEG sample application uses the *EEGMotorCortexLaplacePlugin* mentioned earlier to process each media buffer. The code is very similar to the one presented in the Electrode Mapping section. Codecs are somewhere within the signal processing graph, whereas renderers are always at an endpoint of the signal processing graph.

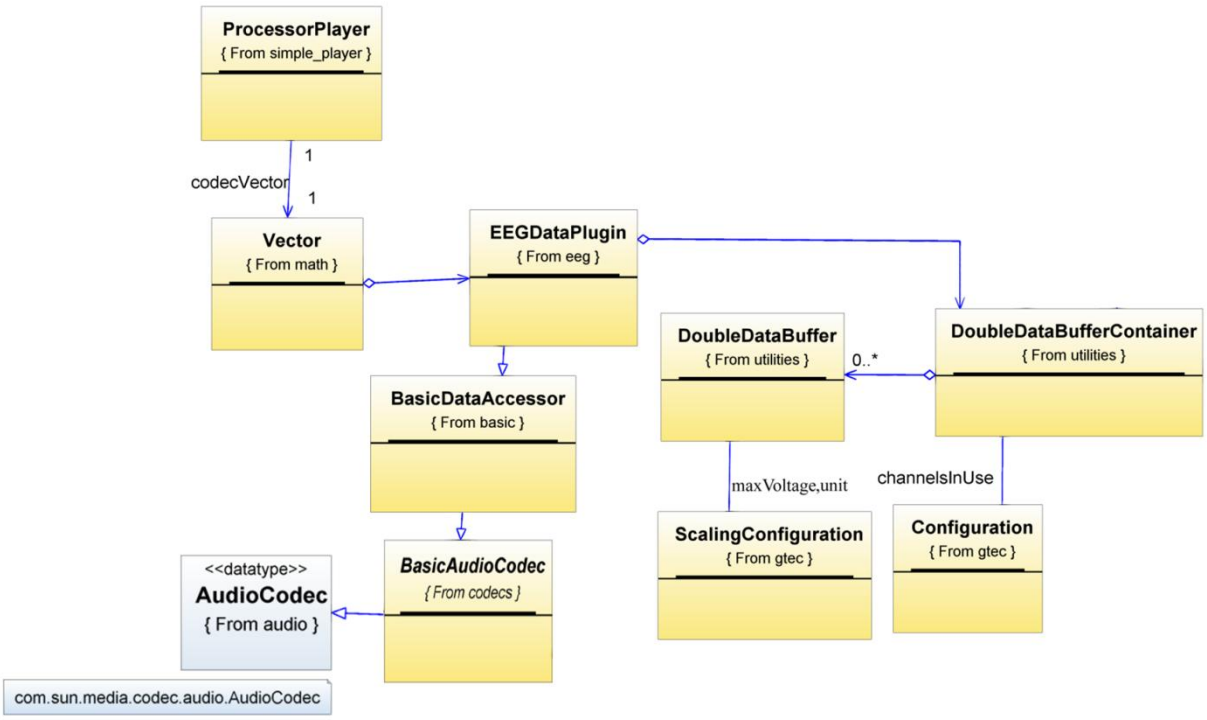


The processing of media is split into several stages. (cf: Java Media Framework API guide (1999), p.32)
There seems to be a bug for audio plug-ins in the JMF implementation. The JMFEL always uses codecs for audio processing.



The *EEGMotorCortexLaplacePlugin* is derived from the BasicAudioCodec class, which is a sub class of the com.sun.media.codec.audio.AudioCode class.

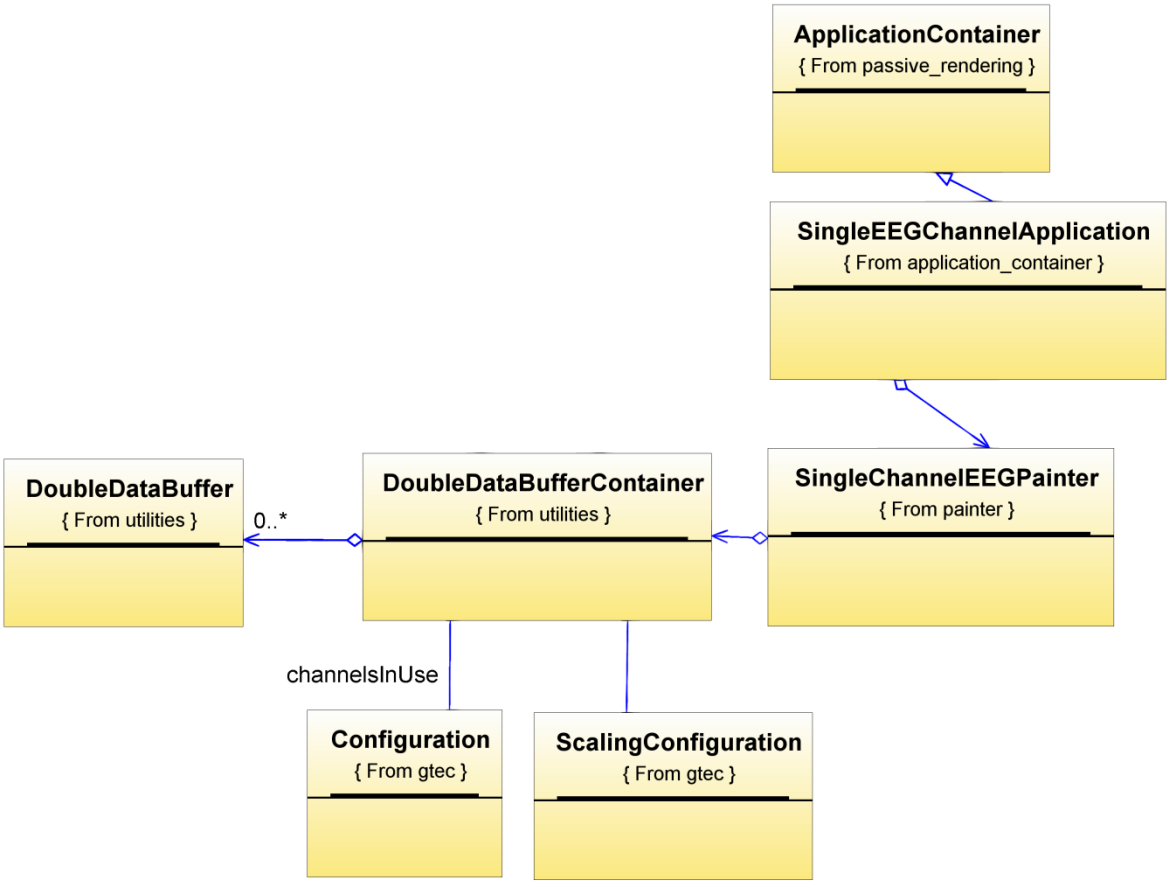
The *EEGDataPlugin* is a *BasicDataAccessor* and the second plug-in used by the EEG capture application. It de-interleaves the EEG signal and writes each channel's signal in a separate ring buffer – the *DoubleDataBuffer*, which also contains the channel's unit and scaling. The *DoubleDataBufferContainer*, which contains all buffers, is configured with the number of channels in use by the *Configuration* from the *g-tec* Java API during initialization. The *DoubleDataBufferContainer* also connects the single channel GUI application's SingleChannelEEGPainter with the JMF part.



The *ProcessorPlayer's EEGDataPlugin* is used to access each channel's EEG signal. There is one *DoubleDataBuffer* for each EEG signal. All *DoubleDataBuffer* ring buffer objects are stored in one *DoubleDataBufferContainer*, which provides individual access to each EEG channel for regular java applications. The number of channels and therefore the number of *DoubleDataBuffer's* is determined from the amplifier's *Configuration*. Each *DoubleDataBuffer* gets its scaling and unit from the amplifier's *ScalingConfiguration*.

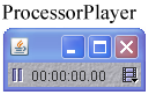
Presentation

The captured signal is drawn on the screen by the *SingleEEGChannelApplication*, which is made entirely out of reusable components.

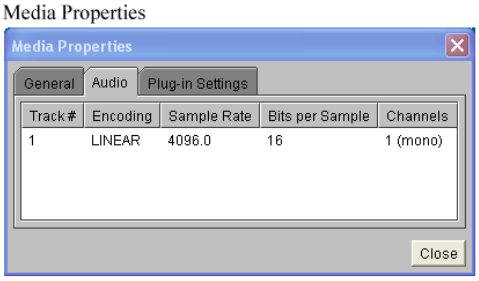


The *SingleChannelEEGPainter* connects the *SingleEEGChannelApplication* GUI with the *DoubleDataBufferContainer*, which contains the ring buffers for all channels.

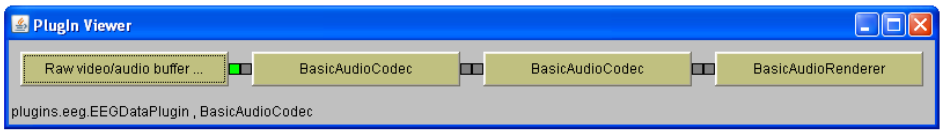
ProcessorPlayer



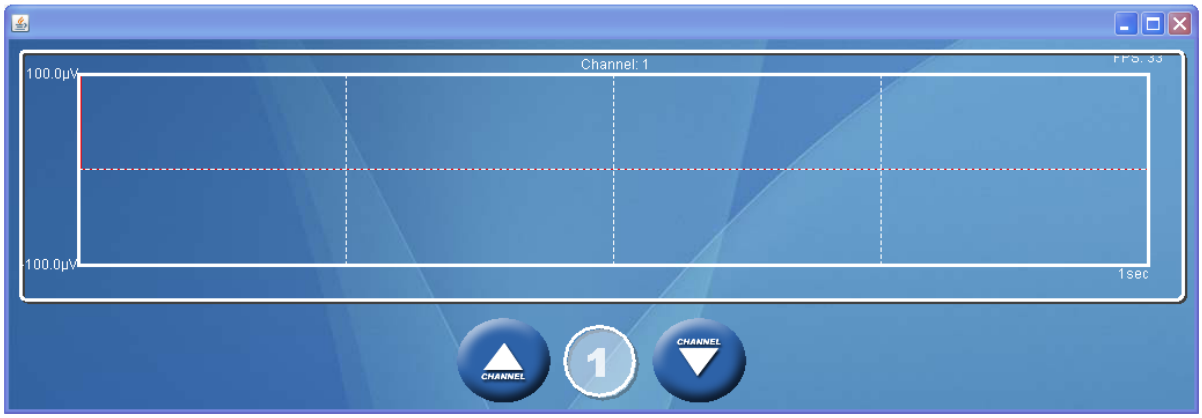
Media Properties



PlugIn Viewer



SingleEEGChannelApplication

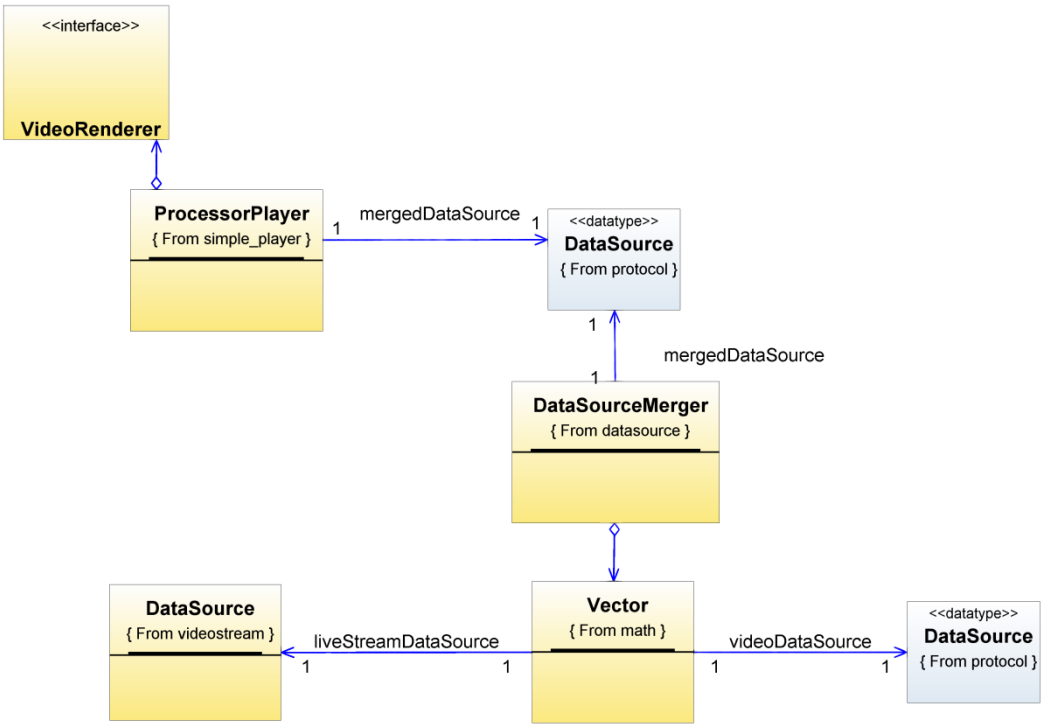


This is what the EEG capture application looks like. The *ProcessorPlayer* has only a few controls, start/pause control, a media counter and properties control. The JMF Media Properties window shows basic information about the captured signal. There are 16 channels being captured at 256Hz. All channels are interleaved into one mono channel; hence the sample rate is 4096Hz. The PlugIn Viewer is a standard part of every the JMF player. The single channel display is written as described in the Graphics Framework section and is made entirely out of reusable components. The display can show one second of a continuous signal at a time.

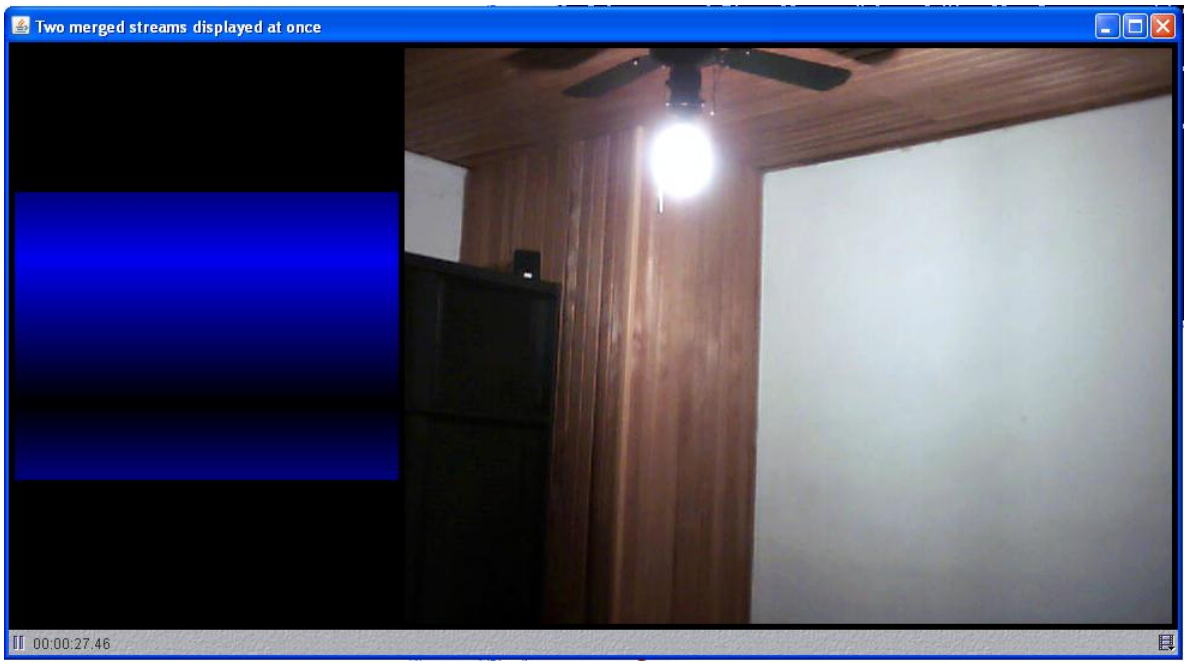
The background image is adapted from Apple's OS X Aqua user interface and can be changed by the user. OS X and Aqua are registered trademarks of Apple Inc.

Merging Application

In this example two video data sources are merged into one *MergedDataSource*, which is then played by a *ProcessorPlayer*.



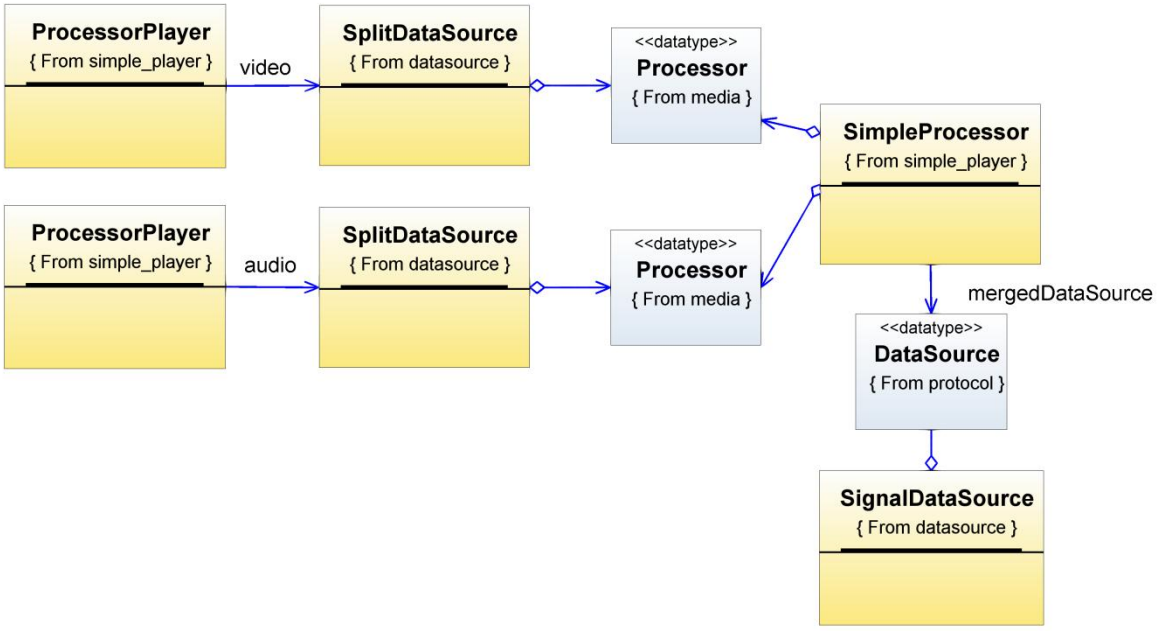
The *LiveStreamDataSource* (cf: http://java.sun.com/products/java-media/jmf/2.1.1/solutions/LiveData.html) and the *VideoDataSource* – a live video of a webcam – are merged by the *DataSourceMerger*. The result is a *MergedDataSource* which is played by the *ProcessorPlayer* and rendered by the standard *VideoRenderer*.



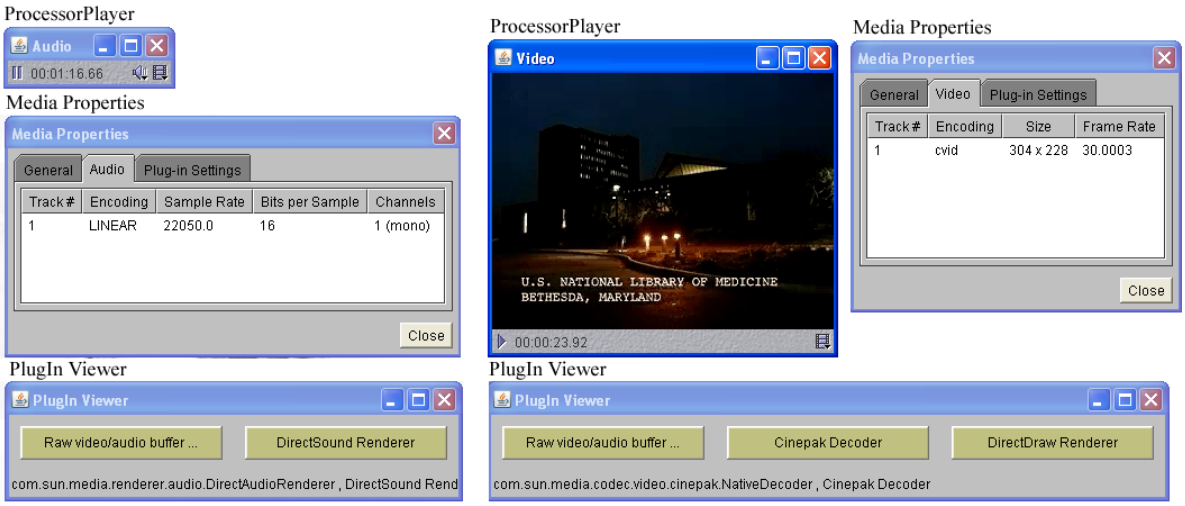
Two merged video streams. The left stream is generated by the *LiveStreamDataSource*; the right stream is captured from a regular webcam. Note that the streams do not have to have the same size or format.

Splitting Application

The *SplitDataSource* performs the converse operation to merging and therefore facilitates access to individual streams of a *MergedDataSource*.



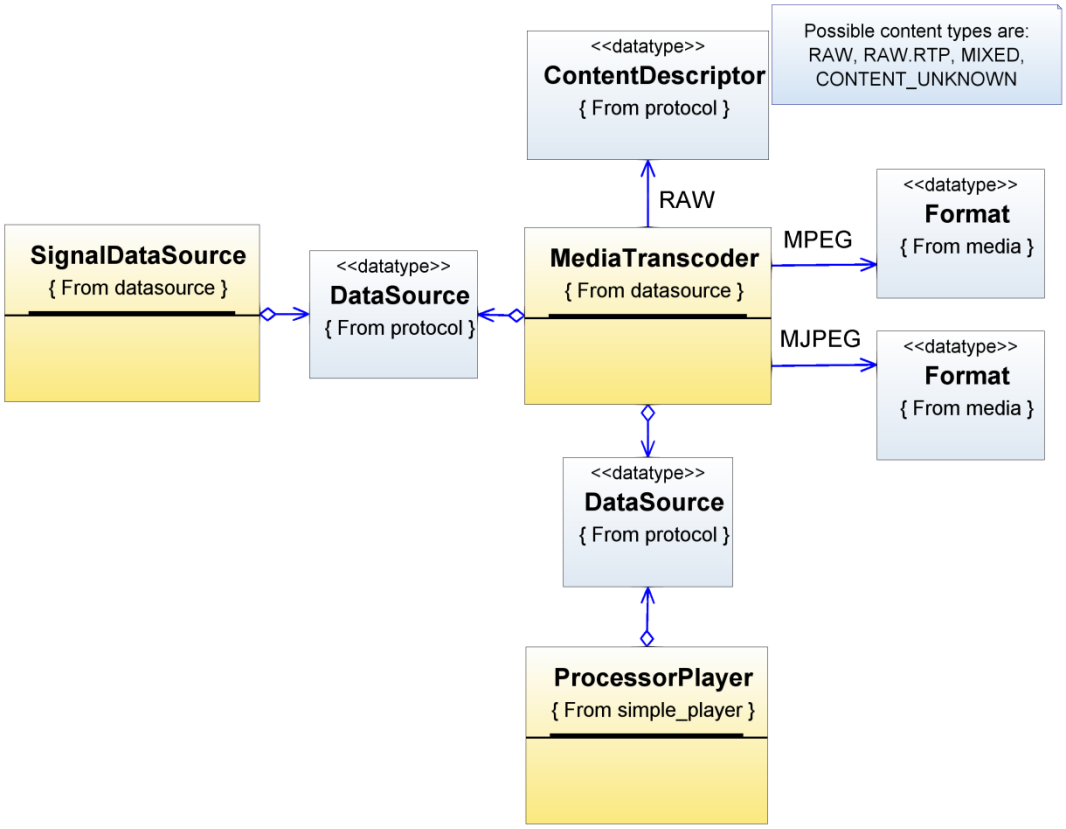
The *SignalDataSource* has a *MergedDataSource*, which contains a video and an audio track. The *SimpleProcessor* doesn't apply any codecs but generates a *javax.media.Processor*, which is needed by the *SplitDataSource* to gain access to individual media streams – the video and the audio.



The video is split into its audio and video track. Note that the video and audio players can be at different media positions, because each player has random access to its individual stream. The video "Frankenstein: Discovering the Secrets of Nature" was produced by the National Library of Medicine (cf: NLM, 1999).
(cf: NLM, http://www.nlm.nih.gov/archive/20040422/pubs/nlmnews/octdec99/od99_frankenstein.html)

Media Transcoding

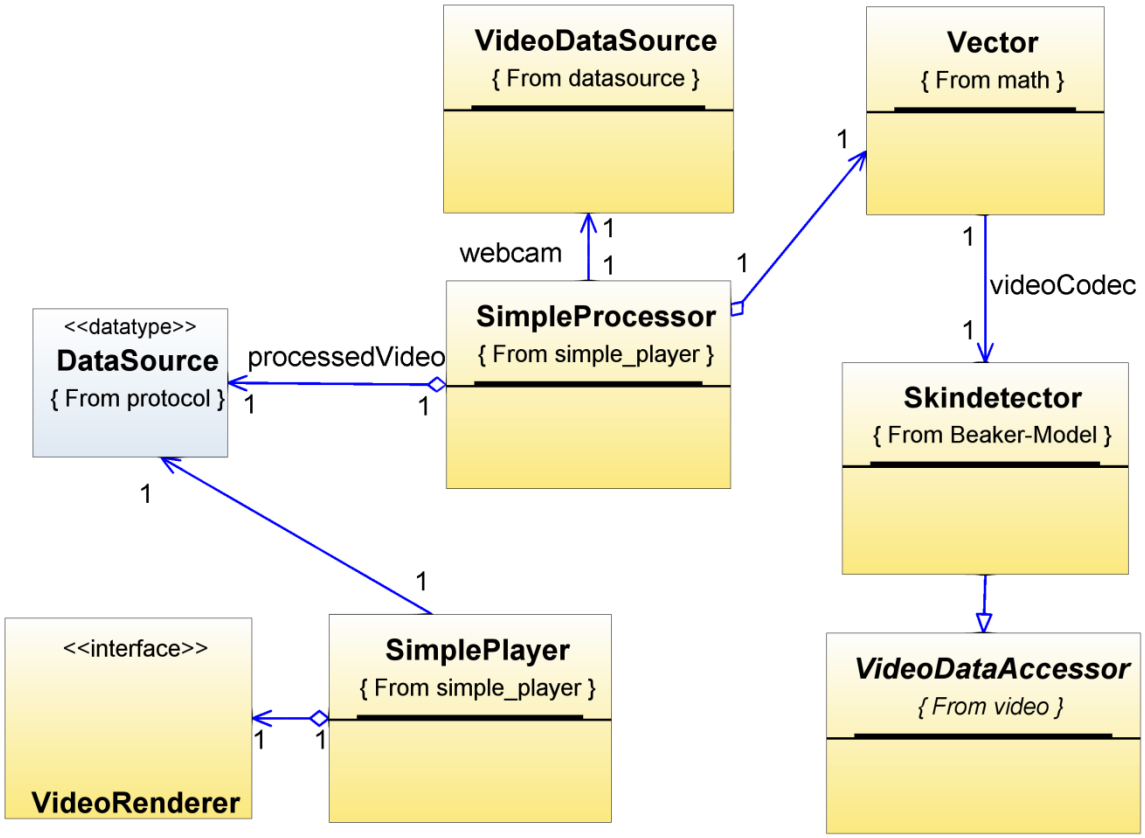
Though media transcoding is not particularly difficult with the JMF. The JMF code normally written is hard to reuse. Thus, the *MediaTranscoder* implements this code and forms a block, which has an input and an output *DataSource*. The transcoding is performed by standard JMF codecs.



The *MediaTranscoder* is a building block with an input and output *DataSource* making transcoding extremely easy.

Video Processing

Processing consecutive video frames is done in the same way as for audio frames. Prior examples always used the *ProcessorPlayer* to use codecs and plugins. This small example shows how to process a media stream with a *SimpleProcessor* before presenting it with a *SimplePlayer*.

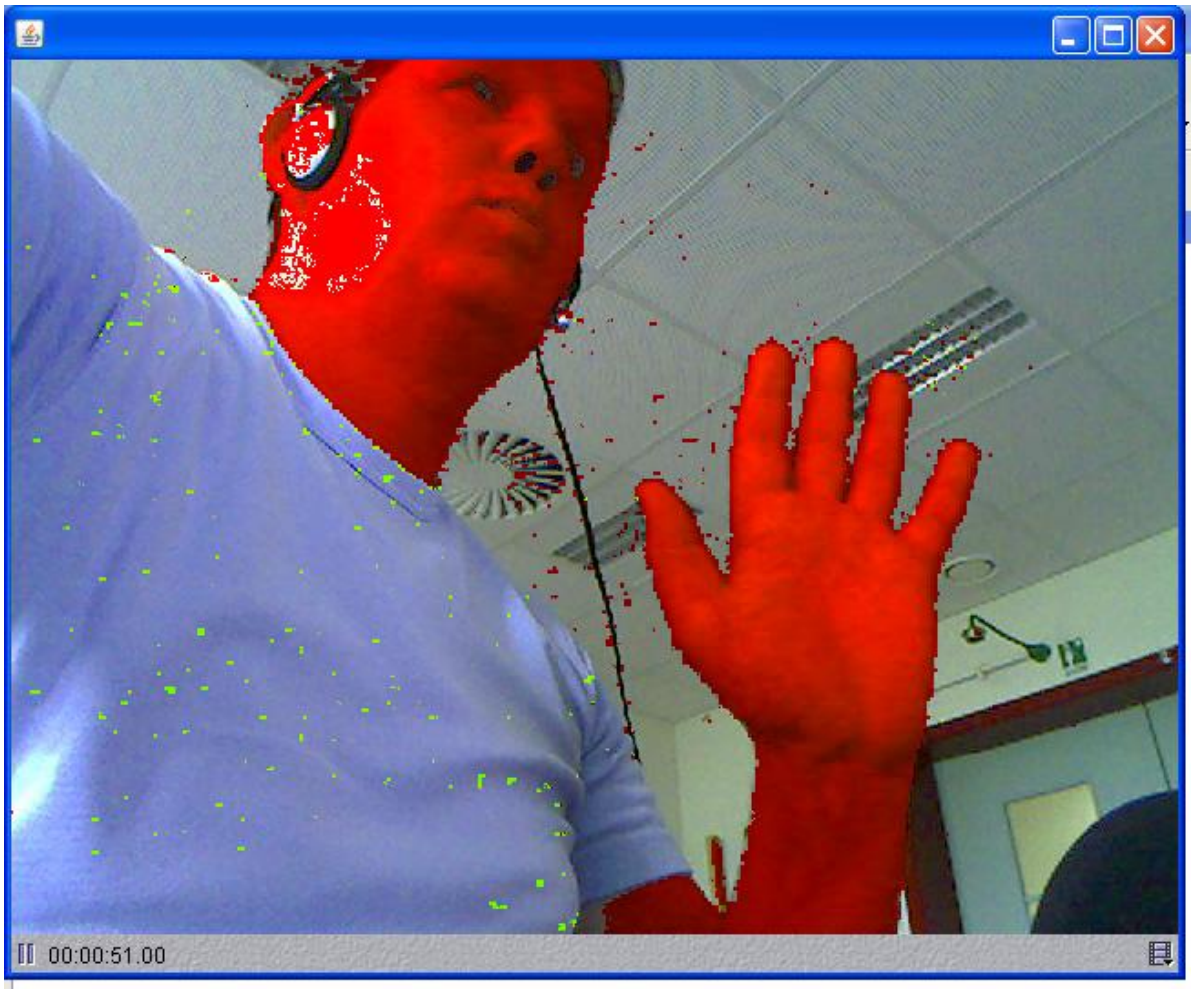


The *SimplePlayer* does not use codecs or plugins. Thus, a *SimpleProcessor* is used to do the processing.

The simple algorithm implemented in the *Skindetector* class uses the CR component of the YCBCR colour space, which is very popular for skin detection. Since it is just a simple example, it uses simple thresholding instead of a bayes-classifier. The algorithm makes a back up of the original image in the *processInData()* method. The actual skin detection is performed in the *processOutData()* method. The algorithm changes the skin coloured pixels in the original picture to red. Hence, the red face and the red hand in the picture shown below. The thresholds work best under normal lighting conditions.

Implementation:

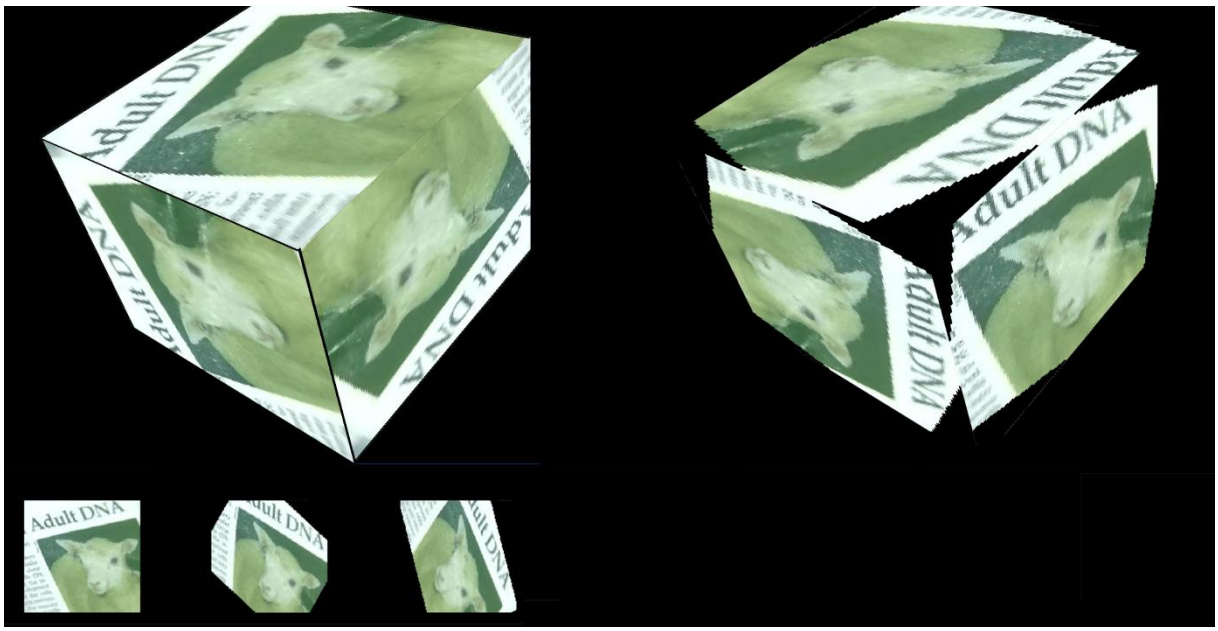
```
private ColorSpaceConverter conv  =  new ColorSpaceConverter();
//for each pixel BEGIN
 int red = getVideoImage().getPixel(xPos,yPos).getRed();
 int green = getVideoImage().getPixel(xPos,yPos).getGreen();
 int blue = getVideoImage().getPixel(xPos,yPos).getBlue();
 conv.rgb2ycbcr(red,green,blue);
if(((red> conv.getColor()[2])&&(green>red)&&(green> conv.getColor()[2]))| conv.getColor()[2]>green){
getVideoImage().setPixel(xPos,yPos,0,0,0);
}else{
 getVideoImage().setPixel(xPos,yPos,red,colorSpaceConverter.getColor()[2],0);
}
red = getVideoImage().getPixel(xPos,yPos).getRed();
green = getVideoImage().getPixel(xPos,yPos).getGreen();
blue = getVideoImage().getPixel(xPos,yPos).getBlue();
if(green>150|red<80){
 getVideoImage().setPixel(xPos,yPos,0,0,0);
} if(red>=80){
 tmp_Image.setPixel(xPos,yPos,red,green,blue);
 }
//for each pixel END
```



Processing video frames with the JMFEL is extremely easy. The *ByteImage* and the *VideoDataAccessor* class facilitate easy and fast access to individual pixels and enable the use of standard Java2D methods for image manipulation.

Interoperability between JMF and JMFEL

JMF and JMFEL components can work together seamlessly. It is very easy to use a *ProcessorPlayer* to play a video, which is rotated by the JMF Solutions' *RotationEffect* example plug-in and rendered on a 3D cube by the JMF Solutions' J3DRenderer custom video renderer.



The video "Frankenstein: Discovering the Secrets of Nature" is rendered to a Java 3D cube by the JMF Solutions' *J3DRenderer* custom video renderer and rotated by the JMF Solutions' *RotationEffect* example plug-in. The bottom of the image shows only a single side of the cube. The video "Frankenstein: Discovering the Secrets of Nature" was produced by the National Library of Medicine (cf: NLM, 1999).

(cf: JMF Solutions, http://java.sun.com/products/java-media/jmf/2.1.1/solutions/)
(cf: NLM, http://www.nlm.nih.gov/archive/20040422/pubs/nlmnews/octdec99/od99_frankenstein.html)
Dolly The Sheep: *July 31, 1996; † February 14, 2003

Experiments

The JMFEL (Java Media Framework Extension Layer) was successfully used in small and larger real world applications. The largest and certainly most complex application was the AdRacer Project, where the effect of in-game advertising with the AdRacer driving simulator was tested, while simultaneously recording from two different data sources – the eye tracker and EEG. The AdRacer Driving Simulator software is a separate application.

The smaller experiments' purpose is to show the applicability to various bio signal processing and acquisition situations.

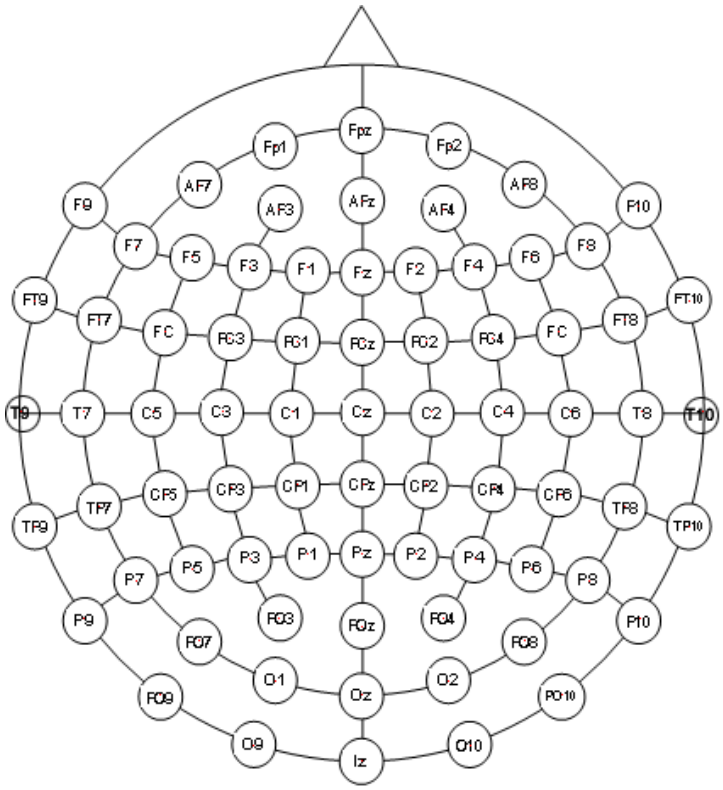
All small applications presented rely on JMF's most precise synchronization method (cf: Synchronization section). That is, recording applications always use the *RawSyncBufferMux* (cf: Synchronization chapter) to synchronize the signals. Any possible lags have been eliminated by the *NullAudioRenderer* and a *ReaNulllAudioDevice*.

Sleep EEG Experiment

Methodology

In this experiment the test subject was wired up for a 16 channel recording. One channel was used to measure the ECG (pulse); eleven channels were used for an EEG recording; the last four channels were used for an EOG recording.

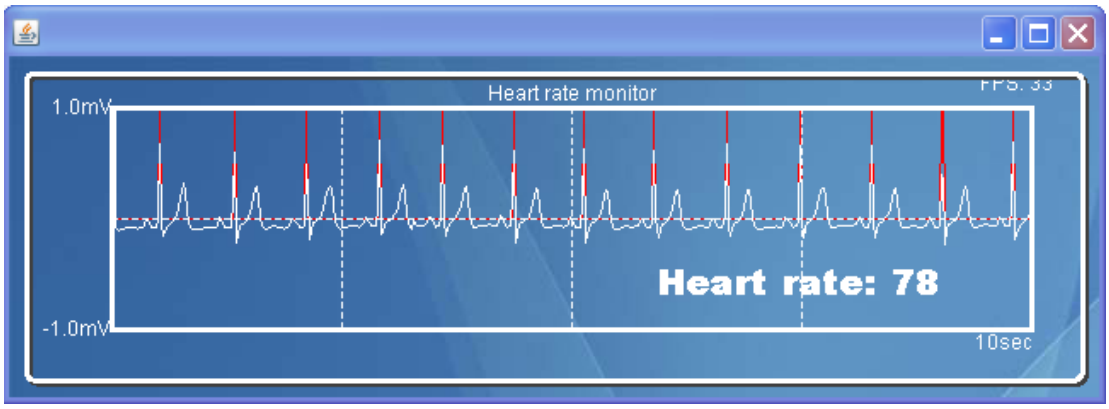
The electrode positions used in the recording are shown below. The recording software simultaneously recorded a video, an EEG, an ECG and an EOG for 1:24:44 hours. The camera image A, the EEG, ECG and the EOG signals shown below are all sampled at the same time. All screenshots are produced from data files that have been recorded earlier by a simple recording application, which is very similar to the EEG capture application presented in the JMFEL Framework Design chapter. Offline and live data sources are handled in the very same way. Please note that push and pull data sources cannot be mixed.



Channel	Electrode
1	ECG
2	Fp1
3	Fp2
4	Fpz
5	C3
6	C4
7	O1
8	O2
9	Oz
10	F3
11	F4
12	Fz
13	EOG left
14	EOG bottom left
15	EOG right
16	EOG bottom right
REF	Fcz
AFz	GND

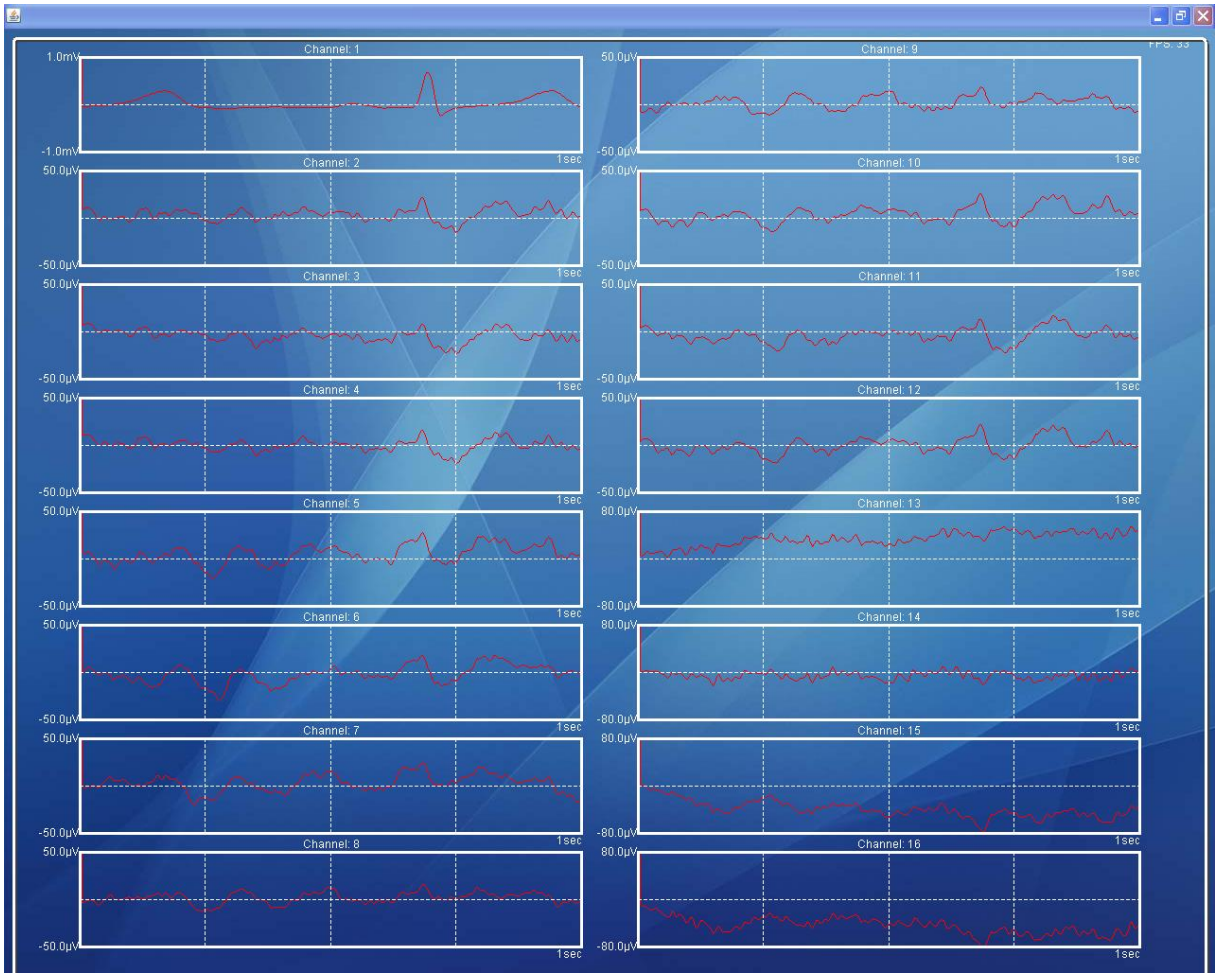
Image A shows the test subject lying on a mattress in the lab. Image B shows the subject at the end of the recording session. Note the EOG electrodes around the eyes.

The ECG signal was recorded at the subject's wrist - a common place used in clinical EEG recordings. The heart rate was determined by the simple adaptive thresholding algorithm presented earlier.



The *ECGApplication* panel uses a simple adaptive thresholding algorithm to calculate the heart rate. (cf: JMFEL Support Classes section). The display shows 10 seconds of the signal at a time.

The image below shows the *MultichannelEEGApplication* displaying all 16 channels at once.



Left column: channel 1 to channel 8; right column: channel 9 to 16. Channel 1 is the ECG; channels 2 to 12 are EEG and channels 13 to 14 are EOG signals. Each channel display shows one second of the signal at a time.

Stability and Performance

This experiment showed that JMFEL applications can run for long times without any problems. Despite its simplicity, the ECG algorithm proved to be very reliable over long periods.

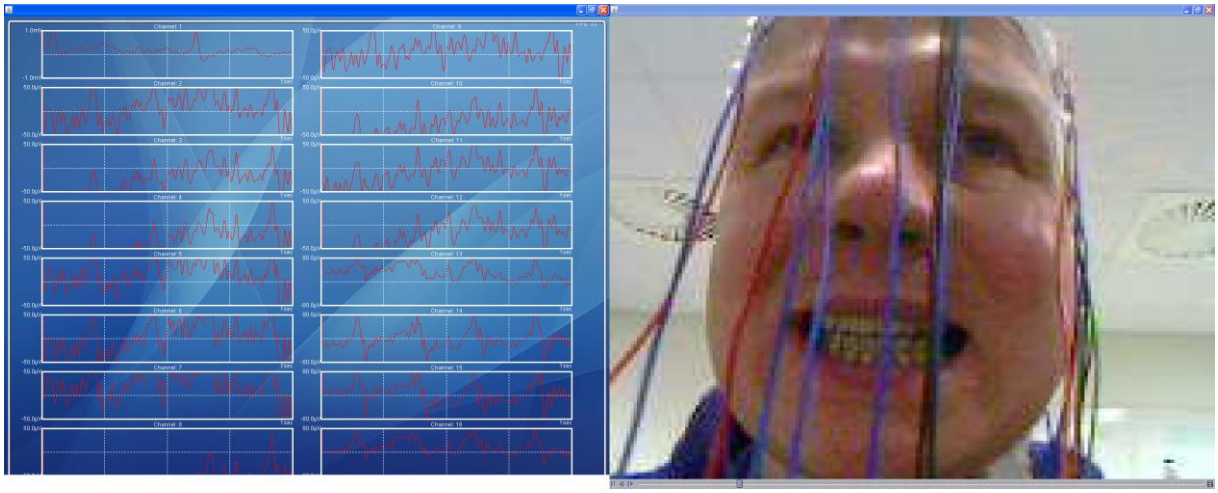
Conclusion

The JMFEL is well suited even for data acquisition applications that are supposed to run for several hours.

Synchronization experiment 1

Methodology

In this experiment the subject deliberately produced muscle artefacts. The video frames are in sync with EEG, ECG signals (cf: synchronization chapter).



The subject periodically grinds his teeth. The video is in sync with the muscle artefacts (cf: synchronization chapter). The display shows one second of the signal at a time.

Stability and Performance

The experiment took only 25 seconds. The performance and stability was as good as in the sleep EEG experiment.

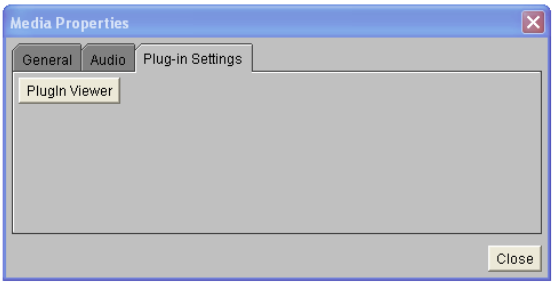
Conclusion

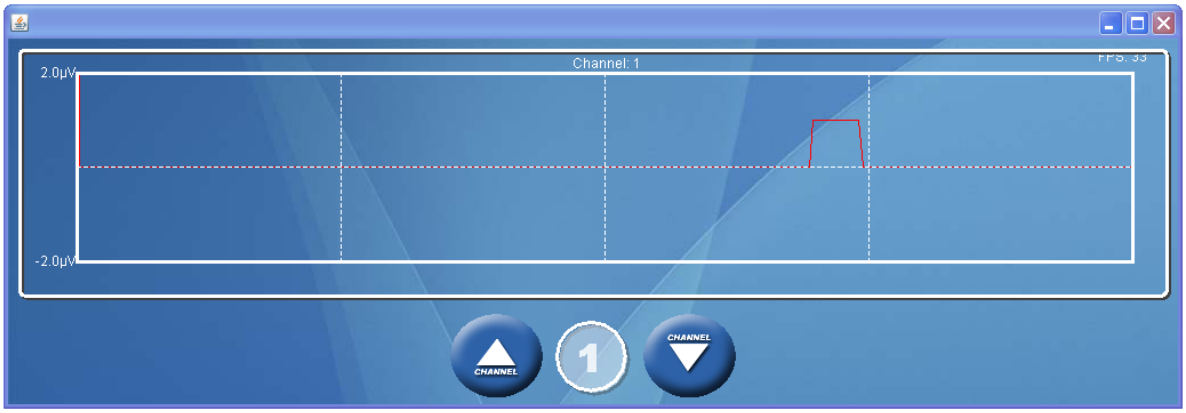
This experiment successfully shows JMFEL's ability to synchronize multiple data sources.

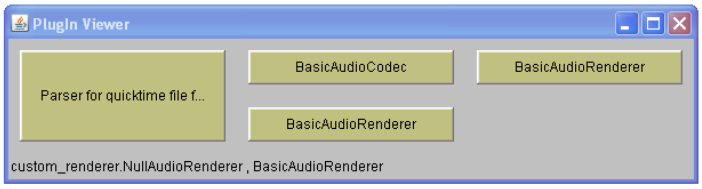
Synchronization experiment 2

Methodology

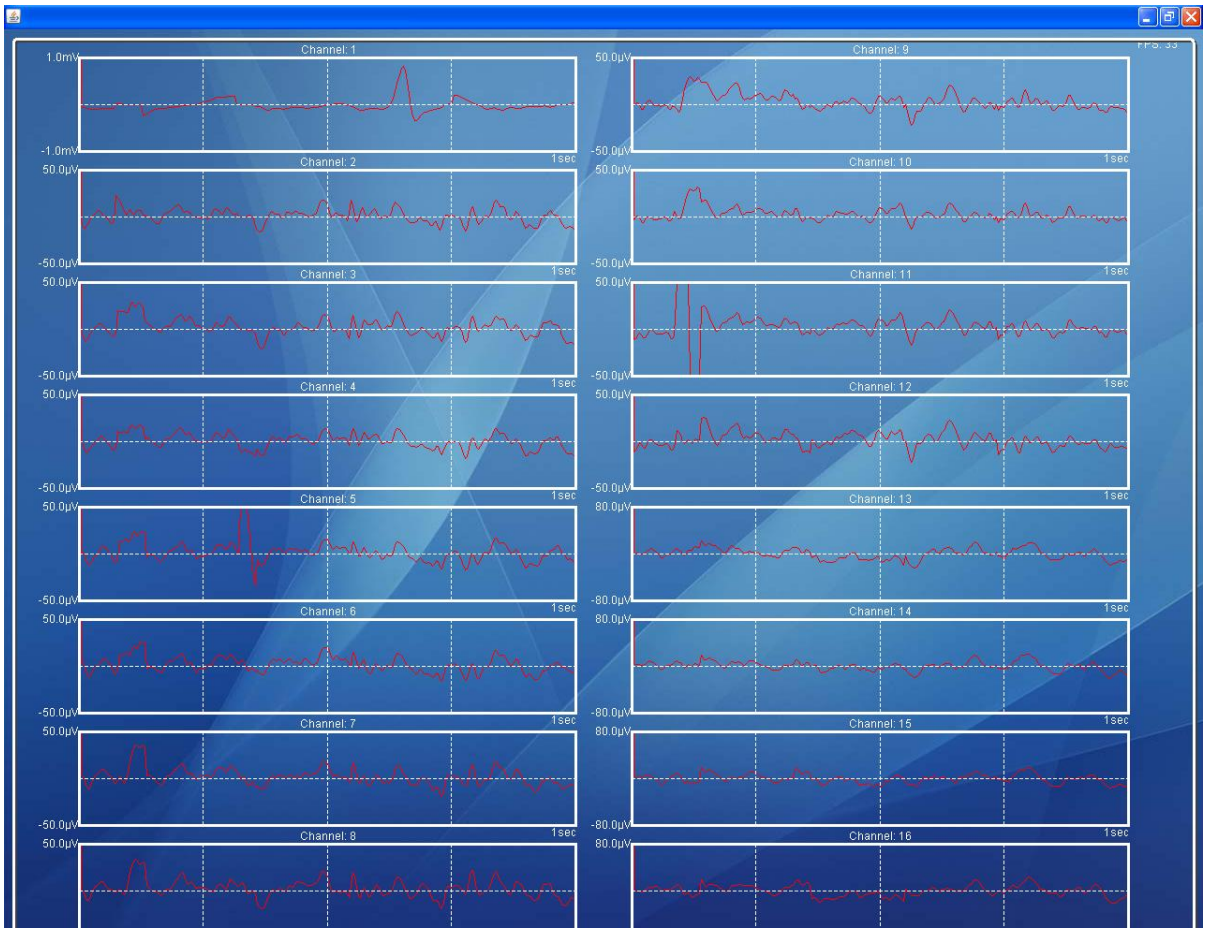
Within the first 20 minutes of this 45 minutes long experiment, the subject was asked to relax to prevent muscle artefacts and wait for the urge to press the left key with his left finger. The second half was exactly the same except that he had to use his right finger. The recording software uses two custom data sources; one EEG source and one keyboard source, which produces zeros for no activity positive values for the left key and negative values for the right key. A rising edge denotes the beginning of the key press; a falling edge denotes the end of the key press. Both data sources were merged and stored in a file. The following screenshots correspond to the exact same time.







The *ProcessorPlayer* uses a plugin to access the EEG data and a custom renderer to access the keyboard signal. Both tracks use a *NullAudioRenderer* and a *RealNullAudioDevice* to eliminate delays. A rising edges denote the point in time where the key was pressed; falling edges denote the point in time where the key was released. The display shows one second of the signal at a time.



Left column: channel 1 to channel 8; right column: channel 9 to 16. Channel 1 is the ECG; channels 2 to 12 are EEG and channels 13 to 14 are EOG signals. Channel 5 corresponds to C3 and channel 6 corresponds to C4. The display shows one second of the signal at a time.

Stability and Performance

The software showed no stability issues.

Conclusion

This experiment successfully shows JMFEL's ability to synchronize different data sources. The software can be used as a base for developing a simple Brain Computer Interface.

(cf: Libet et al. (1983): Time of conscious intention to act in relation to onset of cerebral activity (readiness potential): the unconscious initiation of a freely voluntary act. *Brain* 106, 623–642)

Brain Pong (Pong and EEG)

Methodology

Currently, we still don't have a ready to use Brain Computer Interface, so strictly speaking this is just a pong-like application which simultaneously records EEG signals and keyboard events. Nevertheless, this is an important step towards a brain controlled game.

The simple pong-like game employs the very same drawing surface as all the other JMFEL/Java2D applications demonstrated so far. That is, all components, the score, the three boundaries, the ball and the paddle are all separate panel objects, which are painted by their individual painter. Consequently, the behaviour of the ball and the collision detection are defined by separate classes too. The way how the ball moves across the screen is defined by *Ballbehaviour*, the collision detection is implemented in *CollisionDetection*.

Like the application developed for Synchronization Experiment 2, this application simultaneously records from two different data sources, the EEG and the Pong game. The *BrainPong* application extends a simple *Pong* game and replaces the standard input device – the keyboard – with a pong custom protocol push data source as the new input device, which then polls the keyboard and provides its own codec – the *SimpleGameDataPacketCodec* – to periodically encode key presses as well as paddle and ball positions into 27 out of 32 bytes. The *SimpleGameDataPacketCodec* must be used to decode the values for offline analysis. The EEG and the pong data source were merged into one *MergedDataSource* to maintain synchronization. Again, the *NullAudioRenderer* with a *RealNullAudioDevice* was used to eliminate delays. As a result, the recording application looks almost the same as in all other recording applications demonstrated so far.

Packet format:		
bytes 0 to 7:	packetID	64 bit
bytes 8 to 11:	paddlePositionX	32 bit
bytes 12 to 15:	paddlePositionY	32 bit
bytes 16 to 19:	ballPositionX	32 bit
bytes 20 to 23:	ballPositionY	32 bit
byte 24:	keyLeft	8 bit
byte 25:	keyRight	8 bit
byte 26:	keyUp	8 bit
byte 27:	keyDown	8 bit

The key board capture device generates a continuous stream of byte arrays each containing 32 bytes

A simple pong-like game. It's very easy to store EEG signals and game information into one file. This ability can be used for developing brain computer interfaces or exploring brain waves in various other situations.

Stability and Performance

The software showed no stability or performance problems.

Conclusion

This experiment demonstrates how to capture from rather exotic data sources, a game and the key board. The approach can be used as a base for developing a simple Brain Computer Interface. It demonstrates how conveniently easy such an application could be developed.

AdRacer experiment
When Items become Victims
Testing the Effects of In-Game Advertising with AdRacer

Researchers:

Dr. rer. nat. André Melzer (Dipl.-Psych.) (head researcher)	University of Lübeck, Germany
PD Dr. rer.nat. Ulrich G. Hofmann	University of Lübeck, Germany
Brad J. Bushman, Phd.	University of Michigan, Ann Arbor, MI Vrije Universiteit Amsterdam, The Netherlands

System Development:

Dipl.-Inf (FH) Olaf Christ	Acquisition System to record from the Eye Tracker and EEG. JMFEL and support classes for EEG, Eye Tracking, Signal Processing and database access.
Cand.- Inf Sebastian Otto	AdRacer Driving Simulator Terrain Generator and VR Modelling
Cand.- Inf Henning Hansen	Memory Tests
Cand.- Inf Gregor Radzimski	Database programming, offline analysis.

The goal of this *pilot study* was to investigate the impact of violence on the ability to remember brands in gaming. In television, studies seem to indicate that the ability to remember a brand is affected by violence. “If the television program *bleeds*, memory for advertisement *recedes*” (cf: B. J. Bushman and C. M. Phillips (2001, p. 44)).

There are various explanations. E.g., existing memory traces might be blocked, making retrieval impossible. And if attention is diverted to sex and violence, memory traces might not be established at all (cf: B. J. Bushman and A. Bonacci (2002, p.561)). Information might not be important enough, because people "see or hear hundreds of advertisements each day" (cf: Bushman & Bonacci, 2002). Viewers might also get aggressive, and a hostile mood is "less than optimal for remembering advertising" (cf: Gunter et al., 2005)". However, a recent study by Droulers & Roullet (2004) invalidates the main finding of Bushman & Bonacci (2002). So what was the result of our study?

Driving Simulator

1

3

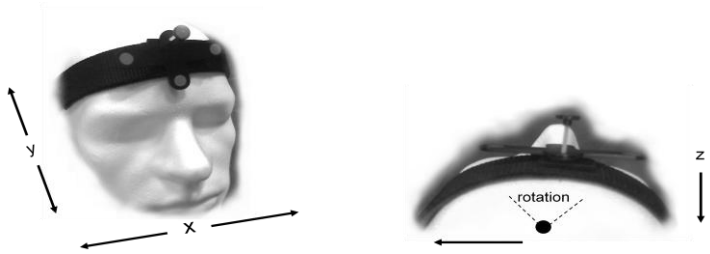
2

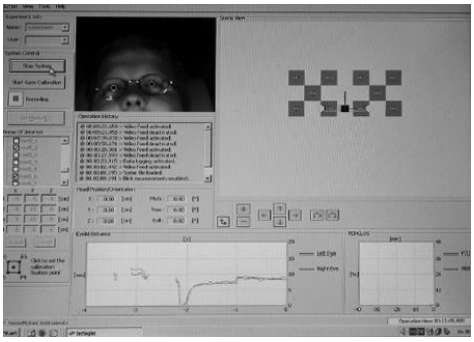
1: The AdRacer Driving Simulator
Features:
-Height adjustable seat
- rear projection screen 3DOF
- Screen width: 700mm
- Screen width: 560mm
- FOV 60° (horizontally)
- Force Feedback steering wheel
- Pedals (see figure 2)
- Stereo sound

2: The eye tracking hardware is mounted behind the rear projection screen and above the steering wheel. The projector behind the steering wheel is used for rear projection.

3: Close-up of the eye tracking hardware. The infrared-sen= sitive camera is located in the centre. The infrared lights are located on the left and right.

Eye tracker



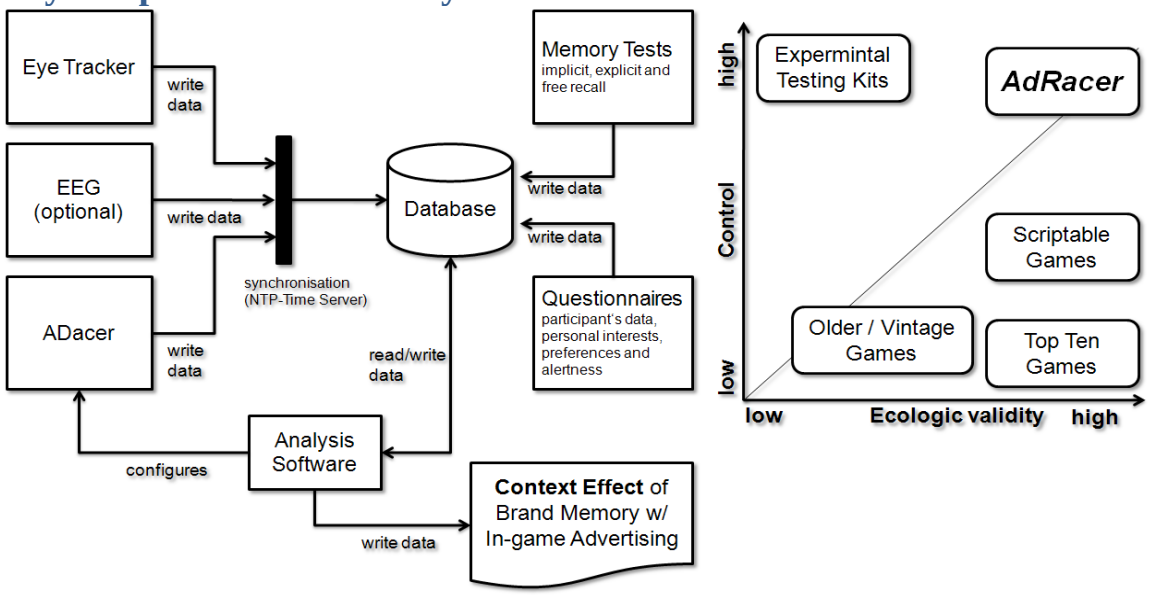


The eye tracker used in this study requires the subject to wear a head band with markers to track the head for point of regard/gaze measurement. Screenshot of the Insight Eye Tracking software by SMI showing a set of calibration points in the top right and a blink signal in the bottom

Specs for Insight Eye Tracker according to SensoMotoric Instruments GmbH:

- Sampling Rate 120 Hz for gaze or blink
- 60 Hz for gaze and blink
- Gaze accuracy (typically) 1° within a 10° forward cone
- 2° within a 20° forward cone
- 3° + beyond
- Head accuracy (6DOF) 1°–2.5° for orientation
- 0.5–2.5 mm for position
- Eyelid closure accuracy 1 mm for upper/lower eyelid
- Operating range for head distance: 45–85 cm
- Horizontal: ± 10–15 cm
- Vertical: ± 10–15 cm
- Yaw: ± 30°–45°
- Pitch: ± 30°
- Roll: ± 30°
- Operating System Windows 2000

Key components of the study



Key components and its dependencies. *Because everything can be configured, full control over the whole experiment is assured.*

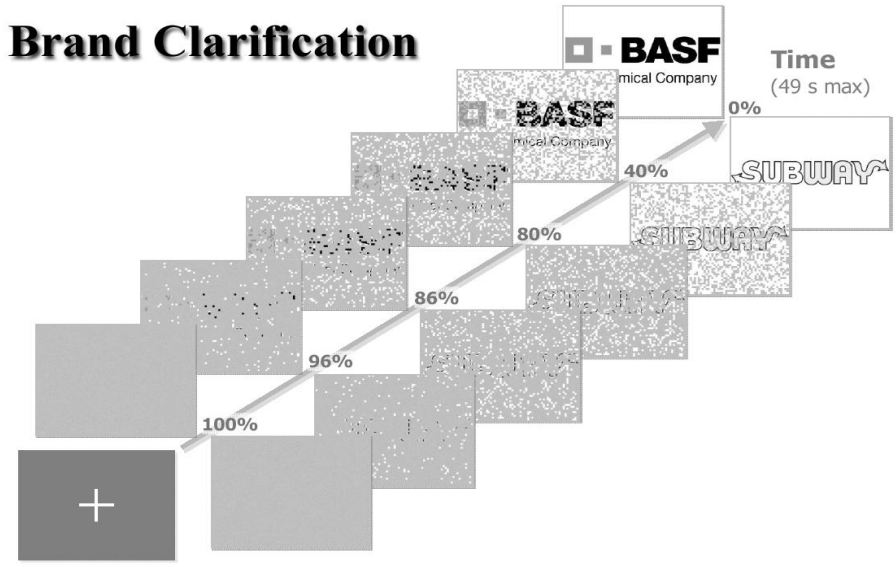


Image by André Melzer

During the implicit and explicit memory test the subject is presented logos which gradually become more and more visible over time. Once a logo is recognized, the subject presses a button and types in the name of the brand. The time it took to recognize the logos are written into the database.

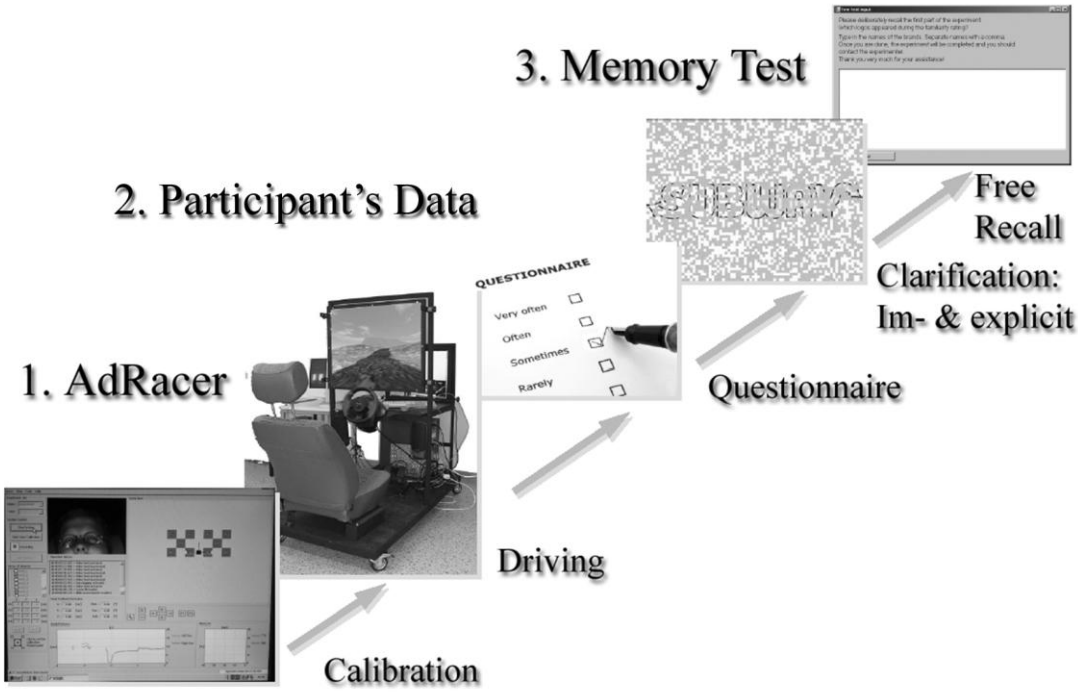


Image by André Melzer

Depending on the lot participants drew at the beginning, they had to take either the violent or non violent race. *Of course they did not know that there were two conditions or what the whole study was about*. After calibration, they were asked to pass the qualification race and then the actual race, followed by a questionnaire, the clarification and a free recall.

Methodology

Brand memory in games is a "virtually unresearched area" (Yang et al., 2006). In fact there is no study so far comparing violent and non violent games in this context. Psychologically, the key aspects are visual attention (encoding) and the retrieval performance given by explicit and implicit memory tests. The impact of game experience, attitude towards games, violence in games as well as age and gender is taken into account by a simple questionnaire. The impact of physiological aspects is taken into account by EEG, ECG and EMG measurements.

19 students participated in the pilot study; 9 students played the violent version and 10 students played the nonviolent version.

Following the calibration of the eye tracking system, the participants got used to the headband, the eye tracking and learned how to drive in the simulator during the qualification race. Every test subject was told to reach a score of at least 180000 points within 3 minutes. All subjects had three attempts to reach that score. This was only to increase the pressure; everyone was of course allowed to play the actual race game. The qualification was in fact only training – *the learning criterion*.

Stressed, well-conditioned by the qualification and completely unaware that there was no time limit from now on, almost every subject developed a lead foot and raced as fast as possible resulting in an average racing time of 12 to 15 minutes.

Following the race, everyone was told to fill out a questionnaire before taking the clarification test which they were told is to test their visual acuity. The clarification part consists of an implicit memory test followed by an explicit one and a free recall.

Results

- A treatment check on the results from the questionnaire showed a substantial difference in violence.
- Participants were well aware they played a violent game.
- Participants playing the violent version were paying *less* attention to brand logos (duration *and* number of fixations).
- Yet, they were the only ones benefiting from repetition.
- Even free recall data indicates a benefit from violence.

Eyetracking problems

We only had 19 participants. And because eye tracking is tricky and requires perfect calibration, this is a big reliability issue. The three big problems are reflections, the relatively low accuracy of the system we used and that participants were not looking exactly the way the system expects it. To minimize reflections for people wearing glasses, we used polarized infrared light, but not with great success. To detect fixations the analysis software should implement some advanced fixation detection algorithms instead of simple ad hoc method, which takes the point of regard at a certain time and an adjustable eccentricity to accept "near hits".

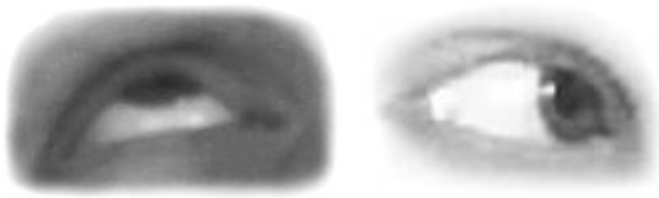
(cf: Identifying fixations and saccades in eye-tracking protocols; Dario D. Salvucci, Joseph H. Goldberg; Proceedings of the 2000 symposium on Eye tracking research & applications)

(cf: Eye Fixation Analysis Functions; http://www.eyegaze.com/doc/FixationSourcecode.htm; LC Technologies, Inc.)

(cf: Salvucci 1999; Salvucci & Anderson 2000;)

The *Insight Eye Tracking* system is not that accurate, our screen is quite large, the system only uses one camera and the participants head is barely within the range of 85 cm. Hence, accuracy significantly drops outside a 10-15° cone. We used up to 42 calibration points with each point centred in a 10x10 cm area of interest and this seemed to keep the accuracy within an acceptable level most of the time. That is, after calibration, the subject was looking at all 42 calibration points and the eye tracker was able to successfully detect almost all gaze positions correctly. We then also performed tests, where we were driving and voluntarily looking at all bill boards. Again, the offline analysis show a sufficient hit rate.

When really pushing the limits, even the way participants look becomes an issue and a big problem. We only got reasonable results if participants were looking exactly right. That is, you cannot move the eyes too much to the left or right or up and down while keeping the head stationary, as this will stop tracking the eyes entirely causing artefacts in the data. Even if you *do* move your eyes and head together you still cannot move too far, as this will again stop tracking the eyes. You cannot move the markers of the headband outside the camera's FOV, as this will again stop tracking the eyes. Consequently, you have to look *exactly* right to assure optimal results, something you cannot expect from participants.
The reason for all those technical problems is that because of the eye's anatomy and the large distance, the camera is sometimes unable to properly extract the limbus, the pupil or the purkinje images. A second camera might improve the situation - especially in the horizontal plane - but at distances of about 80-85 cm and with a screen as large as ours, the problem will always be present.



These images show the problem the tracker is having at extreme angles of vision.
The problem is particularly severe in the vertical plane (left image).

Psychological problems

Violence might significantly decrease the ability to remember brand names shown on television (Bushman & Bonacci (2002)). But according to Droulers & Roullet (2004), the opposite might also be true. For computer games, the situation is quite unclear and further research is absolutely necessary.

The context effect is thought to reflect differences in emotional arousal, not in valence. We didn't measure emotional indicators like heart rate, skin resistance or blood pressure. Instead we just asked them, and their questionnaires indicated *similar levels* of concentration, alertness and excitement.

Conclusion

The number of participants was too small. However, for a pilot study we have learned a lot. We still have to find ways to improve eye tracking results and how to take peripheral vision more into account to test whether "violence" induces holistic encoding of features. Our video game might have also altered our participant's visual selective attention (cf: C. Shawn Green & Daphne Bavelier (2003)). Consequently, we have to find the best attention model for our experiment and an appropriate fixation detection algorithm.

Applicability of the JMFEL

The application to acquire data from the Eye Tracker and the EEG, and writing the data into a database is developed using the JMFEL and its support classes, which fully support the *Insight Eye Tracker* and our EEG device. The classes also provide support for visualization and signal processing as well as manufacturer specific code, which for the Eye Tracker is the *Insight UDP-Packet* decoder class and a JNI-wrapper API for the EEG devices' C++-API. The JNI-wrapper API was used to develop the JMF specific *EEG data source*; the *Insight UDP-Packet* decoder is used to develop plug-ins, which access and decode media buffers containing eye tracking data.

The data acquisition application is made almost entirely from reusable components. Apart from code to configure and set up the required data paths (see examples in the JMFEL chapter), the only custom made code is the user interface, some button event handling and database specific data source handlers to store the data into the database. And even these handlers are just subclasses from generic abstract classes of the JMFEL.

The goal for developing the JMFEL and its support classes was to speed up the development of multimedia applications. Since reading data from two different sources and writing the data to some destination e.g. a database is nothing but a simple multimedia application, the JMFEL was well suited to write applications for this project.

Stability and Performance

The application is made almost entirely from reusable and well tested components, thus stability was never an issue. In fact, the application *never* crashed. The speed was always sufficient with a CPU usage well below critical levels. The actual acquisition consumes very little CPU time. Most of the CPU time was used by the rendering process and online signal processing of the EEG signal. The rendering performance can be significantly improved by using OpenGL. This would also lower the CPU usage.

Synchronization

For synchronizing our machines' system clocks, time stamps were generated and written to the database. Since we were synchronizing eye tracking data, EEG data and data from the *AdRacer driving simulation*, we needed timestamps that represent real dates. To get the system clocks synchronized with a reliable time source we chose an NTP time server. The time stamp value provided by the *Insight Tracker* is not a real date and represents a microsecond value taken from some fixed but arbitrary date. Under ideal conditions NTP provides microsecond accuracy. In our case we achieved a few milliseconds. Good enough, since the standard system clock only has an accuracy of about 16ms (worst case). Later offline analysis (playback) with the data base application proved this assumption to be valid. We used an NTP server on the internet. Later studies should use a dedicated NTP time server in the lab to increase reliability and accuracy or a different synchronization method.

The time stamps were generated and added to SQL insert statements just before they were written to the database. The time elapsed from reading the data from the data source to generating the SQL insert statements is not an issue. It only takes microseconds from reading the data to writing to the database. Please note that we did not use any of the synchronization methods described in the JMFEL synchronization chapter since this requires either the driving simulator to use the JMFEL or to send data directly to the data acquisition software e.g. via a UDP connection.

(cf: Bushman, B. J., & Phillips, C. M. (2001). If the television program bleeds, memory for the advertisement recedes. Current Directions in Psychological Science, 10, 44-47.)
(cf: Gunter, B., Furnham, A., & Pappa, E. (2005). Effects of television violence on memory for violent and nonviolent advertising. Journal of Applied Social Psychology, 35(8), 1680-1697.)
(cf: Bushman, B. J., & Bonacci, A. M. (2002). Violence and sex impair memory for television ads. Journal of Applied Psychology, 87(3), 557-564.)
(cf: Droulers, O. & Roullet, B. (2004). Does crime pay for violent program-embedded ads? Advances in Consumer Research, 31, 1-6.)

(cf: C. Shawn Green & Daphne Bavelier (2003); Action video game modifies visual selective attention, *Nature* 423, 534-537 (29 May 2003))
(cf: Yang, M., Roskos-Ewoldsen, D. R., Dinu, L., & Arpan, L. M. (2006). The effectiveness of "in-game"advertising. Comparing college students' explicit and implicit memory for brand names. Journal of Advertising, 35(4), 143-152.)
(cf: SMI SensoMotoric Instruments http://www.smivision.com/, http://www.smivision.com/fileadmin/user_upload/downloads/product_flyer/prod_smi_insight.pdf)

Overall applicability and future of the JMFEL

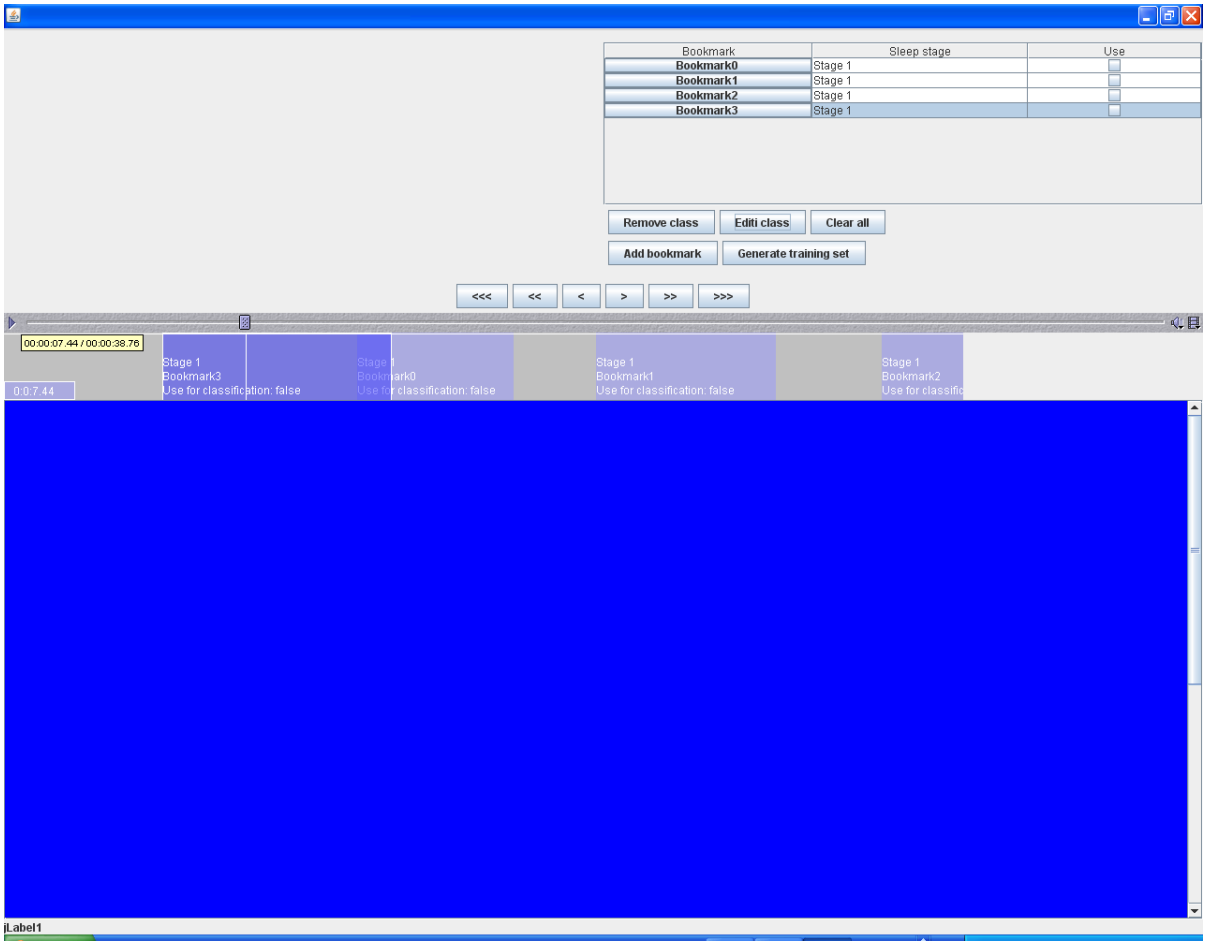
JMFEL, the extension to the Java Media Framework presented in this thesis makes JMF based multimedia programming considerably easier and a lot less error prone. Right now, the JMFEL is the most complete JMF extension ever written. Its' simple design and relatively small number of easy to use core classes make it possible to develop complicated multimedia applications within a few hours instead of days or even weeks. This is because the developer doesn't have to think about complicated details anymore.

The examples and experiments have demonstrated how JMFEL applications are developed and that almost anything can be made into a streaming data source – even whole applications. The JMFEL classes themselves don't need a lot of improvement. The graphics framework should use OpenGL drawing surfaces soon, since this will significantly reduce the overall CPU utilization and consequently make the JMFEL more interesting for game research.

As already stated in the discussion about the *InsightDatagramPacketDecoder* design issues (cf: JMFEL support classes section), more and advanced design patterns should be used to make the code even more stable and safer to use. Those strict implementation standards and guidelines make team programming easier and add even more robustness. Consequently, detailed knowledge about JMF and JMFEL as well as multimedia programming becomes less important, even enabling less skilled developers to be employed in JMF programming.

Manual Editing

Right now the, JMFEL mainly supports data acquisition and data processing and it does that very well. What is still missing are ready-to-use tools for manual and / or automatic analysis. So far, only a very early design study exists. The idea is to have a time line where areas can be selected, labelled and edited with the mouse. In this example the areas represent different sleep stages (cf: screenshot). The finished application would also show the respective EEG channels as well as an optional video image in the top left. All areas and classes should be editable. The “Generate training set” button is supposed to generate a training set, a labelled CSV file e.g., which can then be used to generate the classifier of choice.



Automatic Editing

Manual offline editing of several hours-long recordings is an error-prone and tedious process, which might take several hours to finish. While this does make sense where human experience is required, it is absolutely unnecessary if the length and the position of an EEG interval corresponding to a certain event is already known.

There are at least two possible ways to employ automatic editing. Both methods will use the very same codec plug-in, which cuts out predefined portions of the EEG signal the moment an event has occurred and/or finished. These portions are then stored somewhere to undergo some type of analysis process. The only difference between the two methods is that this analysis can happen either offline, after signal acquisition has finished, or online, while the signal is still being acquired. No matter if online or offline processing and analysis is performed, the final result – some sort of meaningful pattern – will be identical.

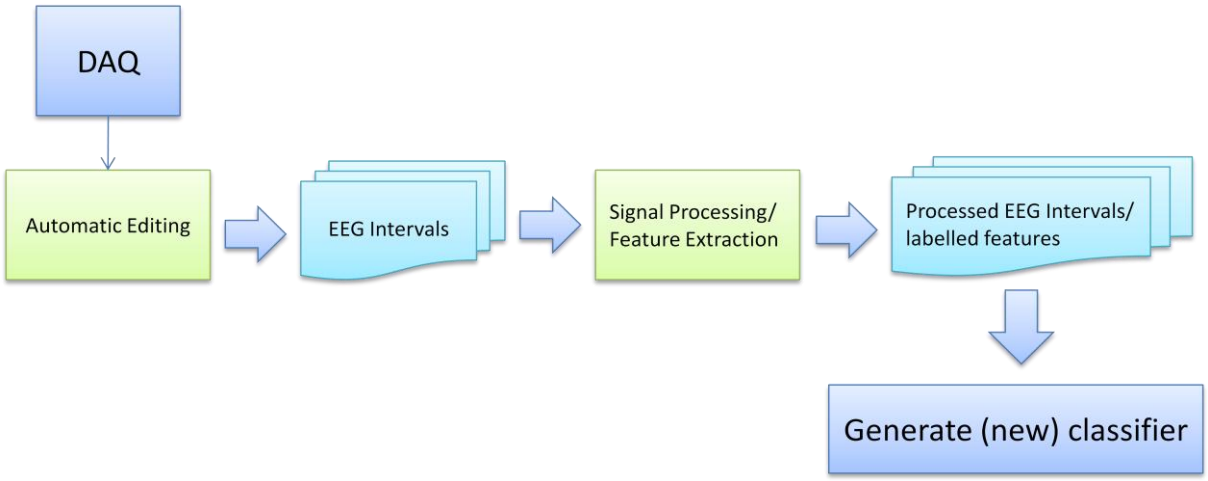
As events occur, event tracks are used to store the event's duration and position in time. A codec-plug-in is then used either online or offline to determine the duration and temporal position of the corresponding EEG intervals. These EEG intervals can then be processed either incrementally – while data acquisition is still running – or offline, after recording has finished. (see also Synchronization experiment 2)

Analysis example

After m events of type red and type green were gathered by a codec plug-in, some processing and analysis algorithm has produced corresponding and meaningful patterns.

Idea for a JMFEL-BCI

The idea is to perform data acquisition and use automatic editing and data analysis as explained earlier, to incrementally generate and improve a classifier, which can detect/predict a certain event. Once the classifier is accurate enough, it can be applied to sufficiently long chunks of the live EEG signal, which have undergone the very same feature extraction process used during the classifier's creation.



While data acquisition is running, EEG intervals, which correspond to certain events are extracted, further processed and labelled. If a significant amount of labelled features is available, a (new) classifier is generated. The longer the acquisition process takes, the more labelled features are available, and the better the generated classifier will be.

Almost all components for this BCI are already implemented and available. Cutting out pieces of EEG signals e.g. from one second before an event has happened to one second after an event has ended, is quite trivial. The real challenge is the feature extraction algorithm, whereas the classifier can easily generated using the tools provided by the Dendronic Learning Engine API or other Classification APIs.

Appendix A

How to use the Java tools for the g.USB Amp

This text is also part of the JMFEL distribution.

All tools use command-line parameters to specify both, the JMF software device and the physical device.

The two parameters are:

-serial and –device

The serial can be found on the label on top of each amplifier. The device can be usbAmpA, usbAmpB, usbAmpC or usbAmpD

For purposes, only the device usbAmpA is used in this howto.

As an example, let us use the following command line to start the the DaqWizard Tool:

java media.protocol.gtec.DAQWizard -serial UA-2007.04.01 -device usbAmpA

This command (along with its' parameters) writes the settings generated by the DaqWizard tool into the folder java media.protocol.usbAmpA, where the software device usbAmpA (utilized by the JMF) is located.

The folder java media.protocol.usbAmpA also holds a file called serial.ini which includes the serial number to be opened by the device usbAmpA.

The serial used in the command line and inside the *serial.ini* DO NOT have to be the same!

The serial number has to be of an application that uses the software device.

You could either use the serial number of an amplifier connected to the computer or employ the serial number of an application that is using the software device. You could then give it to a colleague and do the actual measurements using a different amplifier, e.g. with serial number ***UA-2007.04.02***. Just change the serial number ***serial.ini*** file and you are good to go.

For the tools “CalibrationWizard” and ImpedanceWizard, the serial used in the command line has to be identical to the serial used in the ***serial.ini*** file, because these tools will change

settings inside the amplifier that you want to use during the measurement. You will usually use these tools right before measurements.

The “ScalingWizard”, however, doesn’t need a serial number at all, since the scaling values written by this tool are just for visualization. Hence, you only provide the name of the software device e.g. ***usbAmpA.*** Every application that makes use of the software device usbAmpA will then use these settings.

Here are examples for calling these tools:

java media.protocol.gtec.CalibrationWizard -serial UA-2007.04.01

starts the Calibration Wizard using the amplifier with serial UA-2007.04.01

java media.protocol.gtec.DAQWizard -serial UA-2007.04.01 -device usbAmpA

starts the DaqWizard using the amplifier with serial UA-2007.04.01 to configure settings used by the device usbAmpA.

java media.protocol.gtec.ImpedanceWizard -serial UA-2007.04.01

starts the Impedance Wizard using the amplifier with serial UA-2007.04.01

java media.protocol.gtec.ScalingWizard -device usbAmpA

starts the Scaling Wizard for device usbAmpA.

Command lines such as those above are used by the batch files:

startCalibrationWizard.bat

start DaqWizard.bat

startImpedanceWizard.bat

startScalingWizard.bat